Information Fusion for Automated Question Answering

Tiphaine Dalmas

Doctor of Philosophy
Institute for Communicating and Collaborative Systems
School of Informatics
University of Edinburgh
2006

Abstract

Until recently, research efforts in automated Question Answering (QA) have mainly focused on getting a good understanding of questions to retrieve correct answers. This includes deep parsing, lookups in ontologies, question typing and machine learning of answer patterns appropriate to question forms. In contrast, I have focused on the analysis of the relationships between answer candidates as provided in open domain QA on multiple documents. I argue that such candidates have intrinsic properties, partly regardless of the question, and those properties can be exploited to provide better quality and more user-oriented answers in QA.

Information fusion refers to the technique of merging pieces of information from different sources. In QA over free text, it is motivated by the frequency with which different answer candidates are found in different locations, leading to a multiplicity of answers. The reason for such multiplicity is, in part, the massive amount of data used for answering, and also its unstructured and heterogeneous content: Besides ambiguities in user questions leading to heterogeneity in extractions, systems have to deal with redundancy, granularity and possible contradictory information. Hence the need for answer candidate comparison. While frequency has proved to be a significant characteristic of a correct answer, I evaluate the value of other relationships characterizing answer variability and redundancy.

Partially inspired by recent developments in multi-document summarization, I redefine the concept of "answer" within an engineering approach to QA based on the Model-View-Controller (MVC) pattern of user interface design. An "answer model" is a directed graph in which nodes correspond to entities projected from extractions and edges convey relationships between such nodes. The graph represents the fusion of information contained in the set of extractions. Different views of the answer model can be produced, capturing the fact that the same answer can be expressed and presented in various ways: picture, video, sound, written or spoken language, or a formal data structure. Within this framework, an answer is a structured object contained in the model and retrieved by a strategy to build a particular view depending on the end user (or task)'s requirements.

I describe shallow techniques to compare entities and enrich the model by discovering four broad categories of relationships between entities in the model: equivalence, inclusion, aggregation and alternative. Quantitatively, answer candidate modeling improves answer extraction accuracy. It also proves to be more robust to incorrect answer candidates than traditional techniques. Qualitatively, models provide meta-information encoded by relationships that allow shallow reasoning to help organize and generate the final output.

Acknowledgements

I do not have the words to fully express my gratitude to my Ph.D. supervisor, Professor Bonnie Webber, for her contagious passion for her work, and her most helpful guidance. Throughout my study, she provided encouragement, sound advice, emotional support, and spent an invaluable amount of time reviewing my writing. I would like to thank her for giving me the opportunity to carry out this research, and her constant availability when I needed her input.

I would like to express my sincere appreciation to my second supervisor, Johan Bos, for his support and help in writing.

I also very much appreciated the suggestions for improving my dissertation given by my examiners, Professor Johanna Moore and Professor Maarten de Rijke. I am grateful to Ewan Klein, whom I first contacted at the University of Edinburgh when I was a prospective student, and who gave me useful pointers. I would like to thank him and Mirella Lapata for accepting the role of examiners during my Dissertation Draft Defense, and providing me with important comments and feedback.

I am also grateful to the researchers with whom I have been able to discuss my research: Regina Barzilay, Carsten Brockmann, Sébastien Carlier, Mark Andrew Greenwood, and Kisuh Ahn. I would especially like to thank my fellow Ph.D. student, Jochen Leidner, for his thoughtful and creative comments, as well as his support when I arrived in Scotland.

I wish to thank my friends, Ami Bender, Violette Grinnaert, Julien Dufour, Stéphane Jeanty and Franck de La Bretèche for all the emotional support, caring and entertainment they provided. Special thank as well to my fellow aikidokas at the Takemusu Aiki Club in Edinburgh who helped me relax at times of mental blurs.

Finally, I would like to express thanks to the School of Informatics and the Institute for Communicating and Collaborative Systems for the financial support without which writing this thesis would not have been possible. I very much appreciated spending my Ph.D. years in Edinburgh, not only because of the high-quality environment offered by the University of Edinburgh, but also because I found the city to be a fantastic place to live in.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Tiphaine Dalmas*)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The purpose of automated Question Answering (QA) is to produce an answer based on one or more sources of information to a question expressed in natural language. In state-of-the-art open domain QA on free text, answers comprise one or more relatively short strings that have been extracted from a supporting document.

[Hirschman and Gaizauskas, 2001] showed there had been much progress in the field, especially in fact-based QA for which the best systems can correctly answer more than 80% of the questions with very accurate extractions. However, they notice: *For first time users, it may be important to explain the limitations of the system, so that the user can understand how to interpret the answers returned.*

One such limitation is that the QA task involves extracting strings from free text. In free text, answers may appear in many forms and different contexts. When the string is extracted, the local context is cut off, making the resulting string sometimes difficult to understand. The user has to browse the supporting document to understand the answer. This defeats a main objective of QA, in contrast to Information Retrieval (IR), which is to provide short, self-sufficient results rather than require users themselves to peruse full documents.

Another such limitation is that QA performance, despite recent improvements, is still low. The chances are still high that an incorrect answer appears in the list of results. Because the context that could explain and justify an answer has been stripped off, the user may have difficulty identifying when an extraction is a system error or simply an unexpected alternative answer. A list of extractions is rarely a satisfying choice to acknowledge answer multiplicity to the end user because it does not indicate the kind of relations between the list elements (e.g. alternative answers or not).

In one word, answers from current QA systems are *unstructured*. This means that (1) their interpretation is not straightforward to human end users, and (2) it is difficult to reuse them for other applications or QA extensions such as dialog-oriented QA.

While mainstream QA is mainly concerned with the correctness of a question-answer pair, this thesis focuses on multiple answers and their relationships (answer comparison). The main motivation for answer comparison is to enhance the performance of QA, and, using the structure emerging from comparison, to provide more user-oriented answers, i.e. answers that can be more easily exploited by the end user, be it a human user or an application. In this thesis, I show that answer comparison improves the accuracy of a QA system, and the structure emerging from comparison allows for a more flexible presentation of answers to the end user.

In Chapter 2, I show that questions posed in open domain QA often have multiple answers, and such multiplicity is due both to properties of the resources used in QA and to properties of questions. The first refers to (1) the massive amount of data used for answering, and (2) its unstructured and heterogeneous content: This means redundancy, granularity and possible contradictory information. The second refers to the fact that ambiguities in user questions can lead to heterogeneity in what is extracted.

In recent work, the frequency of an extraction has proved to be a significant characteristic of a correct answer, I review the value of other relationships to characterize answer variability, and techniques that have been successfully used for merging and structuring data in related fields such as automated summarization and clustering. . The last section of this background chapter is dedicated to the evaluation methods and measures currently used to assess QA performance.

In Chapter 3, partially inspired by recent developments in multi-document summarization, I redefine the concept of "answer" within the Model-View-Controller (MVC) pattern of user interface design. An "answer model" is a directed graph in which nodes correspond to entities projected from extractions and edges convey relationships between such nodes. The graph represents the fusion of information contained in the set of extractions. Different views of the answer model can be produced, capturing the fact that the same answer can be expressed and presented in various ways: picture, video, sound, written or spoken language, or a formal data structure. Within this framework, an answer is a structured object contained in the model and retrieved by a strategy to build a particular view depending on the end user (or task)'s requirements. The advantage of the MVC is that it distinguishes three aspects, or development stages of an answer: (1) the answer as found in the original document, i.e. a string, (2) the answer as a structured object on which inference is performed, and (3) the answer as presented to the end user, which in mainstream QA is identified with (1), i.e. a string, but could actually be a cluster of strings, a summary or a picture, depending on the task. In the same chapter, I review the scoring measures I will use to assess the performance of fusion, some of them traditionally used to assess single answer re-ranking, and new ones I propose to evaluate re-ranking of answer clusters.

Quantitatively, I show that, for location questions, answer comparison improves the accuracy of the results. It also proves to be more robust to incorrect answer candidates than traditional techniques. In Chapter 4, I report an improvement of 23% in first ranked answer accuracy for location questions. Qualitatively, answer models (in MVC terms) provide meta-information encoded by relationships that allow shallow reasoning to help organize and generate the final output. In Chapters 5 and 6, I assess the value of fusion on a greater variety of questions using this time answer candidates collected from web searches and manually rated, and compare evaluation results for reranking of single answers, but also answer clusters, in order to demonstrate the flexibility provided by graph-based modeling. I also report a first experiment with machine learning using features computed from graph models.

Finally, in Chapter 7, I summarize the contributions of this thesis, and discuss issues related to the computation of answer models, and address the difficulty of assessing fusion-based answers with the current evaluation methods in QA. I propose new tasks to address the specific challenges raised by considering answer complexity.

Chapter 2

Information Fusion and Answer Comparison

The QA task has evolved since its beginning from natural language front end to databases towards working over unstructured data. In this chapter, I first review the challenges that are specific to open domain QA on free text, focusing on two characteristics that this thesis is concerned with: answer variability and multiplicity.

The two leading objectives of this thesis emerge as (1) using multiple answers to increase accuracy of QA systems, and (2) using graph-based modeling as a flexible representation that allows for different types of answer re-ranking and answer presentation. The following items are reviewed:

- the assessment of answer multiplicity.
- what can be done with multiple answers, i.e., in this thesis, the value of answer comparison and how it is linked to the type of answers envisaged in the QA roadmap.
- the techniques used for answer retrieval which affect answer multiplicity.
- the techniques for answer comparison, i.e. what the relationships between answers are, and how such relationships can be inferred, using what kind of representation.

While question analysis has long been a stronghold of the field, I focus on the kind of answer that is envisaged in the QA roadmap. This "next generation of answers" goes beyond finding a correct answer string to a question, and aims at structuring a response that acknowledges answer variability and multiplicity to the end user.

In the second section, I review existing taxonomies and retrieval strategies in QA, from top-down strategies driven by the question to more data-driven approaches, as well as recent work on answer fusion for re-ranking and on the relationships between answers. The third section of this chapter is dedicated to techniques that have been successfully used for merging and structuring data in related fields such as automated summarization and clustering. Finally, I review the evaluation methods and scoring measures currently used in QA, and discuss issues relating to the assessment of fusion-based answers.

2.1 Overview of the Task from the Answer Side

The goal of QA is to automatically provide a satisfying answer to a user asking a question in natural language. Depending on the type of question, the resources used for answering, and what we define as "user's satisfaction", the field can cover a wide variety of tasks. Among these three elements (question/answer/user), the question side is the one that has most benefited from research so far (e.g. in question classification, in-depth analysis, reformulation in database query or search engine query). The notion of a satisfying answer, and user satisfaction in general, is almost nonexistent. Although attention has been paid to answer candidate retrieval and answer selection or ranking, answers are still simply evaluated as *correct* or not. This thesis focuses on answers and their relationships. I will demonstrate that answers have intrinsic properties, regardless of the question, and those properties can be exploited to provide better quality and more user-oriented answers in QA.

In the first subsection, I remind the reader of the shift in the data used for answering that occurred with the technical improvement of machines in terms of available memory and processor speed, and the emergence of the Internet providing a huge repository of data. The task has evolved since its origins in a natural language front end to structured databases. QA used to be performed on structured, homogeneous data, and the trend is now on unstructured, heterogeneous, free text. The impact of this shift has mainly been on the type of answers retrieved, rather than the type of question that can be asked.

In the second subsection, I provide statistics on the main characteristics to be dealt with in free text QA: the multiplicity and variability of answers found. I show that this multiplicity does not necessarily depend on the type of question asked, contrary to what is assumed in the TREC QA evaluation.

In the third subsection, I present the kind of answers we could aim for if we go beyond the paradigm of answer *correctness* and aim instead at satisfying a user's need. Such answers are tied to recognizing answer variability and user modeling.

2.1.1 Specific Challenges in QA over Free Text

Although the purpose of QA – to provide an answer to a user's question given a source of information – may sound straightforward, the definition actually encompasses a wide range of tasks, and somewhat different paradigms.

Automated QA began as Natural Language Database QA (NLDQA) in the 1960's [Simmons, 1965, Green et al., 1961, Woods, 1968, Woods et al., 1972], as a convenient alternative to formal query languages for people who were not database experts. Work followed on dialog systems and story comprehension until the late 1980s. In the mid-90s, interest in QA re-appeared through the Text REtrieval Conference (TREC) designed and organized each year by the American federal technology agency NIST [Harman, 1992, Voorhees and Tice, 1999]. With the explosion of raw documents available through networking and Internet, there is a need for systems capable of handling large sets of free or semi-structured text documents. The TREC QA task proposed to tackle this need by evaluating systems operating on three gigabytes of documents from newspapers. From then on, web QA was described as an *urgent* need [Hirschman and Gaizauskas, 2001], while interest in Reading Comprehension [Hirschman et al., 1999, Riloff and Thelen, 2000], which goal is to assess automated text understanding through question answering, has receded. In 2003, the European Cross Language Evaluation Forum (CLEF) carried out its first non-English, monolingual and cross-language QA tracks [Magnini et al., 2003]. It has been a yearly evaluation since then. The more recent INEX annual conference [Fuhr et al., 2003, Ogilvie, 2004] has also introduced a QA task over XML data (NLP task), treating the task as one of producing appropriate XPath queries over semi-structured data from a query in natural language.

While the purpose of QA has not changed, the emergence of tasks based on free text has slightly shifted the original approach. In NLDQA, questions are answered by translating them into a formal database query, evaluating the query against the database, and then embedding the results in a response that reflects the original question. In contrast, in free text QA, a question is mapped to a Boolean combination of query terms and/or regular expressions (for string matching), and fed into a search engine to retrieve documents or passages considered relevant to the set of keywords. Answer candidates are then extracted from those retrieved texts and rank-ordered, with the top-ranking candidate chosen as the answer. In NLDQA, the tight coupling between data and the process of translating user questions into queries has been used to recognize and eliminate *ambiguities* in user questions **before** database access. For example, an NLDQA system will need to recognize that the object of the verb "read" could be either a title, or an author, in order to select information in the appropriate database field. In QA from unstructured data however, the coupling between query formulation and data is much looser, delaying ambiguity recognition (if any) until **after** data access. Hence, while an *a priori* understanding of the question was a requirement in NLDQA, the focus has shifted towards more data-driven approaches in QA over free text. Following TREC QA 10, [Voorhees, 2001] comments:

> Many groups continued to build systems that attempt a full understanding of the question, but increasingly many groups took a more shallow, data-driven approach. The data-driven approach relies on simpler pattern-matching methods using very large corpora (frequently the web) rather than sophisticated language processing. The idea exploited in the massive data approach is the fact that in a large enough data source the answer will usually be repeated often enough to distinguish it from the noise that happens to occasionally match simple patterns.

The shallow approach has proved to be successful: The winner of TREC 10 was a shallow system based on a collection of answer patterns [Soubbotin and Soubbotin, 2001].

Nonetheless, in TREC QA, as in Reading Comprehension, web-based QA and other tasks over free text, redundant answers also present significant problems because they may appear in many places and in many forms. The redundancy that has helped web-based QA systems answer factoid questions [Banko and Brill, 2001, Brill et al., 2002, Magnini et al., 2002a, Light et al., 2001] exploits the fact that the *exact same answer* may appear many times over in the vast collection that constitutes the Web. But in general, the lack of controlled vocabularies and the absence of data typing on the web means that even the "same answer" may be found *in many forms whose equivalence is not immediately obvious*. This problem of distinguishing one truly distinct answer from another is one reason that systems have difficulty answering list questions (i.e. questions with multiple answers).

Besides, in QA over large heterogeneous collections (such as the web), documents often contain *disagreements* – e.g. different attestations as to which date Louis Armstrong was born. Web data are also volatile. Some documents are rewritten, new documents appear, while others are deleted, resulting in continual alteration of the search corpus. In contrast, contradictions are absent from traditional NLDQA systems, wherein consistency is a property of the database and it is acceptable to make a closed world assumption.

TREC QA evaluation of system performance has, to date, ignored these issues. Instead, it simply allows for different answers by providing a question with more than one answer pattern. These answer patterns match *extractions* – substrings contained in the source document. So multiple answer patterns conflate *inter alia* (1) referential ambiguities or contradictions that systems are not meant to adjudicate (e.g. different values found for the population of Surinam, different locations for the Taj Mahal); (2) answers at different granularities (e.g., Glasgow vs. Scotland) or in different metric systems (e.g., Fahrenheit vs. Celsius); or (3) different aspects of the concept being questioned (e.g., Desmond Tutu being "Bishop of Cape Town" vs. "an anti-apartheid activist"). While this variety is now acknowledged for "other" questions at TREC (questions about facts of interest with respect to a given topic), this reality needs to be acknowledged for the whole range of questions.

The challenge, besides automatically understanding a user's question, is to deal with unstructured, heterogeneous and volatile text. While most of the research in QA has focused on the analysis of the question, and how to retrieve a correct answer to that question, this thesis focuses specifically on answers themselves. Instead of considering relationships between the question and candidates answers only, I will also consider the answering material (the data set used for answering) as a whole with properties on its own, i.e. *redundancy*, *granularity*, and possible *contradictory information*, hence performing answer comparison to detect answer variability. In the next subsection, I will show that answer variability, as witnessed by answer multiplicity, does not necessarily correlate with a massive amount of data nor with specific question types. It is a more general problem that needs to be addressed.

2.1.2 Answer Multiplicity

Intuition suggests that answering certain types of questions could benefit from answer comparison because of the different possible ways of answering them.

To check this, I investigated cases where different extractions were considered acceptable answers to questions in TREC QA [Voorhees, 2002], and in a corpus of Reading Comprehension tests produced by the MITRE corporation based on texts from CBC4Kids [Light et al., 2001]. Both TREC and CBC4Kids use a collection of newspapers as answering material. Both corpora are described in more depth in Subsections 2.4.1 and 2.4.2. It may be noted that CBC4Kids is intended for a young audience, and specifically aimed at Reading Comprehension rather than open domain questions.

In CBC4Kids, answers have been written down by evaluators and I counted the number of answers per question. For TREC QA, the percentage of multiple answers was calculated using the patterns provided by NIST judges to evaluate systems. These patterns are regular expressions, one for each similar answer (in terms of pattern-matching). Each separate line of patterns was counted as a separate answer. For instance, the question *What is epilepsy?* is evaluated with the list of patterns shown in Table 2.1.

Each pattern line can match a great variety of answers. The first pattern, for example, can match *neurological disorder* as well as *brain disorder*, *nervous system disorders*. Figures counting patterns are actually an *under-count* of multiple answers in TREC QA as a regular expression may match several distinct answers.

Table 2.1: Answer patterns using the syntax of regular expressions for *What is epilepsy?*

(neurological|brain|seizure|paroxysmal|underlying|mental|nervous system) disorders?
seizures?
nervous system
disorders?
affliction
speech loss
convulsions?

? indicates the previous character may or not appear

(A|B|C) indicates that strings A, B, and C are valid alternatives

Besides, the list of TREC patterns is not exhaustive. Patterns are derived from the correct answers strings found by automated systems. They do not stand for all the possible answers actually available in the corpus. Such collection would involve a far too large amount of human work. Hence the difficulty of having a proper recall measure in QA tasks over large corpora, as well as exhaustive data on multiple answers. The figures shown in Table 2.2 are thus to be taken as a lower bound regarding the actual distribution of multiple answers.

Table 2.2: Distribution of multiple answers in TREC QA and CBC4Kids

	TREC 8	TREC 9	TREC 10	TREC 11	CBC
Question counts	200	693	500	500	481
No answer	2	11	67	56	0
Single answer	129	304	211	378	173
Multiple answers (MA)	69	378	222	66	308
% MA	34.5%	54.5%	44.4%	13.2%	64%

The first thing to note is that the proportion of multiple answers in TREC 11 is between

2.6 and 4.1 times lower than the previous tracks (TREC 8, 9, 10). This is because systems in TREC 11 were required to give only one answer per question from TREC 11 on (as opposed to a rank-ordered list of the top five answer candidates).

TREC QA distinguishes between two main types of questions, factoid questions and definition/list questions (later refined in list, definition and "other" questions). Factoids are questions focused on trivia, such as *Where is the Taj Mahal*, *When was the telegraph invented?*. Definition questions involve queries asking for a definition (*What is bipolar disorder?*), or biographic information (*Who was Galileo?*), or an explicit list (*Who are 6 actors who have played Tevye in "Fiddler on the Roof"?*). According to Voorhees [Voorhees, 2002], the target answers are more difficult to assess for definition questions. Definition questions were thus removed from the main QA track at TREC 11. Factoid questions were kept, but systems were required to provide only one answer per question. The underlying assumption is that definition questions tend to generate more multiple answers than factoid questions.

However, despite the rule of one answer only for factoid questions, the proportion of multiple answers is still non-negligible for factoid questions. It is also *not* proportional to the size of the corpus (around 3GB for TREC questions, 500 words for CBC4Kids questions).

The original intuition concerned *types* of questions which could be answered in more than one way. As mentioned above, since TREC 11 [Voorhees, 2002], systems are required to provide only one answer to factoid questions. There is a special set and scoring system for list, definition and "other" questions to which several answers are allowed. To gather data on this distinction, I automatically classified TREC questions before TREC 11 (from TREC 8 through TREC 10) by their wh-word (interrogative word) to identify the most obvious factoid questions (*where*, *when*, *how much*, *how many*) as opposed to non-factoid questions (*why*, *how*). Then, I manually classified the questions into six classes, representing around half of the initial corpus. The classes *when*, *how-adj/adv* and *where* represent factoid questions. The class *who/famous-for* stands for biographic questions, and, along with the classes *why/cause-effect* and *definition*, represents non-factoid questions ("definition" questions).

Table 2.3: Distribution of multiple answers per question class

class	# of questions	No answer	Single answer	Multiple answers
who/famous-for	215	3.2%	56.3%	40.5%
when	93	4.3%	51.6%	44.1%
how-adj/adv	107	5.6%	47.7%	46.7%
where	118	1.7%	33.8%	64.5%
why/cause-effect	16	0%	25%	75%
definition	120	5%	14.2%	80.8%

Table 2.3 shows that definition and cause/effect questions in most cases have more than one acceptable answer. But the proportion is high for all six question types, including factoids, especially location questions (*where*). The recent TREC distinction between factoid (one answer) versus list questions is thus somewhat artificial.

Let us now see the effect of the rule "one answer per factoid" on the subsequent TREC evaluations. Table 2.4 shows the distribution of multiple answers for factoid questions only (i.e. questions for which systems are required to provide one and only one answer), at TREC 11, 12 and 13. These answers are the answers accepted as correct by TREC judges.

Table 2.4: Distribution of multiple answers for TREC 11, 12 and 13 factoids

	TREC 11	TREC 12	TREC 13
Question counts	500	413	230
No answer	56	33	28
Single answer	378	312	187
Multiple answers (MA)	66	68	15
% MA	13.2%	16.5%	6.5%

There is an expected drastic shrinkage in the number of multiple answers for factoid questions. Given the previous statistics on factoid types (Table 2.3) indicating that an average of 51.7% of factoid questions have multiple answers (65.4% for non-factoid questions), if systems had been allowed to continue to return a rank-ordered list of candidate answers to each question, it is highly probable that the distribution of multiple answers would have been much higher (because more systems would have returned at least one, and possibly several, correct answer candidates among their top five). The main issue with the rule on factoids is that the existing pool of answers is not representative of the variety of answers available in the corpus, and therefore difficult to reuse for further research.

One way of understanding the current position of TREC QA is the focus chosen for the task, which is oriented towards information extraction. The chosen paradigm is to fill in a question slot with a correct piece of information (a string), rather than view answers as complex objects. The *ad hoc* nature of this decision has been noted by [Voorhees and Tice, 1999]: *Currently judgments are based on entire answer strings because it is not yet clear how to map specific answer strings into more general answers*. The difficulty of engineering *general answers* is actually a strong impediment to further advances in QA. The current trend in TREC QA is mainly question-oriented: The goal is to provide correct information (correct strings) to more and more elaborate questions. The complexity of the task is related to the complexity of the question: *What kind of questions are automated answering systems good at?*[1] My approach is slightly different, and the question I ask is: What kind of answers are automated answering systems good at? Is it possible to provide results that acknowledge answer multiplicity and complexity, and thus depart slightly from a perspective of QA as an information extraction task. The next subsection provides an example of what is expected to be a next generation answer and shows how it is tied to answer variability detection.

2.1.3 Next Generation of Answers

In their "Roadmap" to future research and development in QA, [Burger et al., 2002] present what they consider a desirable answer to the question *Where is the Taj Mahal?*:

[1]This question was asked to the panel associated with the Knowledge and Reasoning for Answering Questions workshop at IJCAI 2005.

If you are interested in the Indian landmark, it's in Agra, India. If instead you want to find the location of the Casino, it's in Atlantic City, NJ, U.S.A. There are also several restaurants named Taj Mahal. A full list is rendered by the following table. If you click on the location, you may find the address. [Table omitted]

To produce such an answer, systems need to have solved problems of *presentation* and *ambiguity*. With respect to *ambiguity*, Burger et al's answer to *Where is the Taj Mahal?* shows that a system has to recognize that its answers fall into distinct equivalence classes that can be organized into a structured answer of alternative possibilities. From such analysis, it would then be possible to generate an adequate rendering, e.g. a summary and a table of addresses.

Current QA systems cannot produce Burger et al's answer because what they return are rank-ordered *extractions* such as: *Agra*; *the city of Agra*; *Atlantic City, NJ*; *India*, each with a pointer to its source document. To move further towards the kind of answer that [Burger et al., 2002] envision requires solutions to the following problems:

1. *Identifying multiple and/or complex answers*, which may depend on how ambiguous the question is with respect to the corpus, and/or how informative the corpus is on a given topic (e.g. A system may find a more detailed answer to *What is epilepsy?* in MedLinePlus [2] or other consumer health web sites than in the AQUAINT or Reuters corpus of news text.) Also, complex answers have not always been anticipated in a single text. Parts of the answer may occur independently, across documents or across different sections of a document, in which case the answer has to be *reconstructed* by analyzing the *relationships* occurring between nuggets of information found in different places.

2. *Deciding on the amount and kind of information to be presented.* The end user may prefer a detailed answer to a short answer, or the most frequently occurring answer. But in addition to user preferences, one also needs to consider the amount of additional *context* that is needed in order for the user to understand the answer, e.g. it is not enough to say that there is a Taj Mahal in Atlantic City

[2]http://medlineplus.gov

without providing the context that it is a casino, or that there is a Taj Mahal in Springfield, Illinois, without saying that it is a restaurant. In TREC QA, the rule is to interpret systematically such an ambiguous question as a "famous place" trivia. However, it is difficult to evaluate whether systems have top-ranked *India* after this decision, or because it just happened that *India* was the most frequent occurrence around *Taj Mahal*. Depending on the corpus, term frequency may not always correlate with the fame of the entity in question. Hence, my position is to first try to identify multiple answers and then make user-related decisions based on comparison.

3. *Deciding on presentation modality.* There are many, even simple, factoid questions for which text alone is not the best medium. For instance, the answer to a question such as *Where are diamonds mined?* would probably be best rendered with a map (which makes spatial patterns clear). Although the main problem of the QA community is still how to *obtain* correct answers rather than how to *render* them, it is still a worthwhile and interesting question to pursue. Besides, with elaborated answers such as summaries or dialog-oriented answers, it is likely that systems will require more context and more information about the structure of the answer as a whole. Hence again the need for a better understanding of the answering material.

4. *Evaluating answers that go beyond text extractions* – i.e., where simple regular expression pattern matching is no longer sufficient because string matching does not reflect more complex relationships such as synonyms or alternative phrasing.

While solving all these items is beyond the scope of this thesis, I will focus on the automated discovery of relationships between answer candidates, which is the necessary step before further advances towards user-oriented responses involving reasoning on a set of answers.

The following section reviews previous research in answer retrieval strategies and reflects on the relational aspect of the QA task, whether it be between a question and its answers, or between answers themselves. The subsequent section will focus on the techniques used in related areas to identify relationships and structure data that was unstructured in its original form.

2.2 QA Taxonomies and Retrieval Strategies

Past research in QA shows a considerable amount of work related to classifying questions and their relationships with answers, from a cognitive, linguistic, as well as an engineering point of view. I will review relevant kinds but focus on the latter. Although *in order to answer a question, the question must be first understood* [Burger et al., 2002], I will argue that, on the engineering side, it does not systematically translate to deep parsing and full understanding of the question at the start of the pipeline. Having a better understanding of potential answers as they appear in the corpus might give an *a posteriori* complementary help to question understanding. I will connect difficulties of automated answer retrieval to known problems in the community of (human) library reference.

2.2.1 Question taxonomies

Question analysis is a historical stronghold in QA. The challenge of question classification and typing, in terms of engineering, is to attempt to guess the answer before hand, to help select the correct database field in NLDQA, or prune the search collection in free text QA. Within this approach, the question is interpreted once before answer retrieval, and this interpretation is never challenged nor refined. At best the question may have multiple types that define relaxing constraints. For instance, if no answer candidate matches a *college name*, the answer type might be relaxed to a *location*. The question is not refined but the interpretation might be altered from specific to general.

In contrast, it is widely acknowledged, in the domain of library reference, that the first question asked by a user is usually not well-phrased and question answering is a process involving a refinement of the original question:

> [Belkin et al., 1982] well-known ASK hypothesis states that an individual's information need arises from an "anomaly" or gap in that individual's "state of knowledge," but that the individual is generally unable to articulate what that gap is, or what would be required to fill it. [Pomerantz, 2005]

Hence, the reference librarian has often been referred to as a *mind-reader*. Pomerantz follows with a statement that is interesting because he is somewhat an outsider to the field of QA: *While QA treats questions as something to which one and only one correct answer exists, library reference acknowledges that what is an acceptable answer varies depending on the patron's current situation.* And further, *while different individuals may formulate similar or even identical questions, the content and form of an answer that will be useful to these individuals may differ depending on the different situations in which these individuals find themselves.* From an outsider's point of view, it seems the QA task is to seek the right answer *per se*, rather than providing satisfying information in a given context. User profiling and interacting are known paradigms in QA, however they have for the most part been ignored. It is a thought that perhaps we are making the task harder than it ought to be by not integrating any interaction or feedback process[3].

Quoting Pomerantz again: *A question cannot stand on its own, but through conversation may be disambiguated and ultimately responded to.* As a matter of fact, a "question taxonomy" is most often a classification of *expected* answers. However, although such *a priori* classifications help prune the search space, *a posteriori* disambiguation strategies are also possible, if user dialog is yet difficult to achieve automatically. What I discuss next is strategies to improve analysis and reasoning over answer candidates to further refine the interpretation of the question in QA.

[Pomerantz, 2005] reviews existing question taxonomies in the fields of QA, desk and digital reference, and linguistics, which appear to have taken slightly different approaches. He distinguishes five types of question/answer taxonomies, which he associates with the top four levels of linguistic analysis, as reproduced in Table 2.5: pragmatic, discourse, semantic and syntactic levels.

[3]This is currently discussed in the TREC QA community, and the TREC 2006 ciQA (complex, interactive QA) has been introduced to integrate interaction forms to the task.

Table 2.5: Levels of Linguistic Analysis in Question Taxonomies

Linguistic Level	Question Taxonomy
Pragmatic	Types of sources from which answers may be drawn The forms of expected answers to questions
Discourse	The functions of expected answers to questions
Semantic	Subjects of questions
Syntactic	Wh-words

In QA, these four dimensions tend to overlap in question classification. I find it is more meaningful to distinguish between the following three types (also mentioned by Pomerantz)[4]:

1. Classifications based on the question only [Robinson and Rackstraw, 1972], for instance its wh-word (interrogative word): *who*, *which*, *what*, *when*, *where*, *why*, *how*.

2. Classifications based on answers only, as in the library reference world: the LIBGIS scheme [LIBGIS, 1981, Stalker and Murfin, 1996] classifies a question according to the task that the librarian must perform in order to formulate an appropriate answer, and taxonomies based on the sources in which answers may be located and their format [Richardson, 1995] (e.g. atlases, dictionaries).

3. Hybrid classifications of question and *expected* answer pairs. (Examples follow below with the corresponding characteristic answer retrieval strategies).

The latter classification can be linked to the distinction between (1) top down, (2) bottom up and (3) hybrid programming strategies. There is a direct correspondence between top down (analysis from *a priori* knowledge about data) and bottom up (analysis from the available data) strategies in information processing. However, hybrid classifications of question and *expected* answer pairs are most often top down approaches

[4]A sample of question taxonomies can be found in Appendix A for further reference.

in QA. Let us review these approaches and see how the main stream is still top down despite using "data driven" strategies.

2.2.2 Top Down Answer Retrieval Strategies

In QA, Lehnert [Lehnert, 1978] introduced the notion of focus (or asking point) and described 13 conceptual question categories in a taxonomy later extended by Arthur Graesser [Graesser et al., 1994]. Both approaches are based on a functional view of the answer, e.g. concept completion: *What did John eat?*, causal antecedent: *How did the glass break?*), which is somewhat tied to the task that initiated Lehnert's classification, story comprehension (see Appendix A).

In current free text QA, question classifications are designed to facilitate the discovery of text-based evidence and validate answers by type checking. For instance, [Hovy et al., 2001, E. H. Hovy, 2002] describe a hierarchical classification of QA types associated with answer patterns (see Appendix A). These patterns can be seen as template-based regular expressions, based on heuristics rather than conceptual categories. For instance, an archetypal question *When was <X> born?* calls for templates such as *<X> was born on <time-answer>*, where *<time-answer>* is a slot to be filled by a string matching a time reference. These classifications are widely used in QA. Such lexico-syntactic templates, defining constraints on the grammatical category of the answer (e.g. name, adjective) and the lexical/syntactic structure surrounding the answer, are also often mixed with a semantic type. [Prager et al., 2001] describe the use of WordNet hypernyms as a means to validate answers to definition questions. For example, *What is the currency used in China?* calls for an active template that looks up in WordNet whether the answer extraction found has hypernym *monetary unit*. This example was given by [de Chalendar et al., 2002], who also include other WordNet relationships, e.g. holonyms, and apply semantic validation to a wider set of question types. Elaborate systems also make use of first order logic and abduction to prove an answer [Harabagiu and Maiorano, 1999] (now a LCC system[5]). The latter system also includes an important amount of manually or semi-automatically pre-compiled knowledge, as well as a relaxation loop in the main algorithm [Moldovan et al., 2002]: If

[5] *http://www.languagecomputer.com*

the set of predetermined constraints characterizing the answer cannot be satisfied, the set is incrementally relaxed until an answer candidate fits. [Yang et al., 2003b] used a similar relaxation loop and report good results at TREC 12 (0.562 accuracy on factoid questions).

[Nyberg and Frederking, 2003] comment on such systems:

> Both the LCC and IBM systems represent a departure from the standard pipelined approach to QA architecture, and both work well for straightforward factoid questions. Nevertheless, both approaches incorporate a pre-determined set of processing steps or strategies, and have limited ability to reason about new types of questions not previously encountered. Practically useful question answering in non-factoid domains (e.g. intelligence analysis) requires more sophisticated question decomposition, reasoning, and answer synthesis. For these *hard* questions, QA architectures must define relationships among entities, gather information from multiple sources, and reason over the data to produce an effective answer.

Indeed, although data-driven approaches have helped handle the bottleneck of handwriting the required knowledge by learning surface text patterns and lexical inference rules from corpora [Ravichandran and Hovy, 2002, Lin and Pantel, 2001], the answer retrieval process (including relaxation) follows rules dictated by the *a priori* analysis of questions. In that regard, QA is closer to information extraction, based on filling predetermined templates or customized scenarios [Yangarber, 2000]. For instance, BioGrapher, a recent QA system specializing in biographical questions [Tsur et al., 2004], has built-in handcrafted knowledge about biographical relatedness (expression of birth dates, education, societal roles). Open domain QA also makes use of "universal" ontologies or databases whether manually or semi-automatically compiled (e.g. OpenCyc, WordNet, [Moldovan et al., 2000]). Hence, most approaches in QA are still completely top down, following taxonomies either of questions only, or questions and *expected* answer pairs.

Now, what is exactly an answer taxonomy and a bottom up approach in QA? Also, as mentioned by [Nyberg and Frederking, 2003], are some question types *harder* than others? I will argue that the difficulty of a question is tied in part to the material available for answering rather than a property of the question *per se*.

2.2.3 Towards a Bottom up Approach to QA

Pomerantz, from the librarian reference community, considers two taxonomies based on answers only. The LIBGIS scheme [LIBGIS, 1981, Stalker and Murfin, 1996] classifies the tasks to be performed by a librarian to find an answer: It specifies the type of transaction (and thus is somewhat linked to a type of question but in a broader sense) but mainly considers the amount of work necessary to formulate an answer. For instance, reference transactions involves passing on the knowledge concerning sources of information (recommendation, interpretation, use). Another such answer-based taxonomy focuses on the form location of answers (e.g. atlases with pictorial answers) [Richardson, 1995].

In QA over free text or multiple sources (whether they are structured or not), such classifications are not straightforward because the available information is not readily indexed with the QA task in mind. That is, indexing follows the predominant paradigm in IR, based on keywords (possibly their base forms) or phrases. Part of the research in QA involves finding appropriate knowledge indexing methods for answering questions, which is a different paradigm from standard IR (finding documents relevant to a set of keywords). Ahn [Ahn, 2005] proposes for instance a topic-oriented indexing for QA, in which, for instance, named entities (G.W. Bush), roles (president) serve as index keys.

Among other proposals are off-line strategies focusing on data preprocessing for QA. The START system at MIT [6] operates on the web and captures potentially interesting facts for further retrieval [Katz et al., 2001, Katz et al., 2002, Lin and Katz, 2003]. [Fleischman et al., 2003] describe generation of offline relational repositories of facts for QA, hence privileging strategies based on locating appropriate sources for answering. Although they focus on archetypal questions (*Who is <X>?*), their indexing is meant to be more general, expressing relationships between a concept and its instances (e.g. *flutist* $\rightarrow$ *James Galway*), which does fit well the relational paradigm of QA. There are also proposals and on-going research to propose a standard for adding meta-knowledge to the web using semi-structured encoding (Semantic Web, [Berners-Lee, 1998]) including users' annotation to facilitate disambiguation and im-

[6]http://www.ai.mit.edu/projects/infolab/

prove retrieval quality. These are steps towards the type of taxonomies available for librarians, based on knowledge indexing and annotations that are specific to the QA task.

On the other hand, efforts have also been put into studying the answering material retrieved from standard search engines. The original question is tokenized and represented as a set of keywords, which are fed into a search engine with retrieved documents processed "on the fly". The first step is thus top down, directed by question keywords. Nonetheless, in the following works I review, strategies also include, novel, bottom up aspects, in their study of the retrieved answering material.

At their first TREC QA participation (TREC 10), [Brill et al., 2001] ranked 9th out of 36 participants with a score of 0.35 (see TREC QA scores in Subsection 2.4.1 for a definition of this measure of accuracy). Although the best score achieved was 0.68 [Soubbotin and Soubbotin, 2001], 0.35 is actually a relatively good score, as few systems achieved more than 0.4. Their system (AskMSR) makes little use of linguistic knowledge. Instead, it queries the web to take advantage of *answer redundancy* and projects the results in TREC data to retrieve the final answers. In a later experiment on 500 TREC-9 questions answered with web data only, [Brill et al., 2002] report that the best score achieved was 0.507. AskMSR's pipeline is reproduced in Figure 2.1.

Each question is reformulated into a query set, e.g., *Where is the Louvre Museum located?* generates a query set containing: *"+the Louvre Museum +is located"*, *"+the Louvre Museum +is +in"*, *"+the Louvre Museum +is +near"* adapted to the syntax of the search engine used (Google in their experiment). This is typically a top down strategy, involving a priori knowledge about location questions. Surface patterns can be automatically acquired with machine learning techniques, hence reducing the human cost of hand classification. [Ravichandran and Hovy, 2002] propose such techniques using substring collection with suffix trees. Notice though they do not try to overcome the process of question typing by learning a machine-induced question type from the search material. The question is first typed (e.g. definition, birth year questions), and then associated with the answer patterns learned for that question type. Because the same pattern can apply to different question types, and also because patterns can also trigger false positives despite a good precision score, the authors propose to make use of an ontology or named entity tagging to ensure that the answer matches the semantic type expected from the question. Because of the required question typing, the technique is, in part, a controlled strategy, depending on external, *a priori*, knowledge.

Each query generated from reformulation is passed to Google (used as backend in Brill et al), and resulting snippets are mined for n-grams. Next, n-grams are filtered according to how well they fit the question types (using hand-written rules or manually compiled knowledge). Each n-gram is weighted according to its frequency and its "fitness" to the question type. Finally, overlapping n-grams are tiled and merged into longer answers (for instance *A B C* and *B C D* are tiled into *A B C D*). The top 5 n-grams are then presented as answers to the user.

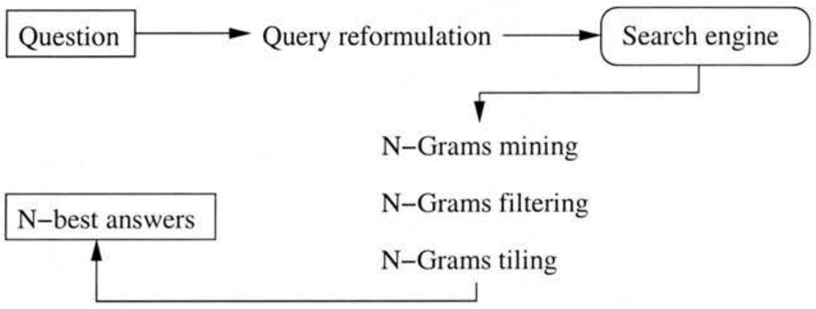


Figure 2.1: Using web redundancy for answering: AskMSR's pipeline

Although Brill et al. insist on using redundancy, it is interesting to notice that redundancy alone (weighted sum of n-gram occurrences) performs at 0.266, a drop of almost 50%, compared to the baseline using rule-based query rewriting and filtering (ranking score of 0.507 on the top 5 ranked answer candidates [Brill et al., 2002]). [Abney et al., 2000] also used frequency in the system's re-ranking scoring method, and report a mean score of 0.356 on TREC 8 data. [Clarke et al., 2001] show more clearly a correlation between re-ranking using redundancy and the size of the corpus they used for answering. They also report a score of 0.39 on TREC 9 data, and a drop of 12% when omitting redundancy as a feature. In all previous work mentioned, *answer redundancy* is understood as matching strings (eventually with some overlap) that occur several times in the search corpus. Synonyms, for instance, are not taken into account, nor relationships such as hypernyms or meronyms. To the question *Where is the*

Taj Mahal?, the candidates *Agra*, *Uttar Pradesh* and *India* each constitute different n-grams but they all refer to the *same* answer. The improvement brought by string-based redundancy could be further enhanced by a more abstract notion of *answer redundancy*, e.g. the set of strings that belong to the same answer. I now review previous research that is more focused on these relational aspects between answer candidates.

[Girju, 2001] describes a technique of *answer fusion* for definition and cause/effect questions. A tree of answer candidates is built incrementally from a root node identified in the question, the focus of the question. The root node is expanded with synonyms and a new query is formed according to the type of the question. For instance *What causes asthma?* is typed as a `cause` question. *Asthma* is extracted as the focus of the question and expanded with WordNet synonyms (of the first sense of the word only): *asthma, asthma attack, bronchial asthma.* Answer candidates are extracted using the corresponding `cause` patterns: `X is caused by Y`, `Y and other causes of X`, etc. For each new answer, a node is created and the same process applies until no new nodes are found. Figure 2.2 shows an example of such tree generated from this technique.

Girju starts with highly precise input (including hand-written rules) and expands it as necessary. This is again a top down approach. The novel aspect is to further mine the dataset from the first pass of candidates found. She reports errors due to word sense ambiguity.

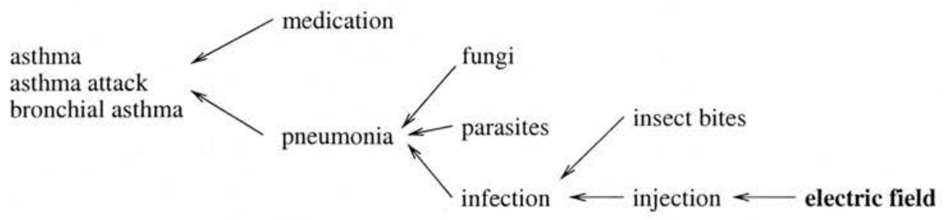


Figure 2.2: Answer fusion: Partial reproduction of the ontology generated for *What causes asthma?*

As highlighted in Figure 2.2, the system found a causal relationship between *electric field* and *injection*. This comes from the following sentence: ... *negative charges dominate in the layer due to the injection* ***caused by*** *the electric field.* As mentioned by

Girju, these errors can propagate in the ontology and add noise. But most often, with *domain-specific questions, most of the concepts involved in the ontology are monosemous* [Girju, 2001]. Also, considering open domain QA and the input retrieved from a given set of keywords, even though one might be ambiguous (e.g. *how long is <X>* can stand for distance versus duration), it is not always necessary to disambiguate beforehand as the data retrieved can be unambiguous: a web query such as *How long is an elephant pregnant?* to Google essentially retrieves duration measures (among the top 10 documents, distance related information is non-existent, and the size of an elephant is mentioned once, against 5 occurrences indicating the length of the gestation period). One advantage of considering a first pass of answer candidates before attempting a full understanding of the question would be to help disambiguation and prune the tree of possible *a priori* interpretations. Whereas a fully top down approach would require elaborate rules such as *pregnancy* implies a duration and not a distance, a first shallow glance at the data itself simply does not support the distance interpretation. Or perhaps, interpretations could be *pulled* from the answer candidates rather than *pushed* from question analysis, to make a comparison with the *push* versus *pull* models in marketing for managing resources and activities. In a *push* model, suppliers initiate events (e.g. producing). In the *pull* model, it is the consumers who activate events first. The problem that might arise from the *push* model is the cost of anticipating all possible needs (as in-depth *a priori* question analysis). *Push* models also treat people as passive consumers. The *pull* models are more flexible and are made to handle a greater range of uncertainty. Customers are given a greater control over the events.

However, few studies consider a more general classification of answers, that could be *pulled* on demand by users of a QA system. [Buchholz and Daelemans, 2001] introduce their work on complex answers as follows:

> The underlying assumption in much research on question answering (QA) is that in most cases there is one (correct) answer to a question, and that this answer has to be found. We think that whether or not a question has exactly *one* answer depends on how one defines an answer. In our view, many answers are *complex* answers, which contain two or more *simple* answers.

A *complex* answer is not the corollary of a complex question. As I showed in Table 2.3 about multiple answers, even reputed "simple" quiz-like questions, such as location questions, accept multiple answer strings. Such multiple answer strings sometimes refer to the same entity or provide different levels of information about it. Other answers, e.g. *What were Christopher Columbus' three ships? The Nina, the Pinta and the Santa Maria*, are composed of simpler elements which, on their own, are not sufficient to play the role of an answer. Unless the string is directly found in the search corpus, the appropriate answer may have to be reconstructed. Answers are complex, as in a conceptual whole made of related parts.

[Buchholz and Daelemans, 2001], as well as [Webber et al., 2002], have proposed different formalizations of answer relations to handle such complexity. Table 2.6 presents a comparison of the sets of relations envisaged by both approaches.

Table 2.6: Relationships between answers

[Webber et al., 2002]	[Buchholz and Daelemans, 2001]
answers determined to be equivalent (mutually entailing)	*different measures, different designations, time dependency*
answers that differ in specificity (one-way entailing)	*granularity*
answers that are mutually consistent but not entailing can be replaced by their conjunction (aggregation)	*collective answers*
answers that are inconsistent, or *alternative answers*	*many answers, ambiguity in the question, different beliefs*

Considering that an answer is a complex structure raises two points that are at the core of this thesis:

1. Answer strings as found in the search corpus are not necessarily competitors as considered until recently with work on answer redundancy and cause/effect relationships between answers. I will focus on considering them as potential

allies, and introduce the notion of *answer clusters* to start reasoning on answer structures.

2. Consequently, if an answer consists of several related extractions, we have to develop evaluation techniques that recognize that an answer is the product of the fusion of distinct pieces of information possibly coming from different locations (e.g. passages, documents, websites).

Considering that, in QA over free text, *variability* is a property of answers, and hence that we may need to compare and organize answer candidates in order to detect such variability, I will now review work in information fusion and general text clustering, which are known techniques for merging and structuring data.

2.3 Information Fusion and Clustering

[Girju, 2001] demonstrated the benefit of *answer fusion*. She improved the accuracy of her system by using relationships between answer candidates and organizing them into a relational tree (see Subsection 2.2.3).

Information fusion is a term that refers to the merging of information from different sources. It is a term used notably in robotics: A robot captures world information from different sensors (e.g. visual, spatial). The different data streams are then combined and analyzed all together for the robot to respond adequately under certain conditions. In natural language processing, the term has been used as a technique for merging information, usually coming from different sources, and performing reasoning over the fused data. In IR, the term most frequently encountered for merging data is *clustering*. *Information fusion* describes the process of merging and organizing data. *Clustering* might be one result of the fusion process, i.e. the grouping into separate partitions or clusters of pieces of information that have been related to each other in some way by fusion. However, in IR and related fields, the term is also used to describe algorithms that relate documents to each other.

In this section, I first review work in automated summarization from multiple documents, which shares several similarities with QA over a large corpus, notably the

intersection of relevant information. On the other hand, I also review recent work on document fusion, aiming at the **unification** of several documents into one coherent, non-redundant, fused document, which is also close to the QA paradigm of eventually grouping answers that have been found in different places. The last subsection is devoted to general text clustering techniques, how they have proved useful for answering definition questions, and why there is still room for adapting them to other types of question.

2.3.1 Fusion in Automated Summarization

Multi-document summarization shares with open domain QA problems of data redundancy and heterogeneity in data. While traditional summarization techniques, such as sentence extraction or text compression, may provide a fair baseline, they are not satisfactory in terms of text cohesion. For instance, anaphora are often broken when sentences are extracted. [Paice, 1990] focuses on solutions to handle anaphoric references dismantled by such approaches in automated abstract generation from a single document. However, anaphora are not the only source of confusion in extraction-based summaries. When dealing with several articles written at different times, from different perspectives, the lack of cohesion is even more striking.

[McKeown and Radev, 1995, Radev and McKeown, 1998] first addressed summarization of a *series* of news stories on the same event, and focused *on techniques to summarize how the perception of an event changes over time, using multiple points of view over the same event or series of event.* The input to McKeown et al.'s system, SUMMONS, is a set of templates produced by MUC-4 systems (Message Understanding Conference [MUC, 1992]) from full text articles relating to the terrorist domain. Templates consist of specialized fields, such as perpetrator, number of victims and type of incident. MUC systems process full text and extract the pieces of information relevant to the template's fields. To these predefined templates, McKeown et al. added four slots meant to pin-point the exact origin of each report: the primary source (a direct witness of the event), the secondary source (journalist or press agency) and the times when both sources made their report. SUMMONS is based on a language generation system, divided into two main components:

1. A content planner selects information to be included in the summary using a set of planning operators.

2. A linguistic component determines the lexical items and the surface syntactic form to realize the planned information into appropriate English sentences.

I will focus on the content planner, which, to me, is the component performing information fusion: It identifies what information to include from the set of templates and how to group it together. McKeown et al. do not use the term *fusion*, but refer to planning steps as processes of *merging* and *combining* information.

The content planner takes as input a set of templates sorted in chronological order with the same initial "importance" weight. The algorithm incrementally applies a set of operators on this input. These operators were identified from the study of a corpus of summaries. Each operator is a scanner seeking patterns expressing relevant relationships between templates. These patterns, or triggers, were collected from a training corpus (2MB dataset from the Wall Street Journal, Reuters, and Associated Press, plus a home-made assemblage of a few sentences describing a single event changing over time). The cues are relevant to summarization as well as the terrorist domain. Each operator generates a new combination of merged templates, in which similarities and differences are marked. The templates that were used to generate this combination are either noted as obsolete or as less "important", while the newly created template is given a more "important" weight. An operator may also change the initial ordering of information. The incremental application of planning operators follows a heuristic approach. McKeown et al. distinguish eight operators, capturing the following relationships between information reports:

1. Time-related change of perspective: an initial reported fact appears to be wrong.

2. Contradictions or disagreements between two sources.

3. Additions in subsequent reports.

4. Refinement, e.g. the terrorist names are reported, whereas earlier reports only mentioned the organization's name.

5. Agreement between sources are marked to heighten the reader's confidence in the veracity of the facts commonly reported.
6. The information overlap (superset) operator combines incomplete information.
7. Trend (not implemented) reflects the recurrence of similar patterns over time (e.g. repeated bombing).
8. The absence of information (present in another source) is notified.

Note that the underlying relations expressed by these operators are close to the list of possible answer relationships enumerated by [Buchholz and Daelemans, 2001] and [Webber et al., 2002]: equivalence, aggregation, alternative and granularity (inclusion).

The final summary is produced with the generation system FUF/SURGE (described in [Elhadad, 1993]), and operator roles are transformed into summary cues previously collected from newswire corpora (e.g. the operator marking the absence of information is translated into *X (the source) didn't confirm*).

Following this research, there have been several proposals to systematize the approach into domain independent systems. Mani and Bloedom proposed a graph-based technique to identify similarities and differences among documents in order to perform summarization [Mani and Bloedorn, 1997, Mani and Bloedorn, 1999]. Fusion consists in building a graph, in which each node represents a word, and edges are relationships expressing either adjacency or similarity (by string matching, synonymy and hypernymy) or co-reference. Using spreading activation, i.e. considering that nodes that are relevant to salient ones are also significant [Collins and Loftus, 1975, Chen et al., 1994], a final output is constructed by combining (1) sentences that contain the most common nodes, and (2) sentences that are different (having a lot of unique words).

However, as argued by [Barzilay et al., 1999], the use of sentence extraction is inherently limited by the fact that *any representative sentence usually includes embedded phrases containing information that is not common to other similar sentences. Therefore, we need to intersect the theme sentences to identify the common phrases and then generate a new sentence.* Barzilay et al. propose such an algorithm for theme intersection leading to natural language generation. Sentences are parsed into dependency trees and compared in pairs. Comparison consists of a tree matching algorithm starting from the root node (the verb). Identical or paraphrasing nodes (based on lexico-syntactic patterns and WordNet synonyms) are merged into an output tree. Once a full tree has been found (a phrase, i.e. a verb with at least two constituents), it is added to an intersection set. The time-stamp for each phrase is preserved, eventually replaced with a non relative reference (e.g. *today* with *10/21*, and the set of phrases are sorted chronologically. They are then fed into a language generation system (FUF/SURGE).

Further advances can be found in [Barzilay, 2003, Barzilay and McKeown, 2005] and [Radev et al., 2000], the latter proposing a technique for summarization of multiple documents based on clustering of centroids.

QA on a large collection of free text and summarization of multiple documents share similar issues. A shared objective is the desired ability to generate an optimal intersection. In summarization, it is obviously the core of the task. In QA, one of the aims is to provide concise answers so that the user does not have to browse through a list of documents. Summarization is already, to some extent, available in most current web search engines, which usually provide a snippet to characterize each document in the list they return to a user query.

Redundancy and similarity are features of the corpus used to construct a summary, or, in QA, an answer. Now, while summarization takes into account potential alternatives, those are currently ignored in QA. Perhaps this is because summarization is more obviously dealing with opinions or subjective reports, and the task does not consider a piece of information to be correct or not. It considers whether it is relevant to report it or not, and it tries to acknowledge differences between sources.

Also, summarization is currently moving from extraction techniques to natural language generation. Ultimately, QA will also move towards generated answers.

In the next section, I review another approach to fusion, based on a **union** of information pieces, rather than an **intersection**.

2.3.2 Document Fusion

[Monz, 2001] also addresses the notions of redundant (equivalence), additional (aggregation), conflicting (alternative) or more precise (granularity, inclusion) information in news wires. He describes the automated fusion of multiple documents into one single comprehensive document eliminating redundancy. Whereas, in summarization, the goal is to pin-point essential facts (e.g. theme intersection [Barzilay et al., 1999]), Monz proposes a union of the information contained in different documents. As Monz mentions himself, this is a rough opposition as summarization also requires a comprehensive approach (i.e. union) to pin-point differences (i.e. intersection using fusion). There is a similar trade-off in QA. As in summarization, the requirements are to avoid overloading the user with too much information and to pin-point relevant information. At the same time, in order to identify what information is relevant, it is necessary to have a comprehensive approach to the answering material (i.e. union of the information) to understand the relationships between possible answers in both the context of the question and the context they have been found in.

For news wires, Monz addresses the following points which are also relevant for fusion in QA:

1. What is the best level of granularity for fusion (words, phrases, or larger)?
2. What relationship will determine information redundancy?
3. What should such a fused document look like?

The first item concerns the selection of information to be fused, which Monz presents as a segmentation problem. He chooses to fuse paragraphs to reduce context-related problems on the sentence level and make the comparison easier. Context-handling also relates to Monz' third item. The choice of a level of segmentation impacts on the result of fusion and its rendering. In QA, context-handling can vary: One characteristic of QA is that an answer might be found incidentally in a document whose topic is actually not related to the question at all. [Lin and Katz, 2005] give an example of this phenomenon regarding the answer to *How many floors are in the Empire State Building?*, which could be found in a document about the Great Depression. Segmentation for fusion in QA may have to be finer grain than paragraph level because the immediate context is not always relevant and may need to filtered out. Although a paragraph is easy to read as a unit, in QA, it may introduce totally unrelated context that will make the answer harder to understand. For instance, *What is the Ohio state bird?*, from a readability point of view, is loosely answered with *My travel buddy is a red cardinal beanie baby that represents the Ohio state bird.* I will address the level of segmentation for fusion in QA in the next chapter.

The second point Monz addresses is the type of comparison to be performed between paragraphs. He proposes two comparisons: *informativity* and *dissimilarity*.

While logically proving an entailment remains a hard task of natural language processing, in some cases, it can be effective to simply compute word overlap. (In the first PASCAL Recognizing Textual Entailment Challenge [Dagan et al., 2005], it also appeared that *system complexity and sophistication of inference did not correlate fully with performance, where some of the best results were obtained by rather naive lexically-based systems.*) Monz proposes to compute informativity as follows: The (non symmetric) entailment score (*es*) between two segments s_i and s_j is computed as the sum of the weights of terms at the intersection of the two segments, normalized by the total sum of weights of s_j:

$$(1) \quad es(s_i, s_j) = \frac{\sum\limits_{t_k \in s_i \cap s_j} idf_k}{\sum\limits_{t_k \in s_j} idf_k}$$

The weight of a term t_i is its inverse document frequency (idf), defined in (2) where N is the number of all segments in the set of documents, and n_i the number of segments in which the term t_i occurs.

$$(2) \quad idf_i = log\left(\frac{N}{n_i}\right)$$

Terms with a high idf score (low n_i) are expected to be more discriminating than terms occurring in many segments (terms with a large n_i, e.g. stopwords).

The algorithm goes as follows in the described implementation:

1. Segment documents on a given topic into paragraphs.
2. For each pair of segments, compute both entailment scores, $es(s_i, s_j)$ and $es(s_j, s_i)$.
3. Select the longest document as the base document. (It is expected that a long document has a good coverage of the topic.)

4. Replace each segment in the base document that is entailed by a segment of another document. If there are several entailing candidates, the segment that maximizes the entailment score is selected. The entailment score has to be superior to some threshold.

5. For any non-base segment that is dissimilar to every base segment, i.e. the cosine similarity of a pair is below some threshold, insert the non-base after the most similar base segment.

The union of information is performed by (1) replacing each entailed base segment by a more informative non-base segment, and (2) appropriately inserting non-base segments that contain different information. This is close to an approach based on identifying similarities and differences, replaced here by levels of *informativity* and *dissimilarity*.

Fusion is evaluated on two grounds: (1) the performance of entailment recognition, and (2) the quality of fused documents. Evaluation of entailment is based on human judges, who annotated subsumption pairs of segments on an entailment scale (see also [Monz and de Rijke, 2001]). Monz proposes to make use of two IR criteria to assess the quality of fusion:

1. *False Alarm* measures the amount of redundant segments that add no information and should have been excluded (false positives).

2. *Miss* measures the amount of missing informative segments that should have been included (false negatives).

Another measure, the *fusion impact factor* (*fif*), describes the distribution of sources in fusion. For example, if the fused document contains segments from only one document, $fif = 0$, and if there is an harmonious distribution, $fif = 1$. The latter measure is a descriptive rather than a qualitative assessment since the quality of the sources themselves might be subject to discussion, and an unbalanced selection could still prove to be high quality.

Both *Miss* and *False Alarm* are difficult to assess intrinsically because segment exclusion and inclusion depend on two separate thresholds (entailment and similarity):

Measuring the effectiveness of different pairs of thresholded values requires extensive annotation. Instead, Monz describes the results of an extrinsic *ad hoc* retrieval task to assess the precision of the fused document based on the original document recall.

The query consists of the topic description, with, on the one hand, the concatenation of all documents as a baseline (i.e. without information missing), and on the other the fused document. The search corpus (249,996 documents) also contains the original topic documents (69 documents). The measure of recall at rank (i.e. the number of original documents retrieved at rank 1, 2 ... *N* for each query – the baseline and the fused document) characterizes the retrieval performance. The baseline provides an optimal score for the baseline, since the baseline query already includes all the original documents, and the fused document returns a subset. This allows measurement of the precision of the fusion document compared to the baseline. By repeating the experiment with various instantiations of the entailment threshold, it is possible to obtain an optimal value for fusion thresholds. The task assesses the different levels of precision in fusion, the best score being a decrease of 11.5% in precision against the baseline when using an entailment threshold of 0.2. The retrieval task approximates the *Miss* measure. However, as mentioned by Monz, it is not clear how to mimic the *False Alarm* without involving heavy human annotation.

Similar problems will occur when using fusion for QA. Evaluating answer correctness is already extremely costly in terms of human annotation. Basing an intrinsic evaluation on relationships inferred from fusion seems both expensive and undesirable.

In the next section, I will cover techniques and methodologies used to evaluate fusion, or more exactly clustering in IR.

2.3.3 Clustering Techniques

Conventional IR can return a long list of ranked documents for the user to browse. But it seems that human users actually do not make use of such a long list. As reported by [Silverstein et al., 1999] from the analysis of a large query log (280GB) of the AltaVista search engine, user sessions during web search are short: 85.2% of the users look at one result screen only. Part of the IR community is currently focused on providing more efficient techniques for browsing such results.

Two aspects are being considered: (1) the discovery and organization of groups of related documents rather than a flat ranked list, (2) the exploration of different modes of visualization using clusters. I will discuss the latter aspect with respect to rendering results from QA in the next chapter. To illustrate the concept in IR, NIRVE [Cugini et al., 1996, Cugini et al., 2000] [7] is an elaborate visualization tool that clusters documents into conceptual groups, and renders these groups in two or three dimensions with appropriate browsing techniques.

In IR as in document fusion, similar questions are asked:

1. What is the best level of granularity (keywords, phrases, sentences, paragraphs or full documents)?
2. On what relationships will clusters be based?
3. How do we represent and visualize clusters produced from fusion?

While IR is usually concerned with organizing data on the document level, recent literature shows research in finer grain clustering.

Scatter/Gather was the first clustering algorithm introduced as a method to facilitate browsing [Cutting et al., 1992, Hearst and Pederson, 1996]. [Zamir et al., 1997] followed on the same paradigm, and described a hierarchical agglomerative clustering method based on phrase intersection between documents. They propose an original algorithm taking advantage of the efficiency of a suffix tree to automatically cluster documents. The suffix tree is fed with the lists of words representing the documents to be clustered. Each node of the tree represents exactly a cluster of documents, labelled with a phrase they all share. The similarity function used for clustering is basically defined by phrase overlap. This work later evolved into the clustering engine GROUPER [Zamir and Etzioni, 1999], which dynamically groups the search results into clusters labeled by phrases extracted from the snippets. The advantage of Suffix Tree Clustering (STC) is that it allows overlapping between clusters, i.e. a document can belong to different clusters at the same time, expressing that the document has multiple topics.

[7]NIST Information Retrieval Visualization Engine, http://zing.ncsl.nist.gov/˜cugini/uicd/nirve-home.html

STC also proved to be faster and more precise than previous algorithms known as fast methods, such as k-means [Rocchio, 1966].

[Pantel and Ravichandran, 2004] also address clustering on the phrase level, using the Clustering By Committee (CBC) algorithm previously described for document clustering in [Pantel and Lin, 2002].

CBC initially discovers a set of tight document clusters (with a high intra-group similarity), called *committees*. The algorithm then proceeds by assigning elements to their most similar committee. CBC has been proven to outperform several well-known clustering algorithm (e.g. K-means) in terms of cluster quality because it uses a set of members rather than one member to perform comparison. *Using a single representative from a cluster may be problematic because each individual has its own idiosyncrasies that may not be shared by other members of the cluster* [Pantel and Lin, 2002]. With a committee to decide whether an element is member of a cluster or not, the features tend to be more typical of a class rather than an individual.

[Pantel and Ravichandran, 2004] apply the same technique to phrases, using grammatical relationships extracted from Minipar as features. The main application of CBC for phrase clustering is semantic class labeling. For instance, the committee for the class *goaltender* would be defined by the members *Curtis Joseph, John Vanbiesbrouck, Mike Richter, Tommy Salo*, and the class signature defined by the typical lexico-syntactic features found around the committee member X, e.g. *X's glove, goalkeeper X*. If a phrase X fits such a signature, it is likely to refer to a goaltender. Pantel et al. show that the relation between a word and the committee of its class is often a hypernym relation[8].

In open domain QA, it has been shown that looking up hypernyms as demonstrated in [Prager et al., 2001], or instance-concept relationships from repositories (strategy described in [Fleischman et al., 2003]) was helpful to answer definition questions such as *What is acupuncture?* (hyponym of *treatment*), *Who is Aaron Copland?* (hyponym of *composer*). Pantel et al. tested CBC on 50 definition questions from TREC 2003, and compared their results against WordNet, and the strategy proposed by Fleischman et al. The results of the evaluation are reproduced in Figure 2.7.

[8]A demonstration of CBC is available at *http://www.isi.edu/ pantel/Content/Demos/LexSem/cbc.htm* on the TREC corpus.

Table 2.7: Clustering for definition questions - Answer correctness in the Top-1 and top-5 returned answers

System	Top-1		Top-5	
	Strict	Lenient	Strict	Lenient
WordNet	38%	38%	38%	38%
Fleischman	36%	40%	42%	44%
CBC	36%	44%	60%	62%

The strict measure refers to answers that are supported by a TREC document. The lenient measure accepts answers that are correct but not supported by a document. Although all systems perform equivalently on first ranked answers, the recall is much higher using CBC with 60% of the correct answers found among the top 5 candidates.

QA would benefit from clustering as well as more advanced answer visualization. A flat list of ranked answer extractions may require some mental gymnastics, as mentioned by [Hirschman and Gaizauskas, 2001]: *For first time users, it may be important to explain the limitations of the system, so that the user can understand how to interpret the answers returned.* Indeed, visualizing the following list: Agra, India, Atlantic City, Washington, as answers to the question *Where is the Taj Mahal?* is confusing. Although the user may connect Agra to India from his own world knowledge, it is not clear why US cities appear in the list. Are they mistakes? Or alternative answers to an unexpectedly ambiguous question? Clustering would help connect related answers and explain the results, especially if the answer clusters contain contextual elements to explain the answer. This is where clustering for QA would need to be more specific than general text clustering.

In IR, clustering is usually based on a similarity function, which tends to cluster documents by the most salient topics. With CBC, besides similarity clustering, another feature is the typical relationship of hypernymy between a committee and its semantic class label. Both approaches work well for definition questions, which tend to expect ontologically related answers, or for which a list of salient topics can be satisfying. Definition questions require indeed little linguistic processing to identify the type of answer expected. For example, the question *What is epilepsy?* can be translated into the simple keyword *epilepsy*, and general text clustering would perform well on such a query.

However, contrary to general text clustering, QA clustering should be a function of the question. This is particularly obvious for factoid questions. For instance, the four Taj Mahal locations are all somewhat *touristic resorts*. But it would be confusing to cluster the four answers because they are related to tourism. A better presentation would be to cluster Agra and India together because they refer to the same Taj Mahal, and then distinguish two singletons, Atlantic City and Washington. Also, one would like to explain such partitioning, and mention that, in the first case, the Taj Mahal is a world wonder, in Atlantic City, it is a casino, in Washington, a restaurant. This is a slightly different paradigm from general text clustering, that organizes documents retrieved from keyword query.

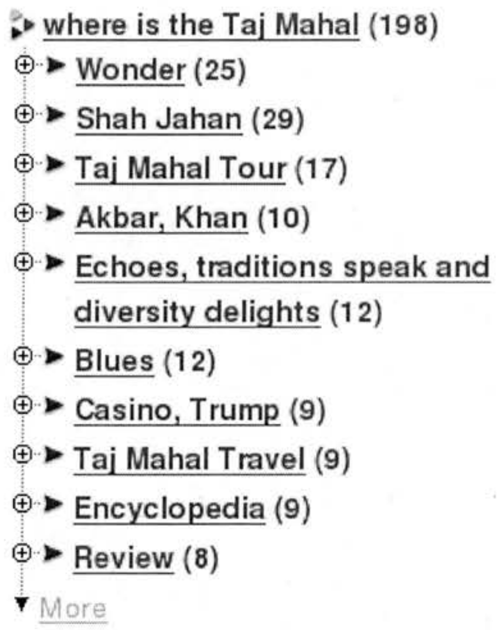

where is the Taj Mahal (198)
Wonder (25)
Shah Jahan (29)
Taj Mahal Tour (17)
Akbar, Khan (10)
Echoes, traditions speak and diversity delights (12)
Blues (12)
Casino, Trump (9)
Taj Mahal Travel (9)
Encyclopedia (9)
Review (8)
More

Figure 2.3: Vivisimo clustered results to *Where is the Taj Mahal?*

For instance, the web clustering engine, Vivisimo[9], displays both a ranked list of

[9]http://vivisimo.com

documents and a small tree of clustered documents on the left menu. Each node of the tree contains a list of leaves (documents) or children nodes. Figure 2.3 shows the clusters returned for the query *Where is the Taj Mahal?*

Clusters do not answer the question, although they do provide some disambiguation of the term, e.g. *Wonder* versus *Casino, Trump* (Atlantic City's Taj Mahal), and *Blues* for Taj Mahal, the American singer and guitarist. If anything, the clusters actually answer the question *What is the Taj Mahal?*.

Information fusion and clustering for QA require more specific relationships than similarity, which is used in general text clustering.

Having considered the use of fusion for the purpose of different tasks, such as automated summarization, I discuss in the next section the current evaluation methods used in open domain QA on a large corpus (TREC QA), as well as the current performance of automated systems. I also present a smaller resource for multiple answers in the context of Reading Comprehension, focused on shorter texts, but also more exhaustive answers. I discuss evaluation issues when assessing answers coming from fusion, especially when such answers are presented in a more complex manner, for instance clusters, instead of a flat list of single answers. I will show that these issues share again a similar paradigm with automated summarization, especially from multiple documents.

2.4 Evaluation of Question Answering

Evaluating answers is known to be a difficult task that requires human expertise and therefore to be slow, expensive and difficult to reuse [Voorhees, 2003]. Recent work [Magnini et al., 2002b, Burke et al., 1997, Leidner and Callison-Burch, 2003] has focused on automatic evaluation and QA corpus generation. On the other hand, time has been spent on the manual evaluation designed and organized annually by NIST (TREC) for open domain QA systems. Each year a significant amount of data is published on the NIST website and gives a fair overview of what kind of answers one can get from current QA systems. On a minor but not smaller scale, Reading Comprehension Texts [Light et al., 2001] have been made available with gold standard answers (CBC4Kids).

I review the history of the TREC QA evaluation, and detail the main type of material available for research on multiple answers for English. The last subsection summarizes the positive as well as problematic aspects of current QA evaluation methods.

2.4.1 TREC QA

The Text REtrieval Conference (TREC), co-sponsored by the National Institute of Standards and Technology (NIST), and the U.S. Department of Defense, was started in 1992 as part of the TIPSTER Text program. Its purpose is to support research within the IR community by providing the infrastructure necessary for large-scale evaluations of text retrieval methodologies. After [Harman, 1992], the first TREC (1992) was specifically meant to improve two research areas in IR that were seen as unsatisfactory:

1. The absence of *a concerted effort by groups to work with the same data, use the same evaluation techniques, and generally compare results across systems.*

2. *The lack of a realistically-sized test collection.*

The first QA track was organized as part of TREC 8 with the same desire of setting up common ground for evaluation in open domain QA on a large corpus [Voorhees and Tice, 1999]. Two hundred questions were gathered from a FAQ dataset as well as suggested questions from the participants themselves. In the subsequent tracks, questions were more realistically extracted from search engine logs (Encarta and Excite), and grammatically corrected if necessary. The size of the set finally stabilized around 500 questions per track, balancing between the need for a realistic size and an optimization of human assessment feasibility and agreement rate among human raters who judge systems' answers.

Questions are mostly factoid, trivia questions (*What is the length of the coastline of the state of Alaska?*) or definition/biography questions (*What is leukemia? Who was Galileo?*). The TREC QA corpus used for answering is a 3.6 GB subset of the AQUAINT collection, and currently comprises articles from the Associated Press, the New York Times and Xinhua English News from 1998 to 2000.

The output of the systems being assessed is evaluated by human raters who check (1) the answer string and (2) the document that supports the answer. Since answers are extractions, they must be supported by an existing document from the corpus. Notice that, since assessing the whole collection for each question would be unpractical, only the set of supporting documents pooled by participants are evaluated by human judges. The evaluation is binary, i.e. the answer string is either correct and supported, or incorrect or unsupported (strict evaluation). For gray areas, there is a middle-way score, for answers that are not completely satisfying but yet not wrong (lenient evaluation scoring unsupported correct answers).

From these basics of the TREC QA evaluation, there have been several developments, with variations on three aspects: the scoring measures, the constraints on the answer strings provided (e.g. size), and the organization of the question set (e.g. with supervised labels such as definition versus factoid, and/or an organization of questions by topic).

Until TREC 11, the main measure to compare performance on the set of questions was the Mean Reciprocal Rank (MRR), which evaluates the overall ranking method:

$$\mathrm{MRR} = \textstyle\sum_{q=1}^{N} \frac{1}{r_q} / N$$

where r is the rank of the first correct answer for each question q and N the total number of questions.

For both TREC 8 and 9, two runs were distinguished, one with a 250-byte limit on the length of the response (the size of a sentence) and another with a 50-byte limit (the size of a phrase). The MRR was computed over a ranked list of five candidates per question. I reproduce the top 5 TREC 9 system scores for both types of runs in Table 2.8 below.

Table 2.8: Top 5 TREC QA 9 scores (strict evaluation)

Participant	**MRR 250-byte limit**
Southern Methodist U.	0.76
IBM (Ittycheriah)	0.46
Queens College, CUNY	0.46
U. of Waterloo	0.46
IBM (Prager)	0.42
Participant	**MRR 50-byte limit**
Southern Methodist U.	0.58
ISI, U. of So. California	0.32
U. of Waterloo	0.32
IBM (Prager)	0.32
IBM (Ittycheriah)	0.29

At TREC QA 10, the 50-byte limit was kept as the rule, and questions were not guaranteed to have an answer in the corpus (possibility of a NIL answer). I reproduce the top 5 system scores in Table 2.9 for comparison.

Table 2.9: Top 5 TREC QA 10 scores - Main task

Participant	**Strict MRR**	**Lenient MRR**
Insight	0.68	0.69
LCC	0.57	0.59
Oracle	0.48	0.49
ISI, U. of So. California	0.43	0.45
U. of Waterloo	0.43	0.46

For TREC 11, NIST proposed as a new rule that QA systems should submit *exact answers*:

> What constitutes an "exact answer"? As with correctness, exactness is essentially a personal opinion. NIST provided guidelines to the assessors so that questions would be judged similarly, but in the end whether or not an answer was exact was up to the assessor. [Voorhees, 2002]

For instance, to the question *What is the longest river in the United States?*, the following answers: *Mississippi*, *mississippi*, *the Mississippi River* are considered correct and exact answers while none of the following are considered exact: [*At 2,348 miles the Mississippi River is the longest river in the US.*], [*2,348 miles; Mississippi*].

It seems that the main point has been to make sense of the idea of providing short answers, i.e. answers that contain the correct information, no more no less. The problem is that informativity does not always correlate with size. Informativity is also a matter of the judges' *personal opinion* [Voorhees, 2002]. The difficulty does not come from the criterion itself but from the fact that the notion of *exact answer* is hard to ground on strings for which the only attributes we have are basically length and character matching. Interestingly this idea of an *exact answer* is rather similar to the notion of a *direct answer* defined by [Belnap and Steel, 1976]: *A direct answer is a piece of language that completely, but just completely, answers the question.* (Notice that Belnap uses the term language in a formal sense so that this language could be a natural language or a pictorial language as well). Oddly, in Belnap's sense, *Mississippi* is not a direct answer but a coded answer, i.e. an abbreviation for the following direct answer: *the Mississippi River is the longest river in the US.*

Other characteristics of the 11th track were the absence of definition questions in the main task and a separate track for list questions. In the main task on factoid questions, systems were required to provide one and only one answer per question. Questions with NIL answers were preserved. Finally, systems were required to rank questions by confidence, i.e. the more certain about the correctness of answer the higher ranked the question. MRR was replaced with the percentage of questions correctly answered, the top 5 scores are reproduced in Table 2.10

Table 2.10: Top 5 TREC QA 11 scores - Main task

Participant	**% of correctly answered questions**
LCC	83.0
National U. of Singapore-Hui	58.0
Insight	54.2
ITC-IRST	38.4
IBM	35.8

In TREC 12 input, list and definition questions returned to the main task and were labeled as such. Factoid questions still expected only one exact answer, while list and definition questions could be answered with multiple (and longer) extractions.

From TREC 13 on (2004), the corpus of questions was divided into topics, and and questions were labeled with three possible types: *factoid*, expecting only one exact answer, *list* and *other* questions, expecting an unlimited list of answers for which exactness is more lenient (e.g. an answer could be a full sentence). Factoid questions may have no answer (NIL). TREC distinguishes vital and optional answers to *other* questions. Systems are penalized if they do not find all vital answers.

TREC systems are now evaluated with three metrics, one for each question type. The *factoid* score is the number of questions correctly automatically answered. Precision and recall are computed for NIL answers, with precision as the number of correct NIL over the total number of NIL produced by the system, and recall as the number of correct NIL identified. For *list* and *other* questions, recall and precision are computed for each question using the number of known correct answers. Redundancy among answers is penalized. For *list* questions, the F-measure has recall (R) and precision (P) weighted equally ($\frac{2*P*R}{P+R}$). For *other* questions, recall is weighted three times as much as precision. The final score of a system is the weighted average of the three scores (accuracy of factoids and average F-measures). The final weighting represents the distribution of question types in the corpus:

$$\text{Score} = 0.5(factoid) + 0.25(list) + 0.25(other)$$

Table 2.11 shows TREC 13 median and best scores for each question type, over 63 runs (Note that a run corresponds to the results submitted by a system, several runs can be submitted per system). Notice the gap between the best systems and the median system is fairly large.

Table 2.11: TREC QA 13 scores (63 runs)

	Best	**Median**
Factoid	0.770	0.170
List	0.622	0.094
Other	0.460	0.184

The main evolution in TREC QA is the modularization of question processing as shown by the scoring metrics. Since TREC 11, wherein systems had to provide only one answer per factoid question, the core component of the evaluation is basically information extraction. This is even more obvious with the organization of questions by topic. For instance, the topic *Black Panthers* had the following factoid questions: *Who founded the Black Panthers organization? When was it founded? Where was it founded?* There are now typical TREC QA questions such as location, date, age, which information extraction systems are good at. (Some systems even have a specific question type: *How did X die?* which is a frequent TREC QA question.)

TREC QA answers are a useful resource to study answers although the set of acceptable answers has been slightly narrowed for factoids. For multiple answer processing, answers to list and other questions are available. The latter material is nonetheless difficult to use as distributed because judgments are done on a per system basis, e.g. each system is penalized individually depending on the specific redundancy of its results. In other words, redundancy is evaluated per system, not for the whole answer pool, which makes the evaluation material difficult to reuse directly.

2.4.2 CBC4Kids

The CBC4Kids corpus was developed at MITRE , based on a collection of newspaper stories for teenagers written for the CBC's website[10]. To each article (500 words each on average) selected for inclusion in the corpus, Ferro and her colleagues added a set of 8-10 questions of various degrees of difficulty [Ferro, 2000]. The corpus also includes one or more answers for each question in the form of a list of phrases or clauses (an "answer key"):

> The answer key was created to facilitate the automated scoring of these exams. Alternative answers were indicated where necessary:
>
> - Same answer but different ways of saying it:
>   - levels of granularity
>   Toronto, Ontario | Toronto | Ontario
>   - amounts of information given
>   he died | he died in sleep of natural causes
>   - wordings/paraphrases
>   Human Immunodeficiency Virus | HIV
> - Entirely different answers:
>   - Where did the boys learn how to survive a winter storm?
>   winter camping tips from a friend | their backyard
>
> [Anand et al., 2000, p.5]

The relationships between the answer keys are not given explicitly (the symbol | could stand for different relationships). Answer keys are also phrased by a human annotator. Some of them are extractions but often answer keys provide a reformulation of one or more substrings of the document. I added to the original corpus *human sentences* (HumSent, [Hirschman et al., 1999]), i.e. sentences extracted from the text that corresponded to each answer key, as well as *automated sentences* (AutSent), corresponding to all sentences for which the overlap with the human answer key is greater than 0.

In [Dalmas et al., 2003, Leidner et al., 2003], we describe how we redesign the corpus in XML by including several linguistic layers such as POS-tags, lemmata, stems

[10]`http://www.cbc4kids.ca`

and syntactic trees. We used this corpus to reproduce the Deep Read system's baseline described in Hirschman et al. We used the same evaluation metrics described in [Hirschman et al., 1999], namely *Recall*, *Precision*, *AutSent* and *HumSent*[11]. Recall and precision are measures of the word overlap between the human answer key and the system's answer:

Recall	$= \lvert cw_{sa} \cap cw_{ha} \rvert \,/\, \lvert cw_{ha} \rvert$
Precision	$= \lvert cw_{sa} \cap cw_{ha} \rvert \,/\, \lvert cw_{sa} \rvert$
With:	
	cw content words
	sa system answer
	ha human answer

HumSent and AutSent compare the sentence chosen by the system against the list of acceptable sentences among human sentences and automated sentences respectively, scoring 1 for a response in the list, 0 otherwise. Table 2.12 shows the average scores obtained with a baseline system that selects the sentence with the largest overlap with question stems.

Difficulty	# questions	Recall	Precision	AutSent	HumSent
Easy	237	0.74	0.18	0.75	0.74
Moderate	177	0.57	0.22	0.55	0.57
Difficult	67	0.49	0.19	0.43	0.43
Average	481	0.63	0.19	0.62	0.63

Table 2.12: Baseline evaluation using stem overlap by question difficulty.

As already noted, the questions constructed for the CBC4Kids corpus are rated as to their difficulty [Ferro, 2000]:

> "Easy: Uses exact wording from the text and/or the question and answer are close to each other in the text. [...] Moderate: Some paraphrasing from the text and/or the question and answer aren't close to each other in

[11]Notice that the definitions for P and R in [Hirschman et al., 1999] appear to have been swapped.

> the text. [...] Difficult: Very or entirely different words are used in question; lots of other tempting but incorrect answers are in the story; subtle knowledge is required to answer the question."

Table 2.12 shows the performance of the baseline system, broken down by difficulty class. For all scoring metrics other than Precision (P), the table shows a strong correlation between the retrieval score and the class assigned according to Ferro's guidelines for Q&A writing. As for Precision, it is not as significant because human answers are phrases and our system outputs a sentence as answer.

The advantage is that this corpus now provides a large set of answers per question: answer keys, automated sentences and answers from our baseline QA system, which defines three kinds of answers: inferred (paraphrases from answer keys), extracted answers (automated sentences) and incorrect answers.

The original corpus has also been studied by [Light et al., 2001] who provide an analysis of the number of answer occurrences per question, which is remarkably low compared to TREC data: *For example, 25% of the TREC-8 questions had only 1 answer occurrence in the text collection, while 80% of the CBC questions had exactly 1 answer occurrence in the targeted document.* Thus analysing redundancy as simple string matching will not be effective on CBC4Kids. Although CBC4Kids provides useful data for our research, it is quite small; it has the quality of being homogeneous, which is actually a problem, and if it is not completely domain specific, it is really related to Canadian topics. I chose to use TREC QA material and leave CBC4Kids for further research on a smaller dataset.

2.4.3 Evaluation Issues Relating to Fusion-based QA

The main issue in fusion-based QA evaluation is to assess the value of answer comparison, and eventually answer clustering. Re-ranking is an evaluation method based on the accuracy of answer selection at a given rank. When computing the MRR for a ranking strategy, the assumption is that the system is ranking single answers, and that an answer can be correct or incorrect (binary evaluation).

One of the experimental objectives of this thesis is to make use of answer comparison and information fusion to provide the end user with a more elaborate view of the answer as a whole, i.e. possibly alternative answers presented as distinct clusters of related answers, with eventually a representation of their respective inner structure (granularity, aggregation). There is currently no evaluation material for such 'golden clusters'. The CBC4Kids corpus [Light et al., 2001] provides questions about news stories, which have multiple answer key represented as a flat list of alternatives. Nevertheless, the given answers are not always alternatives, and can vary in granularity or phrasing. Their relations are also not annotated. Producing such 'golden answer clusters' for evaluation is difficult and might not be actually feasible, as it would require not only identifying answer strings but also judging their relationships. In the type of fusion for QA I describe in the next chapter, I also compare extractions that are not answers themselves, I believe their relationships to actual answers is worth investigating, but a manual annotation would require an enormous amount of work.

Most of the evaluation literature in TREC QA focuses on optimizing answer pinpointing and scoring [Lin and Demner-Fushman, 2005, Marton, 2006] for more and more complex questions. The main trend in current QA research is to improve the coverage of question types that can be correctly answered automatically: *What kind of questions are automated answering systems good at?*[12] Another example of this trend is the new TREC 2006 ciQA task proposed by Jimmy Lin and Diane Kelly, and whose goal is defined as follows:

> The goal of the complex, interactive question answering (ciQA) task within the QA track at TREC 2006 is to push the state of the art in question answering in two directions:
>
> - A move away from "factoid" questions towards more complex information needs that exist within a richer user context.
> - A move away from the one-shot interaction model implicit in previous systems towards one based at least in part on interactions with users.
>
> (TREC 2006 ciQA Task Guidelines[13])

While dialog and complex interaction are indeed challenging and worth investigating, I would argue that (1) the processing of factoid questions can still be improved, and

[12]This question was asked to the panel associated with the Knowledge and Reasoning for Answering Questions workshop at IJCAI 2005.

[13]http://www.umiacs.umd.edu/ jimmylin/ciqa

(2) interactions are likely to be very poor if a system only has a string representation of what an answer is.

In particular, I agree with [Sparck-Jones, 2003], who argues in favour of more *openness* towards provided answers. Hence, my focus on complex answers to simple questions (e.g. factoids) rather than re-ranking strings to answer complex questions. I now review some still unsolved problems with factoid questions, that I think should be investigated before moving towards more elaborate questions.

As in the example from Burger et al. discussed in Subsection 2.1.3, my first experiment had examples of *question ambiguity*. That is, a particular string in the question (e.g. *Taj Mahal*) had multiple denotations, and different answer candidates were appropriate to each. What I then saw in my larger experiment were cases of *answer ambiguity*, independent of *question ambiguity*. That is, there can be ambiguous answers to both ambiguous and unambiguous questions, as shown in the following example:

Where is the Danube found?

(A) *The town is home to Bulgaria's largest Danube port.*

(B) *The Danube is a light riding and draft horse found in Bulgaria.*

With an ambiguous question such as *Where is the Danube found?*, the phrase *the Danube* can refer to the Danube River (A) or the Danube breed of horses (B). But it turns out that Bulgaria hosts both: the Danube River flows through Bulgaria, and Danube horses are bred there. Thus *Bulgaria* turns out to be an ambiguous answer, because the evidence for it is of two completely different types. This can happen even when the question itself is unambiguous. For example, Scotland has two different towns named Tomintoul, both of which are places where one can ski in the winter. The question *Where can one ski in Scotland?* is itself unambiguous, but the answer *Tomintoul* is ambiguous, since the two places can be distinguished geographically. Such complexity among answers to factoid questions is difficult to assess with current TREC QA data since the track now only allows one answer per factoid question, making the pool of acceptable answers for one question very small and thus not representative of the data.

Wherever possible, systems should correctly handle answer ambiguity as well as question ambiguity. The main issue is the choice of an appropriate evaluation method to ensure that systems have a correct handle on the answers they provide, i.e. besides answer strings, one should also assess the respective justifications provided by the system. This relates in general to answer complexity (e.g. alternative answers, variations in granularity) and how this complexity is presented to the end user. [Hirschman and Gaizauskas, 2001] showed there had been much progress in QA, especially in fact-based QA for which the best systems can correctly answer more than 80% of the questions with very accurate snippets. However, they notice: *For first time users, it may be important to explain the limitations of the system, so that the user can understand how to interpret the answers returned.* Truly, an answer such as Atlantic City to *Where is the Taj Mahal?* may look unconvincing to a user that had some idea about the location and was seeking for a confirmation for instance. It looks like the system is not reliable, and requires the user to browse the document the string came from to look for a justification. At that level, it is actually faster to use a standard search engine such as Google that provides a list of text snippets against the query. Answers provided by QA system should be self-contained and satisfactory without extra browsing. In my later experiment with pictures (Chapter 7, Subsection 7.2.4), it also seemed that a rich context (a picture in that case) could also help identifying quickly when an answer is definitely out of topic.

Currently TREC QA is an information extraction task and the document from which the answer string has been extracted plays a role in the acceptance of the string as correct, i.e. to be correct, the answer must be *supported* by a document. For instance, to the question *Who is the President of the United States?*, an answer string such as *Clinton* was considered correct when it was extracted from a document written when he was in office, but considered incorrect otherwise. Fusion somewhat generalizes this issue. Using fusion, *Clinton* is likely to have found support from both kinds of documents. A justification should be given, besides the answer string, to assess why the system perceived this string as an answer.

Hence, besides the question *What is a good answer?*, I would also like to ask *What is good evidence for a given answer?* In the current TREC task, the context of extraction determines the correctness. However, when an answer is generated from different documents as it happens with information fusion, what proof is to be given? Systems may provide the document that best justifies the interpretation of the answer, but it is not clear yet why one document is better than another, at least in open domain QA. (In specialized fields, for instance the medical domain, there is a qualitative ranking of evidence for an answer[14].)

Besides, in TREC QA, alternative answers to *Where is the Taj Mahal?* were judged incorrect, but on another ground than supporting context (as it was for the *Clinton* case): *Unless the question specifically stated otherwise, we assumed that any question regarding a famous entity was asking about the famous entity (...) we accepted only Agra, India, not the Taj Mahal casino in Atlantic City, New Jersey, nor the Taj Mahal Hotel in Bombay.*

My point of view is that both *Clinton* and *Atlantic City* should be accepted as long as a justification is provided. Assuming that the user is actually asking about *the* Taj Mahal is a user profiling decision. Such decision should come after answers have been found because it relates to how informative one has to be regarding a question. For instance, answering *Paris* or *France* to the question *Where is the Eiffel Tower?* when asked by a French user is unlikely to be extremely informative, a preferred answer would be the name of the street or a metro station. As argued by Voorhees, the required granularity level depends on the end user and systems in the long run have to adapt to each requirement. But before making decisions about such requirements, we should try to have systems providing good justifications for their answer.

Justifications were actually introduced temporarily in TREC 11 [Voorhees, 2002] but removed later on. Most participants used the context surrounding the answer string as a justification, but these contexts were not evaluated. Currently, TREC answers must be supported by a document, which has a cost in human assessment. Fusion-based techniques would make this assessment even lengthier, with potentially several documents provided as support for a single answer. I would thus propose systems to provide a justification that should be self-sufficient, and evaluated without having to refer to any document.

In the type of fusion I propose, some extractions, that are not answers themselves but related to actual answers, could provide short and effective justifications. Such

[14]http://www.clinicalevidence.com/ceweb/about/put_together.jsp

nodes could disambiguate the question if connected by fusion to a question term (e.g. *casino* versus *Mughal architecture* for the Taj Mahal). If they were connected to actual answers, they could help interpret the answer: Answering *When was the telegraph invented?* with *1844* associated to *Morse*, besides *1837* to *Wheatstone* and *Cooke*, is clearer than a non justified list of dates.

To sum up, I believe it is a requirement for QA on free-text to provide, besides a well pin-pointed answer, the *perspective* in which the answer was found. I use the term *perspective* along the lines of [Cardie et al., 2004] on multiple-perspective QA. This research addresses questions about opinions, e.g. *Has there been any change in the official opinion from China towards the 2001 annual U.S. report on human rights since its release?* Cardie et al. envisage a *multi-perspective question answering* that views the task as one of opinion-oriented information extraction, i.e. complex questions. However, factoid questions, although simpler, already address similar issues as witnessed by their multiple answers. Multiple answers are not always due to different opinions *per se*. For instance, differences in granularity or phrasing do not indicate different opinions: They may still rely on the same *perspective*. Alternative answers such as the invention of the telegraph may also not indicate inconsistency on the specific date but what one is considering as the first telegraph.

Current evaluation methods in opinion-oriented QA, research in modeling external knowledge from multiple streams [Jijkoun and de Rijke, 2004, de Chalendar et al., 2002, Yang et al., 2003a, Yang and Chua, 2002] and QA oriented towards event recognition (e.g. issues addressed by TERQAS, an ARDA Workshop on Time and Event Recognition for Question Answering Systems) are investigating such issues. Research in general text clustering for IR [Zamir et al., 1997, Zamir and Etzioni, 1999] has also addressed similar issues regarding assessment of the clustering process besides its accuracy.

2.5 Summary

In this chapter, I showed that multiple answers do occur, whatever type the question belongs to. Answer variability is a general problem that has been studied very little.

While information fusion is not a recent technique, the idea of comparing and merging answers is novel to QA. From summarization over multiple documents, we know that there is no correct or incorrect summary, just differences in the quality in reporting commonly admitted facts, as well as divergent opinions or beliefs. It seems to me that this paradigm is also valid for QA: There is no correct or incorrect answer, but answer variability has to be acknowledged to provide the user with better, more understandable answers. In the research covered here (summarization, document fusion and clustering), we can see a tendency in the way results are presented: Comparison and merging are used to post-process a first pass of raw results (e.g. extractions in summarization, or a list of documents in IR). This post-processing is based on the material found, and does not necessarily depend on the query (IR), or input topic (summarization): It has a bottom-up quality. Such a technique also lends itself to generation, either of natural language, or of new structures such as clusters. The question is how do we adapt fusion for QA specifics? In the next chapter, I address the four main points that have been raised in previous research on fusion: What level of segmentation do we choose for the entities to be fused (e.g. sentence, paragraphs)? What relationships should be used in comparing such entities? How do we present or generate answers that come from fusion? How do we evaluate a QA system based on fusion?

Chapter 3

Approach and Scope

Much has been written about the qualities of a good question but little about the qualities of a good answer [Ely et al., 2002].

Although written in the context of medical Question Answering (QA), Ely's remark is actually more broadly relevant. As seen in the previous chapter, research in automated QA has focused on precisely characterizing questions, in order to retrieve correct answers. This includes deep parsing, use of ontologies, question typing and machine learning of answer patterns appropriate to question forms. In such a context, answer candidates are seen as *competitors* and ranked according to individual features such as match with the expected answer type and number of question words in context. This ignores potentially relevant relations between answer candidates. In recent work, frequency counts of answer candidates, i.e. equivalence by string matching, has proved to be useful to identify correct answers. In this thesis, I investigate further relations and focus on the analysis of answer candidates as potential *allies* rather than candidates, in the belief that their relationships can be exploited as well as individual features.

Illustrating this intuition is the infamous *Where is the Taj Mahal?* example from [Burger et al., 2002]. There are several Taj Mahal in the world, the most famous one being in *Agra*, *India*. Recognizing that the two distinct strings, *Agra* and *India*, apply to the same world referent and building an answer cluster {*Agra, India*} increases the likelihood of either candidate being an answer through the cumulative frequency counts of both occurrences. Considering *Agra* and *India* as competitors reduces that likelihood. The same goes for other correct answer candidates such as *Atlantic City*, *New Jersey*. The first intuition is thus to try to automatically discover such relations between answer candidates.

This research is grounded on a property of QA over free text: Answers can appear in many places and in many forms (see Tables 2.2 and 2.3). My general direction is to (1) exploit this property to improve QA accuracy, and (2) investigate appropriate ways of processing and rendering answers to acknowledge and explain their multiplicity to the end user. My approach is based on four areas of investigations, that I present in the following sections:

- What I call *information projection* is the first step required for fusion, when we need to decide which entities are going to be fused and compared.
- The second step is the *discovery of relationships* among the projected entities. I focus on inference of the equivalence and inclusion relations, which I used in my experiments (Chapters 4 and 6). I will also discuss relations indicating an aggregation and an alternative between entities as a means of identifying similarities and differences between answers.
- Although it is not currently taken into account in the community, I believe *rendering* is crucial for QA. Rendering is about constructing or generating an answer for the end user. It involves choosing both content and medium that are appropriate to a given user and for a given question. It relates to user modeling, which is still very minimal in QA.
- Finally, *evaluation* is the last area of concern. In this chapter, I focus on existing methodologies and material. But evaluation in QA is subject to on-going research and discussion. After having presented my experiments that use a known evaluation methodology, I will defend an approach, that might be considered lenient compared to state-of-the-art QA, but has a wider coverage in terms of assessing answer *quality*, i.e. an approach within which *correctness* (truth value of an answer) is replaced by a quality assessment of the evidence in favour of a given answer, which is, in fact, the original source of disagreement between judges about answer *correctness*.

3.1 Information Projection

The first step of information fusion is to project the input on a common board as the pieces of a puzzle to be composed. Implementation-wise, I consider those pieces as a graph's nodes, to be compared and eventually linked in further inference. In the terminology I will use, the process of mapping some input into the graph is called *projection*.

The *projection* process concerns two types of information, (1) the information provided by the question, and (2) the information found in the answering material. I use the term *answering material* as a general name for any set of data assumed to be relevant to a given question. This set may vary in size and homogeneity. It could be documents, or snippets retrieved by a search engine for the question considered as a query from a corpus (for instance newswire for the TREC QA evaluation or the web for other QA applications).

The main aspect I explore here is the size and types of units to be projected and the trade-offs involved. In previous work in automated summarization, document fusion and clustering, that I referred to in the first chapter, there was this recurrent question about choosing the best granularity level for fusion. Semantic and pragmatic information are conveyed at different levels, e.g. syntactic, lexical, morphological. As a consequence, the technique chosen for *normalizing* or *standardizing* the information contained by such entities once extracted is crucial because the whole comparison will rely upon it.

So far, I have used the term *extraction* as a broad term to refer to answer candidates. Extractions can actually be of several kinds: a passage, a sentence, a phrase or simply a keyword. In TREC QA, there are currently (2004) two standards: sentence level extractions for definition questions and phrase level extractions for factoid questions. From the perspective of automated comparison of extractions, especially those coming from heterogeneous and redundant data such as the web, passage or sentence level extractions are difficult to fuse because they are composed of units that overlap but also differ.

The example in (1) shows two sentence level extractions selected as answer candidates for the definition question *What is severance pay?*. Providing one or the other sentence would be a correct answer. A more complete answer would be to fuse information contained in both sentences. To do so, one would like to compare intra-sentential units such as *the employee*, *an employee of the Public Service* as well as larger chunks: *after the employee is dismissed* versus *upon termination of employment*.

(1a) *Severance pay is the money paid to an employee, not including wages and back pay, after the employee is dismissed*
(1b) *Severance pay is an entitlement that may be payable to an employee of the Public Service upon termination of employment*

From this example, it seems best to focus on the phrase level. However the same problem of atomicity occurs in phrases as well:

(2a) *The Nina, the Pinta and the Santa Maria*
(2b) *PINTA, NINA, AND SANTA MARIA*

Stating that (2a) and (2b) are actually equivalent answers requires the coordination to be split. On the other hand, a phrase such as *Bonnie and Clyde* extracted as an answer candidate to *What's the famous movie with Warren Beatty and Faye Dunaway?* should not be split.
Comparison could also be required on the morphological level, such as in (3), where it is needed to insure that the three extractions refer to the same age.

(3a) *19-year-old*
(3b) *19-year old*
(3b) *19*

From a computational point of view, without knowing beforehand what it is relevant to split or not, extracting and splitting at all levels is far too expensive to be technically feasible. The challenge is to project with high precision while insuring good coverage. In order to make a decision on the type and size of extractions to be projected, I assessed what is the most frequent answer span and answer form in TREC QA 10, 11 and 12 using answers retrieved from the web.

I chose TREC QA data because answer patterns (regular expressions that match a correctly judged answer) are available for evaluating extractions. However, I used answers retrieved from the web rather than answers given by TREC QA systems (they are available in judgment files) for two reasons:

1. To assess the accuracy of answers on the sentence level for any kind of questions. TREC systems produce such answers for definition questions only.

2. To evaluate whether web answers (i.e. correct answers found on the web but not in TREC QA data) were different from the ones found in AQUAINT. TREC QA judgments are extracted from a specific corpus (AQUAINT) and thus the kind of extractions may be dependent on this corpus. I manually collected web answers to evaluate web data, and check whether web answers could vary in size from TREC answers.

TREC QA 10 and 11 both comprise 500 questions, 433 and 444 of which have answers in the TREC corpus respectively. For TREC 12, I used the set of 380 factoid questions having answer patterns. (For technical reasons, definition and list questions do not have a set of answer patterns).

For each question, I generated a query based on the question's keywords and retrieved the top 100 snippets from Google for that query. I then counted how many questions were answerable for three levels of extractions using TREC answer patterns alone:

(a) sentences extracted from Google snippets (Notice these sentences are not always grammatical sentences as they are partially split to form snippets. By snippet I mean the list of sentences presented by Google as a block to describe a document.)

(b) nominal phrases (NPs) and prepositional phrases (PPs) extracted from the corresponding POS-tagged sentences. Phrases were extracted using a chunking technique. If the chunk was preceded by a nominal phrase, the attachment was preserved (for instance *animals without backbones*, *90185 people*, *in 1904* are all valid extractions). Coordinations however were split.

(c) 1-token span answers, or keywords, simply extracted by removing stop words from sentences.

Here is an example of each extraction strategy:

Question	*When were William Shakespeare's twins born?*
Query	William Shakespeare twins born
(a)	*In 1585 , twins were born and baptized Hamnet ...*
(b)	*In 1585*; *twins*; *Hamnet*
(c)	*1585*; *twins*; *born*; *baptized*; *Hamnet*

The sentence level is expected to be the format that covers (answers) most of the questions, because it is the largest one. Thus, the score of the sentence level serves as the optimal score.

Table 3.1 shows the number of answerable questions with each type of extraction for each TREC track. Percentages are relative to the number of answerable questions. In both TREC evaluation exercises, NPs/PPs extracted preserved 96% of the answers matched in the original Google sentences (i.e. the loss between sentence level extraction to phrase level was 4%). Thus we see that few questions expect a full sentence or a verbal phrase as answer. Another result is that 2/3 of answers are 1-token keyword, i.e. most answers are contained in one word.

Table 3.1: Comparison of different extraction levels for TREC 10, 11 and 12

TREC	10		11		12	
# questions having an answer (non NIL)	433		444		380	
(a) # answerable with Google sentences	362	84%	384	86%	294	77%
(b) # answerable with NPs and PPs	349	80%	370	83%	285	75%
(c) # answerable with 1 token	280	65%	265	60%	209	55%
Loss from (a) to (b)	13	4%	14	4%	9	3%
Loss from (b) to (c)	60	17%	105	28%	76	27%
Loss from (a) to (c)	82	23%	119	31%	85	29%

I then assessed web answers for TREC 10: For each question, I collected strings from the top 100 snippets that were correct answers but not among TREC patterns and extended the set of answer patterns for TREC 10. I used my own judgment to assess answer correctness, based on my world knowledge and the evidence provided in snippets. Overall, I added 2167 answer strings to the initial collection, about 4 strings per question. (Note that I added strings, not regular expressions that can match several answers, this the reason why the number of web strings is large compared to the initial TREC 10 collection of 1227 patterns.)

Results follow in Table 3.2:

Table 3.2: Comparison of different extraction levels on TREC 10 using an extended set of patterns that include web answers

TREC	10 web	
# questions having an answer (non NIL)	475	
(a) # answerable with Google sentences	433	91%
(b) # answerable with NPs and PPs	418	88%
(c) # answerable with 1 token	344	72%
Loss from (a) to (b)	15	3%
Loss from (b) to (c)	74	18%
Loss from (a) to (c)	89	21%

Web patterns naturally improved the recall of the system. However the loss from sentences to phrases is similar to an evaluation based on TREC patterns only. The amount of 1-token based answers is similar to the one in TREC answers.

Consequently, I chose to focus on the simpler spans, i.e. 1-token based answers and NPs/PPs segments, as they represent an important proportion of the answers in the current QA task and can simplify answer modeling considerably.

The original hypothesis I made is that using answer multiplicity can improve both accuracy and quality of answers. To assess this hypothesis, I chose nonetheless to assume that the answering material contains some candidates that are incorrect. This is a realistic claim for two reasons: (1) QA is still far from providing 100% accurate answer extractions (although the best score on factoid questions at TREC QA 13 was 77%, the median score was 17%, which shows that the average QA system performance is still low), and (2) an answer can be composed of elements that, considered independently, are not answers (e.g. *Pinta ship* alone is not a correct answer) or simply relate contextually to correct answers (e.g. *Morse/1844* for *When was the telegraph invented?*). Thus, while my first aim was to study multiple answers, as in multiple *correct answers*, I discovered through my experiments that incorrect but related answers helped the comparison process during fusion. Hence, the projection considers multiple *answer candidates* in general, whether they are actually correct or not. For that reason, when performing answer comparison, I also use the term *information fusion* rather than *answer fusion* [Girju, 2001], as the merging involves extractions that are not always correct answers. I mentioned earlier research focused on answers by [Buchholz and Daelemans, 2001] and [Webber et al., 2002]. Both approaches assume that the system has found correct answers and the considered relations are between correct answers only. In the following section, I consider relationships between both correct and incorrect candidates, as well as question words.

3.2 Relationship Discovery

Once phrases are projected as nodes of the graph that will serve as the data structure for fusion, edges are drawn between nodes to identify relationships. I focus on the three main relationships I used for my experiments: the equivalence ($\leftrightarrow$), inclusion ($\rightarrow$) and aggregation ($\leftrightsquigarrow$) relationships. The list of techniques discussed here is non exhaustive, and stands for comparison heuristics I have tried during my experiments.

3.2.1 The Equivalence Relation

I consider equivalence as a broad relation including string matching, similarity and synonymy. Table 3.3 lists different techniques I have used to infer equivalence.

Table 3.3: Techniques for inferring equivalence

Techniques	**Features**	**Examples**
Lemmatization	Lemma	illnesses $\leftrightarrow$ illness
Stemming	Stem	ill $\leftrightarrow$ illness
Abbreviation, Acronym	Token	US $\leftrightarrow$ United States
Edit-Distance	Stem	Hindenburg $\leftrightarrow$ Hindenberg
WordNet synonym	Lemma, POS-tag	treaty $\leftrightarrow$ pact
Patterns	Token	bipolar disorder $\leftrightarrow$ manic-depression

All these techniques use case-shifting for comparison except for acronym recognition. Lemmata and stems, as encoded in node features, are all lower case.

Lemmatization is a string matching technique. Automated stemming, as implemented in [Porter, 1980], is not completely reliable. For instance, *capital* and *capitalism* have the same stem with Porter's algorithm (*capit*). Thus, I classify Porter's algorithm along the Edit-Distance algorithm, i.e. as providing an indication of similarity. I implemented the lazy version of the Edit-Distance algorithm described in [Allison, 1992], which has proved to be more efficient than the standard version[1]. To be considered equivalent, the distance between two words has to be lower than a threshold, varying depending on the length of the words compared (heuristically 20% of the average length).

The pattern-based technique involves either looking up the current specific set of sentences with patterns such as *X, also known as Y* or querying the web with an instantiated pattern (*bipolar disorder, also known as manic-depression*) and accepting it as a correct inference if the number of results reaches some threshold. Several methods have been proposed to learn such patterns from a corpus [Lin and Pantel, 2001, Ravichandran and Hovy, 2002, Snow et al., 2004]. Patterns can then be used either directly on the search results, or indirectly by building a database of relationships between words from a training corpus for future look-ups (in a usage that is similar to WordNet).

[1]I am using a language featuring laziness, Haskell.

3.2.2 The Inclusion Relation

As for the equivalence relationship, inclusion refers to a broad family of relations: hyponymy, hypernymy, meronymy, membership and contextual inclusion. Table 3.4 provides a list of techniques that can be used for the discovery of this relation.

Table 3.4: Techniques for inferring inclusion

Techniques	**Features**	**Examples**
Lexical head	Lemma/Stem	expert $\rightarrow$ sunspot expert
Bag subset	Lemma/Stem	sun $\rightarrow$ sun core
WordNet hyponym	Lemma, POS-tag	symptom $\rightarrow$ vertigo
WordNet substance meronym	Lemma, POS-tag	water $\rightarrow$ hydrogen
WordNet part meronym	Lemma, POS-tag	Scotland $\rightarrow$ Edinburgh
WordNet member meronym	Lemma, POS-tag	European Union $\rightarrow$ France
Patterns	Token	medical condition $\rightarrow$ epilepsy

(WordNet also contains verb pointers such as causation (kill $\rightarrow$ die) or troponym (walk $\rightarrow$ march), as well as noun derivations (die $\leftrightarrow$ death), which would be useful for projections based on both NPs and VPs but I focused on nominal comparison).

Pattern based inference and WordNet lookups are the same techniques described for equivalence inference except that looked-up relations and patterns are specific to the inclusion relationship, for instance, X *such as* Y (*medical conditions such as epilepsy*) is a typical search pattern to infer an inclusion from X to Y.

Lexical head and bag subset both use the same shallow technique computing word co-occurrences. They can be computed over a list of stems or a list of lemmata provided as a node feature. Lexical head based inclusion is drawn between a node *X* and a node *Y* when the lemma (or stem) list of node X is a suffix of the corresponding list of node Y, e.g. *[chinese, medicine]* $\rightarrow$ *[traditional, chinese, medicine].*

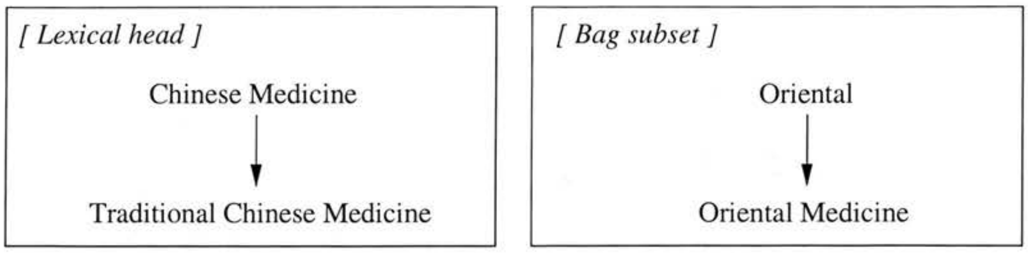


Figure 3.1: Inclusion based on lexical head versus bag subset

Computation on bag subset is more lenient because it does not take into account word order: As long as the list from X (eventually reduced as to form a set by removing redundant tokens) is a subset of the list from Y, then a relation $X \rightarrow Y$ is drawn as shown in Figure 3.1.

3.2.3 Aggregations and Alternatives

In the first experiment I describe in Chapter 4, edges generated in graph models only reflect the inference of inclusion/entailment and equivalence. In Chapter 6, I will add a shallow inference of the aggregation relationship.

The aggregation relationship is meant to measure contextual relatedness between two nodes. I will show later in experiments on answer flexibility that equivalence and inclusion are not sufficient on their own to cluster candidates that refer to the same "answer", i.e. a referential entity that can be described using paraphrases (equivalence), different levels of granularity (inclusion/entailment) but also that can present different aspects or attributes, that are not related in terms of equivalence or inclusion, but do refer to the same entity. For example, vertigo can be defined as *dizziness*, *lightheadedness* (equivalence between candidates), or as a *medical symptom* (entailment), but it can also be described as a *fear of heights* (aggregation). However, if one refers to Vertigo, the *movie*, *film* (equivalence), or more exactly a *thriller* (inclusion), then the candidate *director Alfred Hitchcock* is to be aggregated with nodes referring to the movie, not the symptom.

Table 3.5: Technique for inferring aggregation

Techniques	**Features**	**Examples**
Co-occurrences	Stem	*anatomic* $\leftrightsquigarrow$ *body*

Computation of term co-occurrences can give an indication of such relatedness. I consider that two nodes X and Y are aggregated

if $|C_x \cup C_y| > 1$

and $|C_x \cap C_y| \,/\, |C_x \cup C_y| \geq 0.5$

where C_x represent the context for the node X (and C_y the context for Y). Such a context can be defined as a set of discrete units, for instance the set of sentences or documents in which the stem of node X occurs. For example, if *anatomic* occurs in sentences S1, S2, and S3, and *body* occurs in S2, S3 and S4, then the context union (S1, S2, S3, S4) is indeed greater than 1, and its intersection (S2, S3) represents half of its union: An aggregation edge is drawn between the two (Table 3.5).

In this thesis, I did not consider a relationship that would indicate an alternative relation between two nodes. Instead, I used the absence of path (indicated by either a direct edge drawn between nodes or transitivity) to denote alternatives. Specifically, in Chapter 6, Section 6.3, I consider different heuristics to cluster related nodes together, thus providing a list of answer clusters rather than a list of single answers. I will show how such clusters may or may not map to actual alternatives in answering. In some cases, the lack of connection rather reflects the system's lack of knowledge or reasoning ability, and thus should not be confused with an alternative. It would be relevant to have systems able to infer a positive relationship indicating a barrier to relatedness between two nodes. However, I did not investigate this aspect in the thesis.

Finally, once answer candidates have been normalized and projected into nodes, and edges representing different relationships have been drawn, the graph model stands for an enriched representation of the answering material available for a question. Graph features allow for different strategies of filtering and selection of nodes, which can then be rendered and presented to the end user. While in traditional QA, extracted answer candidates are directly presented "as is" to the end user, I propose to apply the

the terminology and design framework provided by the Model-View-Controller design pattern to show that eventually QA systems could generate different sorts of answers, depending on the task or the end user preferences.

3.3 User Interface Aspects

I consider that, in order to move further towards more user-oriented answers, we need to produce appropriate answers in terms of both content and rendering. To engineer such answers, I propose to apply the Model-View-Controller design pattern to QA.

The MVC [2] is a design pattern used mostly to solve difficulties encountered when designing graphical user interfaces (GUIs). Its primary goal is to facilitate interactions between the end user and the data being manipulated.

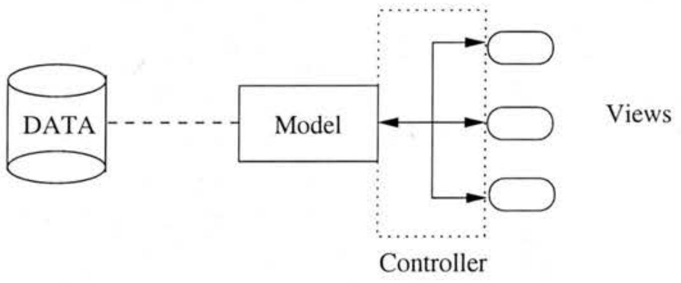


Figure 3.2: The Model-View-Controller design pattern

As shown in Figure 3.2, the model is a formal representation of the data to be processed. Views are user-oriented representations of the model. For example, recent HTML editors provide both a user view showing the rendered web page and a programmer view showing the code used to produce the page. Views are dynamic and can be updated on demand. The controller acts to inform the model of possible changes and select the information to be propagated to the user. The rendering is then accordingly updated.

An MVC-based approach to QA establishes a clear distinction between (1) extracted strings relevant to a question (considering multimedia QA, relevant raw data

[2]MVC was introduced by Trygve Reenskaug at Xerox Parc in the 70s, but the first significant paper is [Krasner and Pope, 1988]

could actually be in another format), (2) the normalized entities they denote and the relationships between them – i.e., the *model* in MVC terms – and (3) *views* of the model, which captures the fact that the same answer can be expressed in various ways: short, detailed, in context, with alternative answers, and in different modes: picture, video, sound, text, speech, or a formal data structure. Within this framework, an *answer* is a structured object contained in the *model* and retrieved by a *strategy* to build a *view*. This strategy is comparable to the *controller* in MVC in the role it plays between the model and the front-end user, in terms of what *content* should be provided and how to *render* it.

In Information Retrieval (IR), several "views" have already been proposed. Instead of a sorted list of documents, [Cugini et al., 1996] provide results in a three dimensional space and relate neighbour results. WEBSOM [Honkela, 1997] uses the Self-Organizing Map algorithm to cluster results into neighbourhoods sharing similarities in a two dimension space. The recent IR engine, Clush[3], proposes results clustered by relations. For instance a query such as *car* returns a list of sorted documents and topic areas classified by relations. The topic *Parts of* for this query contains: *air bag*, *accelerator*, *automobile engine*, *auto accessory*.

QA has not yet considered the issue of *views*, as evaluation only considers the correctness of extractions (answers being checked using regular expressions). Now, *Agra*, *India* and *Atlantic City* are all correct extractions to the question *Where is the Taj Mahal?* But a view that recognizes *Agra* and *India* as part of the same answer will require a more formal notion of what an answer is – not just to generate answers, but also to evaluate them. Finally, a rendering that can automatically generate a map for such questions instead of a list of locations, would be even more appropriate. In order to generate such responses, one needs to distinguish between the content of an answer and its rendering.

In Chapter 5 of [Lehnert, 1978], the author defines the term *content specification* for the answers generated by her system. (Implicitly, an *answer* refers to the content of the string generated in English.) According to Lehnert, the content specification instantiates three separate dimensions:

[3]http://www.clush.com

1. intentionality
2. elaboration options
3. category-trace instructions.

In the MVC terminology, those are actually features of the view, i.e. rendering choices. *Intentionality* defines the user dimension and how the system is supposed to deal with it, for instance, the system could be cooperative, rude, honest or deceptive. This belongs to choices made on the view. That is, what is presented and phrased in different manners depends on the same content. Even when the system aims at misleading the user, the answer can be seen as the negation of the actual answer content. (There is on-going research on agents with personality traits [Isard et al., 2003] that could be plugged into renderers). *Elaboration options* (for instance verification or short-answer options) defines the amount of information to be given to the user, as in the following example from [Lehnert, 1978]:

Q: *Did John go to New York?*
A: *Yes* (short answer option)
A: *Yes, John went to New York by bus* (verification option)

Category-trace instructions are *designed to construct answers that reflect the interpretative process of a question* and are mandatory (except if the system's attitude is to be uncooperative), as in the following example from [Lehnert, 1978]:

Q: *Do you eat out very often?*	Q: *How often do you eat out?*
A: *Yes, about twice a week.*	A: **Yes, about twice a week.*

In current state-of-the-art evaluations, there is no clearly defined distinction between answer content and answer rendering. *User modeling* (which could affect content selection and rendering) is also limited. Instead, taking the example of TREC QA, there are rules such as: an answer must be a string extracted from a supporting document, and the string must be *exact*. In practice, the latter rule has translated into short versus long answers. This notion of *exactness* (Section 2.4) has raised debates among participants, and I believe it is because it confuses answer content (accurate selection of informative text) and answer rendering (short string).

3.4 Evaluation Methods

This thesis is concerned with the evaluation of information fusion for QA. My work hypothesis is that fusion can improve QA performance on two levels:

1. by increasing the general accuracy of QA as measured by improvement in traditional answer re-ranking.
2. by providing a flexible and rich representation to handle and make use of answer multiplicity, by re-ranking for instance answer clusters rather than single answers.

In Chapter 4, I present an experiment on location questions to be evaluated using traditional answer re-ranking: For each question, the output consists of a list of single answer candidates which are then assessed as "correct" or incorrect". The question recall at rank 1, $qrr(1)$ or 1st-rank score, is a measure that indicates the percentage of the N questions that have been correctly answered by the first ranked candidate c.

$$\begin{aligned} qrr(1) = \; & \textstyle\sum_i^N correctness(c_i) \;/\; N * 100 \\ & \text{where} \\ & correctness(c_i) = 1 \text{ if } c \text{ is correct} \\ & correctness(c_i) = 0 \text{ otherwise} \end{aligned}$$

I will also use the Mean Reciprocal Rank (MRR) to compare results between a baseline and a fusion-based strategy.

$$\begin{aligned} MRR = \; & \textstyle\sum_{q=1}^N \frac{1}{r_q}/N \\ & \text{where} \\ & r_q \text{ is the rank of the first correct answer candidate for the given question.} \end{aligned}$$

In Section 2.4, I identified key issues in QA evaluation, such as the notable difficulty of assessing the absolute truth value of a given answer (answer correctness). While judges might find themselves in agreement with the answers provided by a corpus of newswire such as the TREC AQUAINT corpus, assessing the absolute correctness of answers found in opinion-oriented data, such as blogs, might be more difficult, unless judges are asked to assess the value of the evidence as the basis for belief, rather than the answer itself.

Also, when answers are coming from fusion and represented as clusters, we need new scoring measures to assess the quality of an answer cluster, as well as a method to assess that it is well supported by evidence. A cluster may contain answers that have been found in different documents, and may not all be well supported.

In Chapter 5, I propose new scoring measures to evaluate answers coming from fusion. I present a small corpus based on TREC questions for which web answer candidates have been automatically collected from web searches. Those web answer candidates were manually rated on two scales, (1) for evidence, checking that a candidate is contextually well supported, and (2) for exactness, indicating that the candidate is a well pin-pointed string, and that it is informative enough as an answer. Instead of a binary evaluation, correct versus incorrect, I considered different levels of satisfaction, defined in term of evidence and exactness. From these ratings, I define new scoring measures to assess clusters' evidence and exactness, as well as the overall precision and exactness of clustering (Subsection 5.1.3).

Because the web corpus was manually evaluated in an exhaustive manner (all possible answers are known), I provide scores based on the truth table for satisfying answers found versus existing satisfying answers, especially answer recall and answer precision to assess more specifically how well a system handles multiple answers as a whole, rather than simply the first ranked answer for each question (which is a limitation of scores like MRR). I will show that despite a good MRR, or a good question recall (measured on the basis of the first satisfying answer), the actual performance over multiple answers can still be low.

The evaluations of fusion I propose are essentially *extrinsic*, task-based evaluations, where the task is answer re-ranking, where an answer can be a single string, or a more complex expression, such as an answer cluster. This is to be put in contrast with a possible *intrinsic* evaluation of the graph models, i.e. an assessment of the projection of extractions into nodes and the inference of edges as relationships. However, such an evaluation would be difficult and too costly to realize, even when dealing with a relatively small input. For instance, in my later experiment on answer flexibility (Chapter

6), 50 questions with on average 106 candidates per question generated overall 10312 edges, and this for a fixed set of shallow techniques used to infer relationships. Adding new techniques would require to assess again each of the 50 generated models. Instead, I describe extrinsic evaluation tasks assessing the views extracted from models. This allows the evaluation of systems based on different modeling techniques, as long as they can produce the same type of view.

3.5 Summary

The scope of this thesis is to assess the value of information fusion within the QA paradigm. My approach draws on previous work in summarization from multiple documents, making use of graph-based representations to compare similarities and differences [Mani and Bloedorn, 1999, Barzilay et al., 1999]. I present a QA system that takes as input extractions and build what I call a graph model, in which nodes represent normalized answer candidates, and edges relationships between candidates. I use the four types proposed by [Webber et al., 2002] to formalize the possible relations between answers: equivalence, inclusion, aggregation and alternatives, although I will extend these relationships to candidates that may not be answers, but simply related, contextually, to actual answers.

I also propose to make use of the MVC design pattern, both for engineering purposes, as well as a clarification in QA terminology. By distinguishing between data (corpus), representation (graph model) and rendering (views of the model), we can re-define the term *answer* in a more rigorous way. Previously, the term could apply to *answer candidate extractions* found in the corpus as well as the output of a QA system, consisting of list of *answer strings*. When performing fusion, there is not a necessarily direct mapping between found extractions and output strings, answer candidates are merged, normalized. The final output, although based on extractions, actually corresponds to a process of answer generation or rendering, in MVC terms. The MVC framework also allows to generate different views from the same model, i.e. answer rendering can vary, according to a given task, or given user preferences.

In the next two chapters, I present several experiments to assess the quantitative value of fusion for answer re-ranking, as well as the flexibility of graph models to generate different types of views from the same model.

Chapter 4

Fusion on Location Questions

The goal of the following experiment is to assess the value of answer candidate comparison for re-ranking. Most QA system architectures include a re-ranking component, which takes as input a list of candidate extractions found in documents or passages retrieved by the IR component, and outputs a new re-ranked list. Re-ranking is achieved by scoring candidates, for instance, on the basis of their frequency in retrieved documents [Abney et al., 2000, Clarke et al., 2001], and/or on how well each candidate can be validated as an answer by checking other sources such as WordNet or the web [Brill et al., 2001, Harabagiu et al., 2001, de Chalendar et al., 2002, Lin, 2002].

A traditional approach is to consider such candidates as competitors to a question and score each of them separately. In this experiment, I investigate candidates as potential allies and a re-ranking that takes into account their mutual relationships as well as their individual connections to the question. In order to assess the value of answer candidate comparison and overall fusion, I compare two re-ranking strategies:

- A **baseline** strategy scores candidates according to their relation with the question only.
- A **fusion**-based strategy makes use of the relations between the question and candidates, and among candidates themselves.

Figure 4.1 schematizes both strategies.

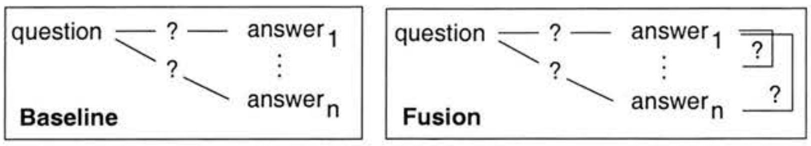


Figure 4.1: Re-Ranking strategies: answers as competitors (baseline) versus potential allies (fusion).

The hypothesis is as follows: Since multiple correct answers are not a rare case, if we find a way of clustering answer candidates automatically, a strategy based on fusion that takes answer connectivity into account, should improve the baseline results.

[Clarke et al., 2001, Brill et al., 2001] demonstrated that answer redundancy, measured by the equivalence relation based on string matching between two candidates, improved re-ranking. In this experiment, I focus on other relationships: inclusion, entailment as well as non-trivial equivalence. In the evaluation material (see next section), the list of answer candidates to each question does not contain any trivial redundancy (i.e. string matching candidates) for this purpose.

A second objective of this experiment is to assess the actual feasibility of answer comparison, with the following questions in my mind:

1. How will fusion perform with the given amount of incorrect answer candidates (at least 50%, see Section 4.1)? Is connectivity going to help if most connections involve incorrect answer candidates?

2. Are the relationships used for fusion (equivalence, inclusion, entailment) adequate for answer re-ranking? What is the advantage of discovering specific relationships, versus clustering candidates using a unique similarity measure for instance?

In the next section, I describe the evaluation material used for this experiment. I specifically focused on a type of factoid question, location questions, and I will explain the reason of this choice. Next is an overview of the system architecture, which is based on the graph modeling explained in Chapter 3 (projection, relationship discovery, rendering). Then I describe the specifics of answer model generation for this experiment and the techniques used for relationship discovery. From there, I present the baseline versus fusion features used for re-ranking, based on the properties of the graph models. I will mention the MVC paradigm for rendering, although, in this experiment, the final output will be limited to a text version listing one answer candidate per line. Finally, I discuss the results and analyze the generated models.

4.1 Evaluation Material

In TREC QA, systems cannot provide more than one answer to a factoid question, while, for other questions, a list is allowed. Previously (in Subsection 2.1.2 about answer multiplicity), I argued that this distinction does not necessarily hold and factoid questions do expect multiple answers. This is especially true of location-type factoids from TREC QA data, with nearly two thirds of such questions answered by several (and distinct in terms of string matching) extractions (Table 2.3). To emphasize the fact that answer multiplicity can also be used in the context of answering factoid questions, I selected a subset of questions from TREC QA 8 to 11 that ask for locations.

After each TREC, questions are made available with a file of answer patterns, e.g. regular expressions that match one or more acceptable answer to the question. TREC also publishes judgment files which consist of extractions submitted by systems. Each extraction is identified by the document id (from the TREC corpus), and comes with TREC assessment (correct versus incorrect extraction). These judgments also reflect how TREC QA has evolved since the first track in 1999. Systems were first allowed to output a string limited to either 50 or 250 bytes [Voorhees and Tice, 1999]. In 2002, systems were required to return an exact answer. The notion of exactness [Voorhees, 2002] was left to judges, but the main new constraint was that extractions containing the correct string could be discarded if they were also containing extraneous information.

All these files are useful for building and training a QA system but they reflect different tasks: Earlier extractions come with some context whereas recent judgments tend to be short. It thus happens that an answer judged correct in a previous track became incorrect for the same question in a recent track given the criterion of exactness.

Table 4.1 provides an example of this evolution.

Table 4.1: Evolution of answer correctness in TREC QA

What does CPR stand for? (TREC 9, 431 and TREC 11, 1516)
Despite the advent of cardiopulmonary resuscitation (CPR), 400,000 to 600,000 persons die every year in the United States from sudden cardiac arrest. (Correct answer TREC 9)
cardiopulmonary resuscitation (Correct answer TREC 11)

For the dissertation research supported here, I needed to homogenize TREC data on that level and extend exactness to earlier tracks (8, 9, 10) to set up a corpus representative of the current task (apart from the NIL case) and make data from earlier tracks usable. The main reason one wants to stick to the 'exact answer' rule is to distinguish properly between what is contextual information and what is the actual exact answer. Contextual information is an interesting research topic, but it must be formalised as it is in the corpus and not be mixed with the answer.

Correct answer phrases were extracted by matching TREC patterns over correct judgments. It appears sometimes that a pattern matches nothing or that a correct judgment is not matched by any pattern. I did not change anything since TREC files are standard files. Those cases have been skipped.

Incorrect answer phrases were extracted by selecting a random span in incorrect judgments. The span length is random, but limited to 1 to 4 tokens, including punctuation marks, which is corresponds to the span length interval of a correct answer. It is actually a difficult task to generate incorrect answers, i.e. segments that mimic a not random but wrong answer. I chose this method arbitrarily, and it appears to give fairly "exact" but incorrect phrases that could be used a supply of incorrect answer candidates that matched the exactness rule.

The main corpus is composed of all judgments, except those that have been skipped as said above and judgments that contain the NIL string (which means there is no answer). Some filtering had to be done as some judgment strings were obviously not extractions but made up by QA systems ("nullnullnull"). Table 4.2 shows a sample of this corpus for a question. The first column contains the identifier of the question, the second the identifier of the document from which the answer string was extracted, the third column is the TREC judgment and the last column the answer string itself.

Table 4.2: Sample of exact TREC judgments for *Who was Galileo?*

896	AP901231-0147	1	astronomer
896	FT924-10273	1	astronomer
896	FT933-6879	1	the Italian sunspots expert
896	AP890214-0137	-1	manager
896	AP891109-0117	-1	NASA's Jet
896	WSJ900730-0009	-1	payloads;Long Duration Exposure Facility
896	AP890925-0015	-1	heresy Pope
896	AP890925-0015	-1	the University of Pisa
896	AP890925-0015	-1	the

This corpus gives a fair overview of the kind of extractions that can be found in the TREC corpus and allows studies on answer redundancy, variability, and on the proportion of answers that are incorrect but related, alternative answers. It currently covers TREC 8 to 11 and contains 361,668 answers (correct and incorrect) to 1893 questions.

From this master corpus, for each question, I selected 5 incorrect phrases and 5 correct phrases (when possible) per question. In this subset, the size of the list of answers is limited to 10. It can be less if for instance only one answer had been found among all systems. Answer phrases are all different in terms of string matching. This sub-corpus has been built to study multiple answer sets (including incorrect answers) with no naive redundancy.

These two corpora are freely available on the web[1] and have been linked by TREC on their website in the QA resources section [2].

[1]http://www.iccs.informatics.ed.ac.uk/~s0239548/data.html

[2]http://trec.nist.gov/data/qa/add_qaresources.html

Besides judgment files, answer patterns are made available by Ken Litkowski and distributed by TREC after each evaluation. Those patterns are unofficial but serve as a basis for TREC training. Answer patterns are Perl regular expressions matching one or more answers, as shown previously in Table 2.1 for the question *What is epilepsy?*.

For this evaluation, I gathered a set of extractions meant to closely resemble those given by an actual system and used the corpus of exact TREC judgments described above. For each question, 5 incorrect judgments and 1 to 5 correct judgments were selected, depending on the number of correct judgments available. 44.7% of the questions had one correct answer, 17.6% had two, 12.9% had three, only 3.5% had four and 21.3%, five. While between 50% and 87% of the extractions associated with each question correspond to incorrect answers, nevertheless, as in Table 2.3, more than half of the questions had several correct answers.

Table 4.3: Sample of the evaluation material for fusion on location questions.

What is the capital of Persia?	What county is St. Paul, Minnesota in?
Tier II	mansions
East	Minn.
Tehran (correct)	Henan
Isfahan (correct)	Ventura
Judah	**Ramsey County** (correct)
Iranian Embassy	Minneapolis
rest	

The material also contains very little trivial redundancy. 82% of the questions do not have overlapped strings among answers and, when repetition occurs, the frequency of the repeated substring is never more than 2. Candidate frequency is thus a limited feature for re-ranking (especially as a few repeated strings are actually incorrect). An example of the input is given in Table 4.3.

This input is directly fed into the system. The output of each strategy (described in Section 4.4) is a ranked list of strings that is assessed against a list of TREC patterns (regular expressions matching correct answers).

4.2 System Architecture

The evaluated software, QAAM (QA Answer Model), is based on the MVC design pattern. Its architecture is based on two main components: (1) a graph model generated from unstructured data (question and answer candidates) and (2) a set of functions in charge of controlling and rendering, i.e. given a task or a user preference, a rendering function will be selected to retrieve appropriate information from the model and cast it into an adequate medium. As mentioned earlier, in the long run, it would be desirable to include a more dynamic interaction between views (rendering) and the model. For instance, user feedback would allow for the model to be enriched or updated according to further user feedback or requests (dialog QA). At the current stage, neither such interaction nor user feedback process is available, and, for this task, rendering is limited to a sorted list of node values.

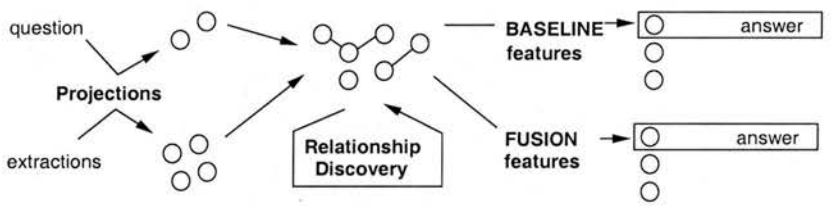


Figure 4.2: Architecture of QAAM for fusion on location questions

The task assesses the performance of answer comparison in terms of answer correctness. The system takes as input a question and a list of extractions, generates a graph and outputs an ordered list of strings, each corresponding to a different node. Each string is then evaluated against TREC answer patterns. Figure 4.2 presents the architecture of the current system.

Note that the two strategies to be assessed, the baseline and the fusion approach, derive from the *same* model. Only the retrieval of answers differs. Each view (one for the baseline, on for the fusion approach) uses a specific set of features, or characteristics of the graph model, to produce a ranked list of nodes. In the two next sections, I describe (1) how models are generated and (2) how the strategies differ from one another.

4.3 Automated Generation of Answer Models

QAAM takes as input a question and a collection of the corresponding extractions, all of which are assumed to have been tokenized and Part-of-Speech (POS) tagged (here POS-tagging using MXPOST [Ratnaparkhi, 1996].

Table 4.4: N-gram regular expressions over POS-tags and strings for node projection

The syntax used is of regular expressions:
$*$ matches any sequence of 0 or more characters.
$+$ matches any sequence of 1 or more characters.
? denotes that the previous character is optional.

Regular expressions	Examples
(NNP?S?)+	"Galileo Galilei", "city", "capital city"
(NNP?S?)+ of DT (NNP?S?)+	"Valley of the Kings"
*JJ (NNPS?)**	"east Asia"
(NN?PS?)+ of (NNP?S?)+	"strip of water", "Isle of Man"

POS symbols			
NNP	proper name (singular)	NNPS	proper name (plural)
NN	common name (singular)	NNS	common name (plural)
DT	determiner	JJ	adjective

What is projected can be a token, a word or a multi-word expression. In this experiment, nodes were projected from nominal phrases from the question and the answer extractions, by matching token and POS n-grams with regular expressions (Figure 4.4).

Most extractions in TREC QA are nominal phrases including numbers (see Section 3.1). Projecting nominal phrases (without numbers) was sufficient to cover the dataset.

Now let us take an example of a pair of question and judgments, as given in 4.5.

Table 4.5: Pair of question and original system judgments

Question	*What continent is Scotland in?* (TREC, 1647)
Extractions	*Europe*
	EDINBURGH
	Africa
	Ireland
	in Africa
	Scotland

After tokenization, POS-tagging and N-gram matching, QAAM projected the following nodes: *continent*, *Scotland* (from the question), and *Europe*, *EDINBURGH*, *Africa*, *Ireland*, *Africa*, *Scotland* from the extractions.

Nodes are represented as features (i.e. attribute-value pairs of linguistic annotation) so that annotation can be used during the comparison process. For instance the node Europe:

$$\begin{array}{rlcl} \text{Node}\, \{ & \text{token} & = & \text{Europe} \\ ; & \text{pos} & = & \text{NNP} \\ ; & \text{type} & = & \text{answer\_node} \\ \} & & & \end{array}$$

has a set of three features containing information on the token itself, its POS-tag (*pos*) and its source (*answer_node*, i.e. answer candidate, as opposed to a *question_node*).

From this set of nodes, a directed graph is built using different resources to discover relationships between nodes. Relations are used to label edges between them. I identified two relationships: equivalence and inclusion (see Section 3.2).

To identify *equivalence*, I used techniques based on string matching for acronym recognition and synonyms from WordNet [Miller et al., 1993]. An *inclusion* relationship is drawn between two nodes when (1) one of three pointers (`hypernymy/hyponymy`, `part of` or `member of`) exists between their corresponding entries in WordNet, or (2) they share the same lexical head(s) (by string matching, e.g. *Pacific* 'includes' *western Pacific*). Finally, the transitive closure is performed over the graph, according to the following rules, where X, Y and Z are nodes of the graph:

$$\text{equivalent}(X, Y) \wedge \text{equivalent}(Y, Z) \rightarrow \text{equivalent}(X,Z)$$
$$\text{equivalent}(X, Y) \wedge \text{includes}(Y, Z) \rightarrow \text{includes}(X, Z)$$
$$\text{includes}(X, Y) \wedge \text{includes}(Y, Z) \rightarrow \text{includes}(X, Z)$$

The answer model shown in Figure 4.3 has been generated by QAAM for the set of question and answer extractions given in Figure 4.5. (Notice transitive closure over inclusion has not been performed on this graph for clarity). This graph rendering is a view based on the translation of the model into the DOT language for graph specifications (Graphviz tools [Gansner and North, 1999]). In all figures, edges without arrows stand for *equivalence* links, the others for *inclusion*. A bold box indicates a question node, otherwise an answer candidate (e.g. Scotland appears twice, once in the question, once as a candidate.)

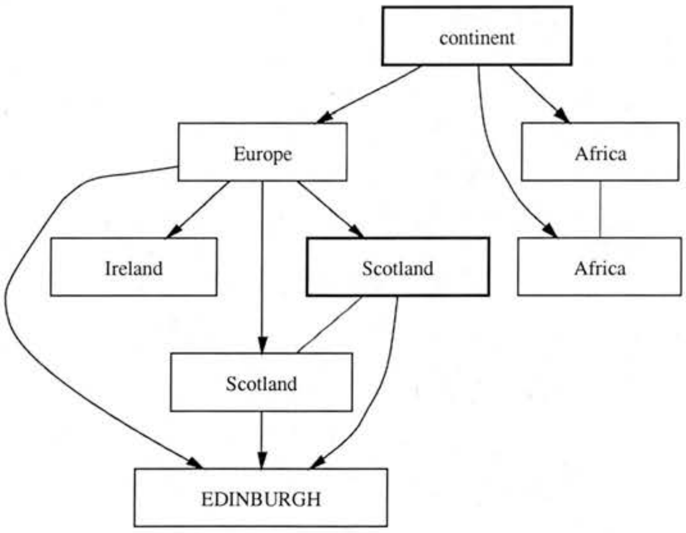


Figure 4.3: Graph model for *What continent is Scotland in?*

Two distinct functions were then used to sort nodes from the same model using different re-ranking features based on the graph's properties. These are described in the next section.

4.4 Re-Ranking Features and Rendering

Once a model is generated, distinct views can be produced to the end user. A function in charge of producing a view performs two sub-tasks:

1. Select the amount of information required for answering.

2. Cast the answer into the required medium.

In this experiment, all nodes except question nodes and their equivalents are considered potential candidates for answer content. The final output for each strategy is a ranked list of nodes retrieved from the graph model for the question. The rendering is the same for both the baseline and the fusion strategy: an ordered list of strings representing nodes of the model. However, the selection of information, i.e. the node ordering function, differs.

Figure 4.4 shows the answer model generated for the question *Where is Glasgow?*. Once such a model has been generated, several properties, based on the topology of the graph, are computed to compare nodes.

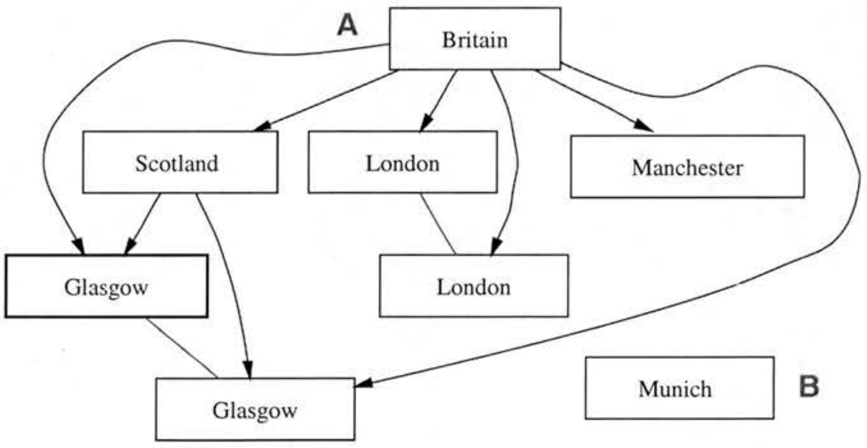


Figure 4.4: Graph model for *Where is Glasgow?*

The first characteristic to be noticed is that the graph has two distinct components: the connected subgraph with root node *Britain* (A) and a singleton (B). When I refer to the *partition* of a node, I mean the connected component the node is member of, e.g. *London* is in partition (A).

Another characteristic is the connectivity between an answer node and its neighbours: What answer nodes are related to a question node? Are they equivalent or do they entail or include the question node? How do they connect to other answer nodes?

So for each answer node, the following features were computed:

(a) Does the node derive from the question or is it equivalent to a question node?
This filtering feature excludes nodes that paraphrase the question[3]. In Figure 4.4, *Glasgow* has been found as a potential answer by a QA system but a graph edge indicates it is equivalent to a node from the question and thus it is eliminated from the list of candidate nodes.

(b) How many question nodes is the node directly related to?
This checks for the direct connectivity of each answer node to question nodes, whatever the relationship is. For instance, the node *Scotland* scores 1, but *Manchester* scores 0.

(c) How many question nodes are present in its partition of the graph?
While (b) measures the number of direct connections, this feature makes use of transitivity among nodes to check for indirect connections. The node *Scotland* still scores 1, but *Manchester* now scores 1 as well, since it is connected to *Glasgow* via *Britain*.

(d) How large is its partition?
This measures the size of the partition of the node being considered. It provides an indication of semantic coverage. For instance, in Figure 4.4, (B) is a singleton and stands apart from other candidates, while the partition (A) connects candidates related to *Britain*, and has a large coverage. By featuring each node with its partition size, we can use the fact that a node belongs to a dominant semantic group, or not, for re-ranking. *Britain*, *Scotland*, *London* and *Manchester* will all score 7, while *Munich* has a singleton score of 1.

[3]Only for questions such as *What do you call a newborn kangaroo?* should the answer be a paraphrase of (part of) the question.

(e) How many children does it have by transitive inclusion?

This feature gives a measure of specificity. The fewer are its children by inclusion, the more specific we take a node to be. For instance, *Britain* has more children by inclusion than *Scotland* and is thus considered less specific. This is a weighting to sort nodes that are members of the same inclusive branch: Specific information will be ranked higher, e.g. *Scotland* will be ranked higher than *Britain*. The intuition is that specific nodes are more informative and provide better *exact answers*, which is a rule imposed by TREC QA. (Although, as mentioned in Subsection 2.4.1, the notion of *exact answer* is subject to debate. The rule seems to address exact answer pin-pointing in terms of string length rather than semantic exactness. In the latter sense though, *Scotland*, for instance, is more exact than *Britain*.)

(f) How many nodes is it equivalent to?

This latter feature measures the redundancy of a node.

Note that all connections (inclusion and equivalence) were judged of the same importance. The fact that some edges had been inferred from string comparison, and other from WordNet lookups, was not used to distinguish between edges. (I experimented with the latter distinction to assess the value of WordNet inferred relationships, as reported in Chapter 6, Subsection 6.2.4.)

The experiment carried out compares two different methods of combining these features in order to choose one of the nodes as an answer to the original question.

- The **baseline** method sorts nodes based only on features that relate the question and a single answer – i.e. (a) and (b): If a node is a question node or equivalent to a question, it is discarded. If two nodes X and Y have passed this stage, X is ranked higher if it is connected to more question nodes than Y.

- **Fusion** makes use of all the features in the order: (a), (b), (c), (d), (e) and (f), and reflects relations between (i.e. fusion of) multiple nodes.

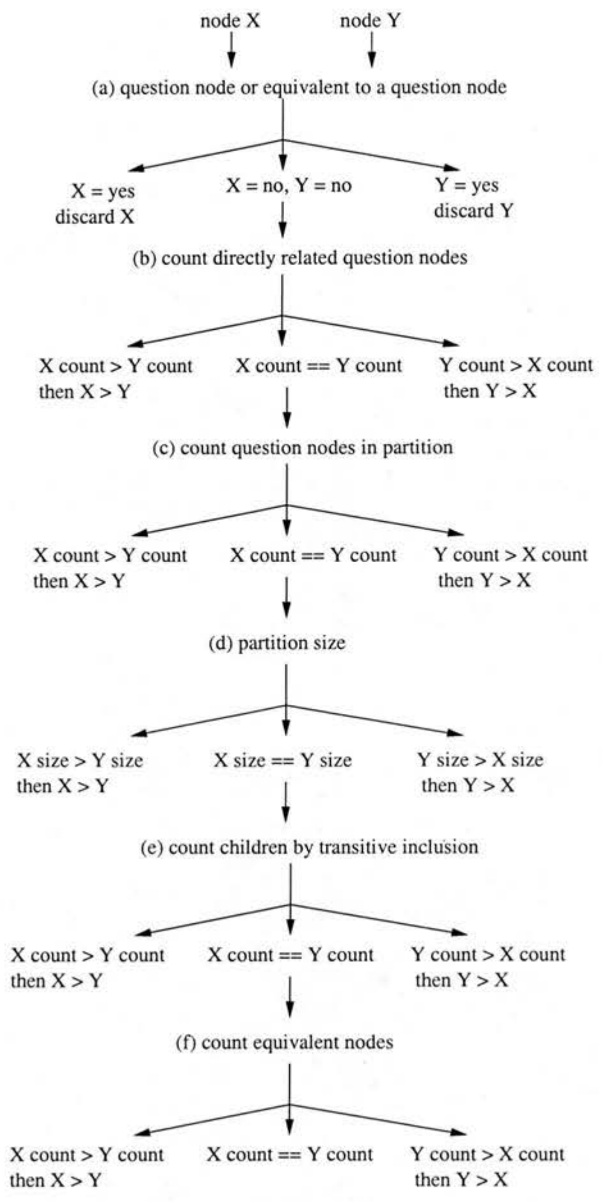

node X
node Y
(a) question node or equivalent to a question node
X = yes discard X
X = no, Y = no
Y = yes discard Y
(b) count directly related question nodes
X count > Y count then X > Y
X count == Y count
Y count > X count then Y > X
(c) count question nodes in partition
(d) partition size
X size > Y size then X > Y
X size == Y size
Y size > X size then Y > X
(e) count children by transitive inclusion
(f) count equivalent nodes

Figure 4.5: Decision tree for the fusion system, comparing two nodes for re-ranking.

The experiment allows us to quantify the contribution of answer candidates, be they correct, incorrect but related, or just incorrect.

Notice that features (c), (d), (e) and (f) make use of relations between any kind of nodes and thus cannot be used by the baseline. For instance, according to feature (c), *London* in Figure 4.4 is related to one question node, *Glasgow*, by following paths between answer nodes. This cannot be used for the baseline because it involves making use of transitive relationships discovered through answer comparison. If the path does not actually contain any answer node, it is then equivalent to feature (b), which is a baseline feature.

Finally, the node with the longest string was selected for tie-breaking as the task is to get the most accurate answer, which I heuristically associate with length. (Tie-breaking, however, is not used by the baseline because it involves answer comparison.) The original order of the input list is otherwise preserved.

Figure 4.6 shows the score of each node for the model given above. Glasgow is filtered out by both strategies because it scores positive to feature (a) (equivalence to a question node).

Table 4.6: Candidate scoring for the question *Where is Glasgow?*

	(a)	(b)	(c)	(d)	(e)	(f)
Britain	0	1	1	7	6	0
Scotland	0	1	1	7	2	0
Munich	0	0	0	1	0	0
Manchester	0	0	1	7	0	0
London	0	0	1	7	0	1
Glasgow	1	1	1	7	0	1

The baseline, using feature (b) (direct relationship with a question node) ranked nodes as follows: *Britain*, *Scotland*, *Munich*, *Manchester*, *London*. The fusion-based approach ranked them in the following order: *Scotland*, *Britain*, *London*, *Manchester*, *Munich*.

The re-ranking task, despite the fact that it gives an indication of the ability of fusion to provide additional features for sorting, does not evaluate the core advantage of fusion, which is to organize information, rather than filter it. For instance, in the Glasgow example, the baseline ranks correct answers at the top, as the fusion approach does. But re-ranking does not evaluate the fact it is known from fusion that *Scotland* and *Britain* are related. While information fusion could select several nodes with their relationships and generate an answer, for instance *Glasgow is in Scotland, Britain*, here I stick to a single node and the string it yields, as required by the re-ranking task and the current rules in TREC QA.

4.5 Results and Error Analysis

For the 85 questions, the top-ranked answer from the **baseline** was correct in 42 cases (49%), and the Mean Reciprocal Rank (MRR) over all 85 questions was 0.63. In contrast, the top-ranked answer from **fusion** was correct in 62 cases (72%), and the MRR over all questions was 0.82. (MRR evaluates overall ranking: $MRR = \sum_{q=1}^{N} \frac{1}{r_q}/N$, where r is the rank of the first correct answer for each question and N the number of questions). A two-sided Wilcoxon-Rank-test for pairwise ascertained samples over the first ranked answer shows that the improvement at rank 1 provided by fusion is significant ($p < 0.01$).

Fusion found the same 42 correct answers as the **baseline**, plus 20 others. The first answers provided by the **baseline** were incorrect in the following cases:

- In 9 cases, no relation could be drawn between a question node and an answer node. In these cases, the selection among answer nodes is random. In contrast, the strategy based on **fusion** was given a clue by connections between answer candidates and selected the most specific and redundant node in the largest partition. From the model shown in Figure 4.6, the **baseline** selected the incorrect answer *free-lance* as first answer, using **fusion**, the system proposed *Luxor*, the most specific node, and then *Egypt*. The inclusion was drawn from WordNet.

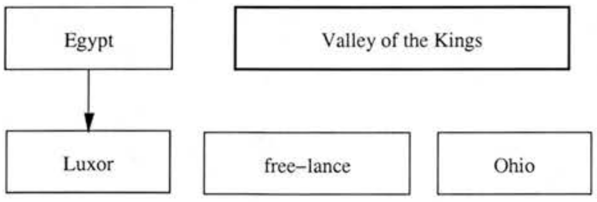


Figure 4.6: Graph model for *Where is the Valley of the Kings?*

- In 11 cases, there were relations between question nodes and incorrect answer nodes. In these cases, **fusion** was helped by comparing the specificity of each node (6 cases), the redundancy score (1 case) and the string length (4 cases) while the baseline's choice was again random. For instance, in Figure 4.4, the baseline chose *Britain*, which is not accepted as a correct answer in TREC. The fusion controller's choice was first *Scotland* and in second position *Britain*.

The experiment generated 85 answer models with an overall count of 841 nodes (16.64% from questions and 83.35% from extractions) and 785 relations. 59.6% of the relations occurred between answers only, 38.7% between a question and an answer node, while 1.7% were discovered among question nodes. There can be a few relations among the nodes projected from the question. For instance, the model draws an inclusion between the two nodes *Rome* and *Italy* projected from *What river runs through Rome, Italy?* (TREC, 1836). The proportion of relations among answers only is the largest one, which may give a clue why fusion performed better than the baseline.

Most of QAAM's relation discovery exploits the inclusion relation (on average 6 inclusion relations per model against 3 equivalence relations), probably a consequence of the kind of question focused on. Location questions often induce an inclusion or entailment relationship between a word of the question and the expected answer (*Where is X?* X is part of <*answer*> or <*answer*> 'includes' X). In addition, such spatial relationships are well covered in resources such as WordNet, the knowledge required for inference was readily available.

Although our data set is biased towards spatial inclusion, it is clear that it is worth exploiting other kinds of relations among answers. Besides, in the set of extractions being worked on, trivial string matching redundancy was expressly taken out, with the exception of a few overlap cases, involving incorrect answer strings. While previous studies on answer re-ranking were mainly based on the computation of answer frequencies [Brill et al., 2001, Clarke et al., 2001], the approach envisaged here could be adapted to smaller corpora wherein frequency counts are less meaningful.

Interestingly, the inclusion relation also involved incorrect nodes, especially if the question contains a common functional term like *capital* in *What is the capital of Ethiopia?* (TREC, 1161). In this case, *London* was linked to the question node projected from *capital*. Although incorrect with respect to the given question, it helped

fusion by enlarging the partition of the graph relating to capitals, which actually contains the correct answer.

Figure 4.7 describes the overall distribution of correct and incorrect answer nodes among relations.

The count of relations involving incorrect answer nodes is actually massive – 541, representing 68.9% of the total number of relations. The distribution of relations between correct answer nodes only is small (13.2%). On average there were only a few inclusion relations between them. It might be that incorrect but related answers actually helped. If we had an oracle indicating which answer nodes are correct or incorrect, so that the system only draws relations between question nodes and correct answer nodes, only 29.42% of the current relations would have been inferred, meaning that most relationships involved an incorrect answer node.

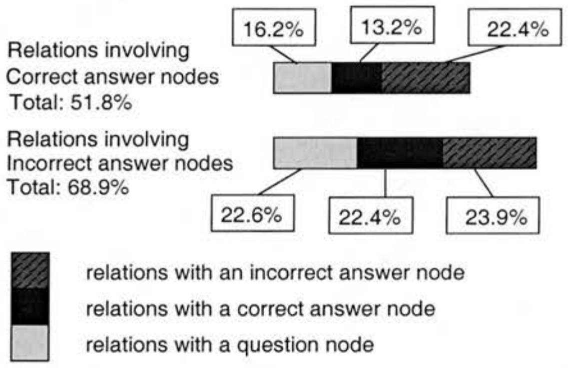


Figure 4.7: Distribution of relationships among correct versus incorrect answer nodes

To check the role of incorrect answer nodes, another experiment was carried out with such an oracle. Figure 4.8 is the graph generated by QAAM with an oracle for the answer model in Figure 4.3, generated without an oracle from the set of question and extractions given in Table 4.5.

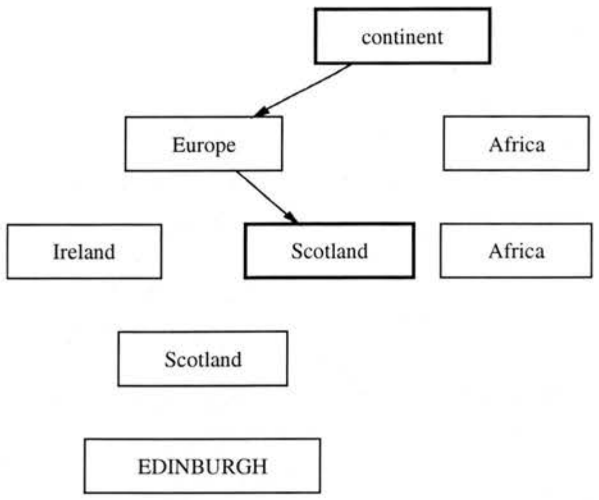


Figure 4.8: Oracle graph model for *What continent is Scotland in?*

The intuition behind this experiment was that if it could improve the baseline by blocking relations between question nodes and incorrect answer nodes, thereby ranking highly correct nodes with a relationship to the question, it could also lower **fusion** results by influencing features such as the partition size or the number of children by inclusion, which are influenced by relationships between correct and incorrect answer nodes.

Table 4.7: Comparative results for the baseline versus fusion-based re-ranking. Significance testing between baseline and fusion, and standard versus oracle systems, is marked + ($p < 0.01$) or - ($p > 0.01$).

	Baseline		Fusion		Significance
	1^{st}-rank score	MRR	1^{st}-rank score	MRR	
stand-alone	49%	0.63	72%	0.82	+
with an oracle	65%	0.71	78%	0.85	+
Significance	+		-		

Results shown in Table 4.7 show that the use of an oracle significantly improved the baseline ($p < 0.01$ at rank 1). However, results for **fusion** unexpectedly showed improvement as well, though not as large as for the baseline. This shows that a strategy that considers relations among both questions and answers is not only better but more robust and resistant to incorrect answers than a strategy that considers only relations between questions and answers alone.

4.6 Summary

On a restricted set of question types (location questions), answer comparison appears to improve significantly the task of answer re-ranking. While the relationships between question terms and answer candidates are relevant, relationships between candidates themselves, whether correct or incorrect, can help building a supportive network around a correct candidate. In previous work [Abney et al., 2000, Clarke et al., 2001, Brill et al., 2001], frequency, i.e. answer comparison on the basis of string equivalence, has been shown to improve re-ranking. In this corpus, for which answer frequency was low, inclusion based on overlap, as well as non trivial semantic inclusions and equivalences, were the most exploited sources of connectivity between answer candidates. The corpus itself shows that it is not rare for factoid questions to have multiple correct answers. While the QA community tends to focus on multiple answers in the context of definition or list questions, this experiment demonstrates that multiple answers can also be worked on for factoid questions.

There are nonetheless several limitations to this experiment. First, the questions were limited to location types. This may have impacted on the results because the expected relationship between a question node and a correct answer node, or between correct answer nodes, was likely to be of an inclusive type that is well-covered in WordNet. Ultimately, connectivity depends on the ability of a system to infer relationships between nodes of a graph model. There is no guarantee that, using another source of knowledge, or working on different types of questions, the decision tree elaborated for the fusion system will generalize improvements. A second bias, whose impact is more difficult to assess, is the fact that the candidates for this corpus were collected from judgments provided by existing QA systems, and thus they reflected the pre-processing performed by these different systems. It is not clear to what extent this may have biased the experiment.

In the following chapters, I describe experiments using web answer candidates collected directly after the information retrieval stage, which, for each question, consisted in querying the web search engine Google with the question string (Chapter 5). The material is thus only biased by the processing algorithm of the search engine. In Chapter 6, I also extend the comparison between a baseline and a fusion system to new question types, including definition, time, personal-related and other question type. Finally, while the experiment described here, was focused on re-ranking single candidates, the next experiments in Chapter 6 demonstrate that fusion can also be used to provide more flexibility in answering, by proving clusters of related answers rather than a list of single answers.

Chapter 5

A Web Corpus of Multiple Answers

From the experiment on location questions, information fusion proved to significantly improve a re-ranking baseline, by computing features based on the network, or graph, of answer candidates fused together, hence seeing candidates as potential allies rather than competitors.

They were, however, several limitations to this experiment. First, only location questions were considered, and the input was rather small (10 candidates per question). Besides, this input consisted of judgments from real QA systems, and thus was the result of a first pre-processing, and it is not clear what impact this could have had on the results.

Also, the research questions of this thesis involves assessing not only the qualitative improvement offered by fusion for QA, but also the value of generating a rich, structured, representation of the question and the answer candidates. While the experiment on location questions demonstrated the value of fusion for re-ranking, a more complete characterization of information fusion requires further investigation.

The primary research question addressed in this and the following chapters is the value fusion for handling multiple answers. In TREC QA, only definition, list of "other" questions are assessed with scores measuring recall and precision over multiple answers. For factoid questions, systems are required to provide a single answer only, and what is measured is the percentage of questions correctly answered with a single string (question recall at rank 1). While there are motivations for offering QA systems mining for single answers, this thesis is concerned with systems that are able to handle, not so much complex questions, but rather multiple answers, if not complex answers. To evaluate such systems, it is necessary to be able to systematically assess answer recall (the percentage of answers identified), as well as question recall. Also, in order to characterize fusion as means to provide more structured and more flexible answers, for instance by providing clusters of related answers rather a flat list of single answers, new measures are required to assess the exactness and precision of such clusters.

In order to perform this kind of evaluation, I needed a corpus exhaustively annotated with multiple answers. In this chapter, I describe a new corpus based on TREC QA questions for which web answers have been collected and annotated. I opted for a web-based corpus with the assumption that I would obtain a greater variety of answers than on the TREC QA AQUAINT corpus. I also used a test set with a greater variety of question types, including, for instance, definition questions.

The first section of this chapter describes the process of data acquisition and the manual rating performed to allow for an automated evaluation of answer candidates. I review the scoring measures to be used to assess non only single answer candidates, but also clusters of answer candidates, in order to evaluate answers that have been merged during the process of fusion. Finally, the second section provides a characterization of this new corpus, for questions as well as answer candidates.

5.1 Evaluation Material Acquisition and Methods

The evaluation material used in this chapter consists of a selection of TREC QA questions, and text content retrieved from a web search engine. I used the whole question string as query to retrieve the results provided by the first screen page of Google. Web snippets were then "projected" (tokenized and parsed into keywords and phrases) into answer candidates using QAAM. The whole set of candidate answers of each question was then manually evaluated.

The following subsection describes the process of question selection and answer candidate acquisition. I then review the evaluation procedure I used to assess answer candidates, and describe in depth the characteristics of this dataset. The final subsection presents the scoring measures used for the experiments described in this chapter, and assesses the value of Google as an IR tool for QA.

5.1.1 Data Acquisition

While on-going research shows more interest in "complex" questions, such as list or definition question, little attention has been paid to multiple answers to factoids. Questions from the TREC QA tracks 10 and 11 were thus selected so that a large part of the corpus would represent a large variety of factoid questions (64%, N=32) as well as a fair distribution of definition questions (36%, N=18). The test set consisted of a total of 50 questions, and 10 other questions were additionally selected for development purposes. A more complete overview of question types is given in Section 5.2.

Answer data collection consisted in querying the web search engine Google with each question string (without any further processing). For each question, the first screen page, equivalent to 10 web snippets, was retrieved and stored for further processing[1].

Each web snippet was then tokenized, POS-tagged and chunked using OpenNLP (v1.3) tools[2] [Baldridge et al., 2001]. Answer candidates were extracted using the projection technique described in further details in the next chapter, Subsection 6.1.2, i.e. only nominal keywords and phrases were extracted.

For each question, the 10 snippets and their respective list of extracted answer candidates were then manually evaluated.

5.1.2 Answer Rating

The particularity of the answer evaluation is that the assessor was not requested to evaluate the *correctness* of the answer, i.e. some universal truth value attached to the answer, but rather the level of *exactness* of the answer candidate and the *quality and strength of the evidence* supporting the answer candidate.

[1]Snippets were collected during January 2007.

[2]http://opennlp.sourceforge.net - OpenNLP is a set of parsing tools using maximum entropy models.

5.1.2.1 Evidence Rating

Evidence is the textual basis for belief or disbelief. In this evaluation, the source snippet serves as evidence. It may be seen as a textual justification that would argue in favour of the textual content of the snippet as an actual answer to the question.

Assessors may have their own set of beliefs about what does and does not constitute a "good" answer, independently of the source of snippet. However, the goal of the evaluation is not to assess the snippets and the answer candidates with respect to such beliefs, but with respect to the evidence provided "as is" by the snippet. Hence the judgment is not on the correctness of the snippet content but on the quality of the evidence. Agreement may vary as to what degree of explicitness is required to admit that the evidence indeed supports an answer, but the judgment should not be made on the basis of one's beliefs about the answer itself. Thus, this evaluation allows positive ratings for beliefs or opinions that may be false, incorrect or inexact, but for which the textual evidence is strong.

For instance a snippet containing the following string *tungsten has the highest melting point* (in response to the question *What metal has the highest melting point?*) is explicit enough to strongly support the answer candidate *tungsten* as a satisfying answer. So is the string *niobium has the highest melting point* to support *niobium*, although one may know that this belief about niobium is mistaken. Similarly, a snippet that only describes tungsten without specifying it has indeed the highest melting point does not provide any strong basis for the belief that tungsten is the answer to the question. The rationale for a rating of evidence quality rather than answer correctness *per se* is that automated question answering processes are going to be evaluated on their mining quality rather their ability to decide whether a statement is universally true or false.

Evidence is evaluated for each snippet, as a whole, independently from the list of extracted answer candidates, and also for each answer candidate in relationship to its source snippet. The two evidence-based ratings are further detailed in the next section.

5.1.2.2 Exactness Rating

Exactness is a rating that measures the quality of the automated process that extracts answer candidates from snippets. As such, it only applies to answer candidates, not to snippets.

This notion of exactness resembles the notion of exactness described in the TREC QA task [Voorhees, 2002]. For instance, in TREC QA, the answers *"Mississippi"*, *"mississippi"*, *"the Mississippi River"* are all considered correct and exact answers to the question *What is the longest river in the United States?*, while neither *"At 2,348 miles the Mississippi River is the longest river in the US"*, nor *"2,348 miles; Mississippi"* is considered exact. The requirement assesses the quality of pin-pointing the precise span that corresponds to an answer. It is meant to avoid (1) extraneous, possibly redundant or irrelevant, information, be it punctuation or words, or (2) the opposite, which is not given as an example in TREC QA, i.e. cases when the extracted span is not complete, for instance *2,348* is not exact because the unit is missing.

This is the first aspect of exactness rating, and it is essentially a string-based criterion for which it is possible to set specific arbitrary rules specifically defining what an exact string is. (Guidelines for exactness can be found in Appendix B.)

The second aspect evaluated under exactness is the level of specificity of a given answer candidate. For instance, the candidates *world* or *Earth* may not be considered informative enough to answer the question *Where is the Taj Mahal?* in a satisfying manner. Exactness here is taken in the sense of precise degree of informativity. Such exactness may vary depending on the question. For instance, *Mars* is an exact enough answer to the question *Where is the volcano Olympus Mons located?*.

The next subsection describes the rating made on the basis of these two measures, evidence and exactness.

5.1.2.3 Snippet and Answer Ratings

The rating task consisted in performing two manual assessments:

1. The evaluation of each snippet found for a given question.

2. The evaluation of answer candidates found for each snippet.

The objective of the first, snippet-level evaluation is to assess the value of the snippet as **evidence**, independently from the answer candidates that have automatically extracted from it. Because the extraction process sometimes misses out relevant candidates, this first assessment provides an indication of the value of the raw material (unprocessed snippets).

Table 5.1: Ratings for snippets and answer candidates

What are Quaaludes?	
Rating	**Snippet/comment**
Strong	*interview - quaaludes ... Such is the mystery of methaqualone, the downer that stamped its mark on the 1970s under its altogether more sexy American trade name: Quaaludes. ...*
	The snippet explicitly refers to Quaaludes as a trade name for the substance known as methaqualone. It also indicates its depressant quality. The evidence can be rated as strong.
Weak	*methaqualone: Definition and Much More from Answers.com ... On cult television show Strangers With Candy, the main character Jerri Blank talks about 'the good old days' and how no one makes good quaaludes anymore. ...*
	One may infer that methaqualone is the term defined in the document and that the excerpt mentioning Quaaludes allows to associate both terms. The assessor may also know, from self-knowledge or other snippets that indeed the word Quaaludes is a name for methaqualone. However, because this requires making an assumption and involves some inference, the evidence is only rated as weak.
None	*Ask Erowid : ID 143 : What are the effects of quaaludes? ... Ask Erowid Question and Answer: What are the effects of quaaludes?*
	The snippet does not contain any information relevant to answering the given question. It is thus marked as providing no evidence.

The judgment consists in assessing whether the snippet provides a well-supported answer to the question, i.e. if it represents good quality evidence. Three ratings are possible: *no evidence*, *weak evidence* or *strong evidence*. Table 5.1 provides a commented example distinguishing between these three ratings.

The second evaluation aims at assessing the answer candidates automatically extracted from each snippet. The assessor is requested to evaluate two characteristics of each answer candidate in the context of its source snippet: its **exactness** as a string, and whether the candidate is well supported by the source snippet, i.e. a measure of the **evidence** for the candidate in relationship to its source snippet.

Answer candidates can thus be rated as follows:

- no answer (irrelevant, off-topic)
- inexact answer - weak evidence
- inexact answer - strong evidence
- exact answer - weak evidence
- exact answer - strong evidence

It is important to note that the evidence rating for answer candidate is relational. It evaluates the quality of the snippet as evidence *for* a given answer candidate.

Indeed, a snippet could actually contain several relevant answer candidates, but provide strong evidence for only one candidate. In other words, the textual content of the snippet may provide a good justification for one candidate, but a rather poor one for another candidate. Thus, the evidence rating may be different from one candidate to another, although both have been extracted from the same snippet. The candidate evidence rating may also be different from the snippet evidence rating, i.e. a rating for the value of the snippet on its own, independently from candidate extraction.

For instance, the following snippet found to the question *What are Quaaludes?*

> *Methaqualone - Wikipedia, the free encyclopedia ... In the sitcom The King of Queens, after doing a favor for Doug, Supervisor O'Boyle mentions, "I wouldn't mind if a handful of Mexican quaaludes found their ...*

can be rated as strong evidence because it explicitly indicates that Quaaludes can be of Mexican origin, and this is a relevant piece of information to answer the question.

The snippet also indicates methaqualone, however it is not explicit that Quaaludes are another name for the substance. Thus, among the following sample list of candidates: *Methaqualone* is assessed as an exact answer, based on weak evidence, *Mexican* is rated as an exact answer, based on strong evidence, and *sitcom* for instance would be marked as irrelevant. An extraction such as *a handful of Mexican quaaludes* would be marked as inexact, based on strong evidence. Note that the exactness rating of an answer candidate does not vary from one snippet to another, whereas the evidence rating may. Exactness is rated with respect to the question, evidence with respect to the question and some context, i.e. the snippet.

To summarize, for both snippets and answer candidates, the value of the **evidence** can be rated as *strong* (good evidence) or *weak* (some evidence), or eventually set to *none* if the snippet does not support the answer. **Exactness** is not measured for snippets since they are evaluated as a whole, independently from the extraction process. The **exactness** of answer candidates only applies to those candidates that are supported by weak or strong evidence. The string is judged either *exact* or *inexact*. Table 5.2 summarizes the ratings for snippets and answer candidates.

Table 5.2: Ratings for snippets and answer candidates

	Evidence			**Exactness**	
Snippet	None	Weak	Strong	NA	
Answer candidate	None	Weak	Strong	Exact	Inexact

Appendix B provides a more detailed documentation of the guidelines for evidence and exactness ratings, as well as case-based examples of ratings.

Note that a limitation of the current manual evaluation is that, given time limitations, I was the only assessor. I strictly followed the guidelines given in the appendix, but additional judgments and inter-agreement scores will be indeed required to validate human assessment. Thus, this evaluation must be seen as a prototype evaluation.

5.1.3 Scoring Measures

As shown in the previous section, the evaluation of answer candidates is centered on two core ratings: the quality of the evidence supporting an answer candidate, and the system's ability to identify an exact answer span (i.e. that is well pin-pointed and informative enough). Each rating is assigned to an answer candidate (a string) within its source snippet.

For an automated evaluation, a QA system is presented with a question, the list of 10 snippets and their respective answer candidates. Different types of outputs can be assessed automatically.

I will distinguish here between the evaluation of (1) a ranked list of single answer candidates, and (2) a ranked list of clustered answer candidates.

A QA system can provide a ranked list of individual answer candidates, each of them associated with its source snippet, e.g. *methaqualone* from snippet 6. In this case, the evaluation consists of checking the ratings of the answer candidate for the given snippet, such that the evidence score of a candidate c from a snippet s_c takes the following values:

$$\begin{aligned} evidence(c, s_c) &= 0\% && \text{if the evidence was judged as null.} \\ &= 50\% && \text{if the evidence was judged weak.} \\ &= 100\% && \text{if the evidence was judged strong.} \end{aligned}$$

The exactness score of an answer candidate c is computed as follows:

$$\begin{aligned} exactness(c) &= 0\% && \text{if the candidate was not supported by any evidence.} \\ &= 50\% && \text{if the candidate was judged inexact.} \\ &= 100\% && \text{if the candidate was judged exact.} \end{aligned}$$

This is the most straightforward evaluation, and a similar one is used in the TREC QA track, for which answer strings as well as supporting documents are checked out.

However, when a QA system is based on fusion, the output list is rarely made of candidate-snippets or candidates-documents pairs. Rather each list entry consists itself of a set of answer candidates coming from different snippets merged together during the fusion process.

To assess such clusters of answer candidates, I propose to compute a mean evidence score and a mean exactness score. For each cluster of answer candidates C, where N is the size of the cluster, and c is an answer candidate found in snippet s_c, and member of cluster C, evidence and exactness scores are computed as follows:

$$cluster\_evidence(C) = \sum_i^N evidence(c_i, s_{c_i})/N$$

$$cluster\_exactness(C) = \sum_i^N exactness(c_i)/N$$

For the specifics of the experiments described in this chapter, I propose a lenient evidence-based score that selects the best evidence score for a given answer candidate c occurring in different snippets:

$$lenient\_evidence(c) \quad = \quad max(evidence(c, s_c))$$

where s is a snippet member of S, the set of snippets containing the answer candidate string c. For instance, the answer candidate string *methaqualone* may be found in three different snippets and be rated as follows:

$$\begin{array}{lrl} evidence(methaqualone, snippet1) = & 50 & \% \text{ (weak)} \\ evidence(methaqualone, snippet2) = & 100 & \% \text{ (strong)} \\ evidence(methaqualone, snippet3) = & 100 & \% \text{ (strong)} \end{array}$$

Methaqualone is well supported in two instances, and poorly supported in one instance. The lenient evidence score for the answer candidate string *methaqualone* is 100%. This lenient evidence score is meant to assess whether a given answer candidate is indeed well supported or not in the material. The standard evidence score is more oriented towards assessing the ability of a system to identify what is the best evidence for a given candidate. While it would be worthwhile in the future to have systems capable of assessing why a piece of text provides a better justification than another, this is beyond the scope of this thesis.The lenient evidence score provides a simpler indicator that the answer candidate has been to found to be well-supported.

The lenient evidence score also applies to clusters. The lenient evidence score of a cluster C is defined as the mean lenient evidence score of its members:

$$lenient\_cluster\_evidence(C) = \sum_i^N lenient\_evidence(c_i)/N$$

The main advantage of an exhaustive manual evaluation of answer candidates is the possibility to compute a complete truth table for each question, and assess the ability of the system to identify multiple answers that have been judged **satisfying**.

Table 5.3: Statistical measures computed from Gold standard judgments (satisfying vs not satisfying) over a system's list of answer candidates (detected vs not detected – meaning not in the list).

System/Judgment	Satisfying	Not satisfying
Detected	True Positive (TP)	False Positive (FP)
Not detected	False negative (FN)	True Negative (TN)

Sensitivity = Recall = TP / (TP+FN)
Specificity = TN / (FP+TN)
Positive Predictive Value = Precision = TP / (TP+FP)
Negative Predictive Value = TN / (FN+TN)

Accuracy = (TP+TN) / (TP+FP+FN+TN)
F-measure = (2 * Precision * Recall) / (Precision + Recall)

I distinguish between three groups of **satisfying** answer candidates: (1) the EXACT-STRONG group (exact answer candidates, supported by at least one strong piece of evidence), (2) the EXACT group (exact answer candidates, supported by strong or weak evidence), and (3) the LENIENT group (exact and inexact answer candidates, supported by strong or weak evidence). The membership of an answer candidate c to one of the group is based on the lenient evidence score and the exactness score:

$c \in$ EXACT-STRONG iff $lenient\_evidence(c) = 100\%$ and $exactness(c) = 100\%$
$c \in$ EXACT iff $lenient\_evidence(c) \geq 50\%$ and $exactness(c) = 100\%$
$c \in$ LENIENT iff $lenient\_evidence(c) \geq 50\%$ and $exactness(c) \geq 50\%$

An answer candidate c can thus be considered **satisfying** on a given level, if it belongs to the corresponding group (EXACT-STRONG, EXACT or LENIENT), e.g.:

$$\begin{aligned} satisfaction(group, c) &= 1 \text{ if } c \in group. \\ &= 0 \text{ otherwise.} \end{aligned}$$

Table 5.3 summarizes the core measurements computed from the truth table over answer candidates. Note again that an answer candidate is judged satisfying according to the groups defined above. Thus, per question, three truth tables are actually computed: one for satisfaction defined as EXACT-STRONG, one for the EXACT group and one for the LENIENT group. Also, the truth table does not assess clustering, but the performance of a system at finding multiple satisfying answers, whether those answers were clustered or not.

Given the satisfaction function, it is also possible to compute a precision score for a cluster, indicating the number of satisfying answer candidates over the size N of cluster C, e.g.:

$$cluster\_precision(group, C) \quad = \sum_i^N satisfaction(group, c_i)/N$$

Two measures can then be computed to assess the quality of the clustering process. Given a *list* of clusters, **clustering precision** is defined as the mean cluster precision score of each cluster C member of the $list$, over the size N of the list.

$$clustering\_precision(group, list) \quad = \sum_i^N cluster\_precision(group, C_i)/N$$

The **positive clustering precision** is the computation of the clustering precision over satisfying clusters only, i.e. the mean cluster precision score of each cluster C member of the $list$, such as C contains at least one satisfying answer candidate. M is the number of satisfying clusters in the $list$.

$$\begin{aligned} &positive\_clustering\_precision(group, list) \quad = \sum_i^M cluster\_precision(group, C_i)/M \\ &\forall\, C_i \; cluster\_precision(group, C_i) > 0\% \end{aligned}$$

Both scores provide an evidence-based and exactness rating for clustering, as defined by the group level (EXACT-STRONG, EXACT, LENIENT). The first measure (clustering precision) indicates the overall quality of the clusters in terms of mean answer recall. The second measure (positive clustering precision) focuses more specifically on the quality of satisfying clusters, as opposed to irrelevant clusters.

To focus on exactness more specifically, it also possible to compute a **clustering exactness score**:

$$clustering\_exactness(list) \quad = \sum_i^N cluster\_exactness(C_i)/N$$

as well as a **positive clustering exactness score**:

$$positive\_clustering\_exactness(list) \quad = \sum_i^M cluster\_exactness(C_i)/M$$
$$\forall\, C_i\ cluster\_exactness(C_i) > 0\%$$

Finally, the following measures assess the ranking strategy of a system, and can be computed for each group of satisfying answers (EXACT-STRONG, EXACT and LENIENT).

The **Mean Reciprocal Rank (MRR)** measures the answer candidate ranking strategy of a system, and is defined as: MRR = $\sum_q^N \frac{1}{r_q}/N$ where r is the rank of the first satisfying answer candidate for each question q and N the total number of questions.

In fusion-based evaluations, the output ranked list provides sorted clusters rather than individual answer candidates. The rank r is thus the the rank of the first satisfying cluster, i.e. a cluster that contains at least one satisfying answer candidate. Of course, if a system produces rather larges clusters, e.g. a cluster at rank 1 containing 10 candidates, its MRR may be overall better than a system that produces very small clusters, e.g. a cluster at rank 1 containing 1 or 2 candidates only. The MRR is a limited measurement when it comes to assessing clusters, and it must be balanced with measurements indicating the exactness and evidence scores for clusters (clustering precision). Those scores both penalize clusters that contain too many inexact or not well-supported answer candidates.

Another limitation of the MRR is that it only takes into account the first satisfying answer, and thus does not evaluate the ranking strategy for multiple answers.

The **question recall at rank** is a function $qrr(r)$ that indicates the percentage of questions that have been satisfyingly answered at rank r. As the MRR, the question recall at rank over clusters needs to be balanced with a measure of the clustering quality.

The **answer recall at rank** is a function $arr(r)$ that indicates the percentage of satisfying answers that have been found at rank r. The representation of this function indicates how good is a system at ranking multiple answers.

Table 5.4: Evaluation measures for clustered answer candidates.

Measure	**Definition**
Sensitivity = Recall	Number of satisfying answers found by the system over the existing number of satisfying answers.
Specificity	Number of non-satisfying answers correctly discarded by the system over the existing number of non-satisfying answers.
Positive Predictive Value = Precision	Number of satisfying answers found by the system over the total number of answers returned by the system.
Negative Predictive Value	Number of non-satisfying answers correctly discarded by the system over the total number of answers discarded by the system.
Accuracy	Performance of a system at identifying satisfying answers AND at discarding non-satisfying answers.
F-measure	Weighted harmonic mean of precision and recall.
MRR	Cluster re-ranking score.
Question Recall at Rank	Distribution of questions satisfyingly answered at a given rank.
Answer Recall at Rank	Distribution of satisfying answers found at a given rank.
Clustering Precision	Mean precision of clusters.
Positive Clustering Precision	Mean precision of satisfying clusters.
Clustering Exactness	Mean exactness of clusters.
Positive Clustering Exactness	Mean exactness of satisfying clusters.

As the previous function, the answer recall at rank over clusters has to be considered with an evaluation of clustering as well.

Table 5.4 summarizes the measures computed in the following experiments for a system that outputs clusters of answer candidates. Sensitivity, specificity, precision, negative predictive value, accuracy and F-measure evaluate the ability of a system to identify satisfying answers and discard irrelevant ones. Those measures do not take into account clustering, and are computed over all the answer candidates returned by the system. The MRR, question recall at rank and answer recall at rank evaluate the cluster-based ranking strategy. As mentioned above, they must be considered with respect to clustering quality, provided by clustering precision and positive clustering precision.

Note again that the measures are computed for each group defining a level of satisfaction expressing requirements in terms of evidence quality and exactness (EXACT-STRONG, STRONG and LENIENT satisfaction groups).

5.2 Data Characterization

This section describes the test set of 50 questions selected from the TREC QA corpus (tracks 10 and 11), as well as the answering material (snippets and extracted answer candidates) retrieved from Google. I first describe the distribution of question types, and then characterize the quality of the answering material, based on the rating performed during the manual evaluation.

5.2.1 Question types

Each question was manually tagged with a question type. Table 5.5 details the distribution of question types over the test material.(The training material, which consists of 10 questions, reflects a similar distribution.)

Table 5.5: Distribution of question types in the test material.

Type	#	%	**Example**
FACT-LOC	13	26	*What is the fourth highest mountain in the world?*
FACT-PERS	9	18	*Who invented the instant Polaroid camera?*
FACT-TIME	5	10	*When was the first liver transplant?*
FACT	5	10	*What do bats eat?*
Total factoids	32	64	
DEF	15	30	*What is autism?*
DEF-BIO	3	6	*Who was Galileo?*
Total definition	18	36	
Total	50	100	

The two main question types were the factoid type (64%, N=32) and the definition type (36%, N=18). Among factoids, I distinguished between:

- location questions (FACT-LOC) requesting a place name.
- person questions (FACT-PERS) requesting a person name.
- time questions (FACT-TIME) requesting a time expression.
- other, unclassified, factoids (FACT), requesting a named entity (e.g. currency) or a common name (e.g. a color).

Among definition questions, I distinguished between:

- questions requesting actual definitions (DEF).
- biography questions (DEF-BIO) requesting information about a person.

5.2.2 Characterization of Snippets

For each question, the top 10 Google snippets were retrieved and assessed in terms of strong or weak evidence (Subsection 5.1.2). Table 5.6 shows the results of this manual assessment for snippets.

Table 5.6: Assessment of snippets as evidence.

Snippets	**No evidence**	**Weak evidence**	**Strong evidence**	**Total**
Total #	170	42	288	500
% (N=500)	34	8.4	57.6	100
Average # per question	3.40	0.84	5.76	10

All questions were answerable within the 10 first Google snippets. In comparison, Saggion et al. [Saggion et al., 2004] report the following question coverage measures at rank 10 for search engines performing passage retrieval: between 48.1% and 60.5% for Okapi, and between 40.1% and 51.7% for Lucene (depend on the strict and lenient evaluation of answers). In this evaluation set, the coverage is 100%, leading also to a much higher answer redundancy: On average, more than half of the snippets contain a satisfying answer. Saggion et al. report, at rank 10, an answer redundancy between 0.80% and 1.04% for Okapi, and 0.68% and 1.42% for Lucene. The test corpus used here is definitely much denser in terms of answers. A few reasons can explain this redundancy:

1. The set of questions worked on here is small and it may simply have been a feature of this particular set of questions to generate more answer redundancy.

2. Web data may generate more multiple answers, in terms of both string frequency and also distinct answers.

3. The pool of TREC answers to factoids is very likely to be below realistic average, hence reducing answer redundancy in TREC data for factoids.

4. The assessment for the corpus I describe accepted opinions and beliefs as long as they were supported by textual evidence. This may have increased the amount of multiple answers.

Saggion et al. report measurements on a much larger set of questions (500 questions from TREC 12), and compute measures from passages retrieved from the AQUAINT corpus. The questions selected for this Google-based corpus were perhaps more easily and more densely answered with a web search engine.

Another related hypothesis explaining answer redundancy from Google snippets would also be that, because the web is a much larger corpus than AQUAINT, and also more heterogeneous, it potentially leads to not only more frequent answers, but also a larger distribution of distinct, multiple answers. (This will be reviewed in the next subsection concerning answer candidates.)

Table 5.7: Strong evidence per question type

Type	**# "strong" snippets**	**# snippets**	**%**
DEF	104	150	69.3
FACT-PERS	58	90	64.4
FACT	28	50	56
FACT-LOC	68	130	52.3
DEF-BIO	13	30	43.3
FACT-TIME	17	50	34
Total	288	500	57.6

The second hypothesis above is related to the fact that, from TREC QA 11 on, systems were required to provide only one answer to factoid questions. The evaluation of answers in TREC QA is not exhaustive for the obvious reason that such a manual assessment over the AQUAINT corpus (about 3.5 GB) would be extremely time-consuming. Thus answers candidates assessed by judges are coming from the pool of answers provided by all the evaluated systems. Since the one answer per factoid rule, the pool of candidates for factoids has been extremely reduced, going from 44.4% of questions having multiple answers in TREC 10, to 13.2% in TREC 11 (Table 2.2).

Table 5.7 shows the distribution of snippets judged as strong evidence per question type. Standard definition questions, as opposed to biography questions, are the ones with important answer redundancy: 69.3% of all snippets for standard definition questions were assessed as strong. The lowest answer redundancy score, 34% for time questions, is still much higher than the average 1% reported on the AQUAINT corpus for a larger set of questions. The percentage of "strong" snippets for all the factoid questions in this Google corpus is 53.4% (171 "strong" snippets over 320).

It is an hypothesis that the high answer redundancy of this corpus is partly due to the fact that answer candidates were assessed in an exhaustive manner, while, on the other hand, the pool of assessed answers to factoids in TREC 12 was below a realistic average because of the new one answer per factoid rule. In the following subsection, I will show that, indeed, the amount of distinct, multiple answers is much higher for the web data examined here than for TREC QA data.

Another reason perhaps for answer redundancy is the type of evaluation performed. The manual assessment of answers on this corpus was evidence-based, i.e. an answer was judged satisfying if it was well supported by a snippet, not if it was a "correct" or "incorrect" answer to the question, in absolute terms. Thus opinions that were not necessarily "correct" were accepted, although such occurrences were rather infrequent: only three questions had answers expressing perhaps unusual beliefs. For example, to the question *What does the word fortnight mean?*, one snippet stated very explicitly that the term was sometimes understood as *ten days*, when *fortnight* actually refers to a period of 14 days. It is possible that such an assessment generated a larger number of satisfying answers.

Finally, the MRR for snippets, i.e. based on the reciprocal rank of the first snippet containing a satisfying answer, is 0.83 for snippets with strong evidence, and 0.87 for snippets with strong or weak evidence. This score is high. However, it does not correlate with a good MRR based on answer candidates listed in the snippet order, as I will show during the experiments described in this chapter.

The next subsection is focused on the characterization of answer candidates, i.e.

the keyword terms or phrases that were automatically extracted from snippets.

5.2.3 Characterization of Answer Candidates

Answer candidates were extracted automatically from snippets (see Subsection 6.1.2 of the next chapter for further details). This process, because automated, sometimes missed out relevant candidates. In the subsection above, I reported 330 relevant snippets (strong and weak evidence). During the extraction process, 6 of them did not actually generate any answer candidate at all.

Table 5.8 summarizes annotation information for answer candidates, which was based on two criteria: exactness and the quality of evidence for the given extraction in relationship to its source snippet (Subsection 5.1.2).

Table 5.8: Exactness and evidence-based annotation of answer candidates.

N-A: Not an Answer (irrelevant)

IW: Inexact answer - Weak evidence	IS: Inexact answer - Strong evidence
EW: Exact answer - Weak evidence	ES: Exact answer - Strong evidence

Candidates	**N-A**	**IW**	**IS**	**EW**	**ES**	**Total**
#	4216	68	337	61	621	5303
% (N=5303)	79.5	1.3	6.4	1.1	11.7	100
Avg # per question	84.3	1.4	6.7	1.2	12.4	106 (N = 50)
Avg # per snippet	8.43	0.14	0.67	0.12	1.24	10.6 (N = 500)

On average, 106 answer candidates were extracted per question (10.6 per snippet). The percentage of relevant answers, comprising exact and inexact answers, based on weak or strong evidence, represents 20.5% (1087) of the total number of candidates (5303). Almost two thirds of those relevant answers are exact and well-supported answers (11.7% of the candidates).

Table 5.9 shows figures by satisfaction groups of answer candidates: the EXACT-STRONG refers to candidates judged exact and strongly supported, the EXACT group to candidates judged, whether strongly or weakly supported, and the LENIENT group refers to exact or inexact candidates, whether strongly or weakly supported. The LENIENT group basically represents all relevant candidates, while the EXACT-STRONG group stands for the best set of candidates in terms of exactness and evidence.

On average, each snippet either strongly or weakly supports about 2 EXACT-STRONG answers, and about 3 LENIENT answers. Note that this redundancy is high because the extraction process generated overlapping candidates. Thus a snippet containing an answer was likely to be linked to a high number of satisfying candidates. For instance, to the question *Who was Galileo?*, the same snippet generated the following answers: *"Italian natural philosopher"*, *"Italian"*, *"philosopher"* and *"astronomer"*. This is because keywords as well as chunks were extracted, thus the piece of text *Italian natural philosopher* actually generated 4 candidates, and not just one. The motivation for this redundant type of extraction is explained in the next section about the choices about node projection for graph-based fusion, but, in short, the rationale behind collecting keywords as well as phrases was to increase the ability of the system to compare and distinguish between different aspects of a complex answer. For example, Galileo, in one hand, was known as a *philosopher* and an *astronomer*, and, in the other, he happened to be *Italian*.

Table 5.9: Annotation figures for answer candidates by satisfaction group.

Candidates	**LENIENT**	**EXACT**	**EXACT-STRONG**
#	1087	682	621
% over candidates (N = 5303)	20.5	12.8	11.7
% over relevant candidates (N = 1087)	100	63	57
# per question (N = 50)	21.7	13.4	12.4
# per snippet (N = 500)	2.17	1.34	1.24
# per weak or strong snippet (N = 330)	3.3	2	1.9

Certain of the 1087 answer candidates appear multiple times in the set. Looking at

distinct answer candidate strings, Table 5.10 shows their distribution over the corpus. The first row gives an indication of answer redundancy. In all groups, about 2/3 of all the candidates are actually distinct strings, i.e. multiple answers are more of a case of distinct answer strings rather than multiple occurrences of the same string (frequency).

Table 5.10: Distinct answer candidate strings (ignoring case) per question type (Count per question type and average per question of the same type).

Type/Group	**LENIENT**		**EXACT**		**EXACT-STRONG**	
% over all satisfying candidates	64.2 (N = 1087)		59.3 (N = 682)		60.8 (N = 621)	
	#	Avg.	#	Avg.	#	Avg.
Total (N = 50)	698	14	405	8	378	7.5
DEF (N=15)	356	23.7	222	14.8	201	13.4
FACT (N=5)	70	14	48	9.6	47	9.4
FACT-PERS (N=9)	117	13	63	7	61	6.8
DEF-BIO (N=3)	45	15	14	4.7	13	4.3
FACT-LOC (N=13)	83	6.4	47	3.6	45	3.5
FACT-TIME (N=5)	27	5.4	11	2.2	11	2.2

The highest frequency score for an answer candidate string was 8. But 507 of the LENIENT answer candidate strings (N = 1087) had only one occurrence: For almost half of the relevant answers, trivial frequency is not a factor indicating relevance. For EXACT-STRONG answers (N = 621), 262 answer candidate strings (42%) occured only once. Note though that these figures count the same string only once per snippet. The actual frequency, counting strings that co-occur several times within the same snippet is a bit higher. However, I will show later in this chapter that, although trivial frequency may help top-ranking a relevant answer (hence obtaining a good MRR), it is definitely not sufficient when the task is focused on finding multiple answers, i.e. maximizing answer recall.

Overall, every questions but one had multiple and distinct LENIENT answers (ignoring case). The exception was *What is Africa's largest country? Sudan.* On the EXACT-STRONG level, only 6 questions (12%) had a single satisfying answer. The question that has the most distinct multiple answers on a STRONG-EXACT level was *What is sodium chloride?* with 29 answers. Unsurprisingly, definition questions account for most of the multiple answers, although all types of questions did have multiple answers. It is important though, not to generalize too much from these figures, because the dataset was small (50 questions).

Note that biography questions have very few EXACT-STRONG answers. For instance, to *Who is Duke Ellington?*, the candidate *artist* was judged exact, and strongly supported. On the other hand, *band* (which was judged a "correct" answer in TREC QA) was strictly judged inexact (an exact equivalent would be *bandleader*), but strongly supported (because the source snippet explicitly referred to *Duke Ellington and his band*).

For comparison, Table 5.11 shows the number of TREC answer patterns and distinct TREC systems' answer strings (ignoring case) per question type for the same questions.

Table 5.11: Number of TREC answer patterns and distinct TREC systems' answer strings (ignoring case) per question type compared to EXACT-STRONG answers.

	#TREC patterns	**# TREC strings**	**# EXACT-STRONG**
Total (N = 50)	172	267	378
DEF (N=15)	75	110	201
FACT (N=5)	16	21	47
FACT-PERS (N=9)	28	37	61
DEF-BIO (N=3)	14	17	13
FACT-LOC (N=13)	30	42	45
FACT-TIME (N=5)	9	40	11

This table shows how much TREC patterns are actually an undercount of actual multiple answers (as distinct strings): TREC patterns represent 64.4% of the actual count of distinct answer strings. In TREC data (i.e. answer strings provided by TREC systems processing the AQUAINT corpus), time questions appear to have generated a larger pool of multiple answers. Biography and location questions seem to have a similar number of multiple answers in both corpora. On the other hand, DEF, FACT and FACT-PERS questions had about twice as much multiple answers on the web corpus compared to TREC data.

5.3 Summary

This chapter introduced a new corpus to assess QA tasks concerned with handling multiple answers. While the data collection only represents a small subset of the web data available for 50 questions, it is provided with an exhaustive annotation of multiple answers.

Instead of the traditional binary annotation, correct versus incorrect answer, I described a scaled rating for answer candidate strings, based on two core assessments:

1. the *exactness* of the answer string, in term of pin-pointing and level of informativity.
2. the value of the *evidence* for the answer string, i.e. a score for the textual context (in this corpus, the snippet the answer string was extracted from) as a basis for belief.

While this annotation may appear more lenient than the TREC QA evaluation, because it accepts for instance opinions as long as they are supported by explicit evidence, it is meant to focus specifically on the ability of systems to identify supported answers, rather than assess the absolute truth value of each them. Indeed, when an answer is annotated as "incorrect" it is not clear whether it is because it is unsupported, inexact, or because the expressed opinion defies common knowledge. I argue that, in the later case, systems should not be penalized. I believe that QA systems should be concerned with appropriately reporting existing answers, rather than judging whether such answers are true or not. In automated summarization, systems are not penalized if they summarize information that proves to be incorrect. That is a characteristic of the source of the summary a system cannot be blamed for. Similarly, especially if open-domain QA moves from datasets such as AQUAINT to more heterogeneous, opinion-oriented collections, such as blogs, as it is currently discussed in the TREC QA community, I believe it will be necessary to move beyond answer "correctness".

On the basis of evidence and exactness scaled ratings, I distinguished between three categories of answers. EXACT-STRONG answers represent a core set of candidates that are both supported by strong evidence as well as exact answers. The group of EXACT answer considers candidates that are exact answers but could be supported by either strong or weak evidence. The LENIENT group consists of candidates judged exact or inexact, and supported by either weak or strong evidence. On average, 10 snippets per question provided 7.5 EXACT-STRONG answer candidates, 8 EXACT and 14 LENIENT answer candidates (counting distinct answer strings only). While definition questions (36% of the corpus) have many distinct multiple answers, all question types, including factoids, actually expect multiple answers, showing that answer complexity is not necessarily a matter of question complexity nor question type.

Finally, on the basis of an exhaustive annotation of answer candidates, I described scoring measures to assess an output that provide a list of answer clusters, rather than single answers. The next chapter describes a set of experiments on this corpus, aiming at evaluating the value of fusion with respect to (1) answer re-ranking, (2) multiple answer recognition (focusing on answer recall rather than just question recall), and (3) more structured and more flexible outputs such as answer clusters.

Chapter 6

Information Fusion and Answer Flexibility in Open Domain Web QA

In Chapter 4, I showed that information fusion improved QA performance on location questions. Graph-based modeling and answer candidate comparison (whether those candidates are correct answers or not) provide features that help re-ranking single candidates to a question. While a more traditional approach considers each single candidate with respect to a question, fusion also considers other candidates and makes use of answer multiplicity for re-ranking.

This chapter is concerned with two research questions:

1. Does fusion also improve QA on a corpus with a greater variety of question types, more answer candidates per question (the previous experiment considered 10 candidates per question only), and also answer candidates that are coming directly from an IR engine rather than from the elaborate pipeline of different QA systems, as it was the case in the previous chapter?

2. The process of fusion generates a graph model, i.e. a structured representation of question terms, answer candidates and their relationships. Is it possible to use this representation to provide more structured answers, for instance a list of answer clusters grouping related answers together, instead of a flat list of single answer candidates?

In order to answer those questions, I devised a set of experiments on the corpus built from existing TREC QA questions and answer candidates collected from the web using a search engine (Google), described in Chapter 5. Those experiments evaluate the performance and flexibility of fusion for re-ranking single answer nodes, as well as node clusters.

The first section provides an overview of the system architecture, and describes the processes of node projection and relationship inference for the following experiments. I then define each experiment in terms of view-based tasks. In most experiments, distinct views are derived from the same model for each question, only the the algorithms to select and re-rank nodes vary. I evaluate four main views:

1. A view providing a list of ranked nodes in which re-ranking is based on answer frequency.

2. A view providing a list of ranked nodes in which re-ranking is based on answer redundancy, I will explain the distinction between answer frequency, and answer redundancy. For this view, I will also assess the value of WordNet for relationship inference, as opposed to shallow inference.

3. A view providing a list of ranked clusters of nodes, meant to group together answer nodes that vary in granularity.

4. A view providing a list of ranked clusters of nodes, this time meant to group together related answers, so that the final list of clusters represents either aggregated answers (different aspects of the same answer), or alternative answers.

Finally, I describe a first experiment on using machine learning to build an answer classifier trained on graph models, to show that, although the views described here are derived from models on the basis of heuristics manually devised on a training corpus, it is possible to automatically learn heuristics to obtain a similar performance in answer recall.

6.1 System Architecture and Task Definitions

The experiments described in this chapter aim at showing the value of fusion for QA on a greater variety of question types (location as well as 5 additional types), and a larger set of answer candidates (on average 106 candidates per question against 10 in the experiment on location questions), as well as the flexibility of graph-based models to represent multiple answers.

Also, in the previous experiment (Section 4.1), I used candidates that were provided by the QA systems assessed at TREC QA tracks. Such candidates had already been through a form of QA processing, and this may have generated some kind of bias. In this experiment, I use the raw text provided after a first IR stage, i.e. snippets returned by a web search engine, Google, to the question string.

I first describe the architecture of QAAM for this experiment, which is similar to the one presented in Section 4.2, with the exception of an IR Google-based module. I review the projection process in charge of extracting answer candidates from snippets and mapping them into nodes, and the relationships inferred to draw edges in graph models. I then distinguish between the different graph controllers and subsequent views generated by QAAM, each view corresponding to a different re-ranking strategy.

6.1.1 Architecture

For the experiments described in this chapter, the architecture of QAAM, based on the Model-View-Controller design pattern (Figure 3.2), is similar to the architecture presented in Chapter 4 (Figure 4.2), with the exception of an IR-based Google module retrieving web snippets to answer questions. Modeling is also based on a graph structure in which nodes represent answer candidates extracted from snippets, and edges relationships between candidates.

In an unsupervised pipeline (Figure 6.1), the user types in a question, selects the type of view he or she would prefer the answers to be presented with, and submits the request. On submission, QAAM queries Google with the question string, retrieves the top 10 snippets, generates a model and renders the selected view from the model. This view is then presented to the user. The user can eventually choose another type of view for the same question, and be provided with another view from the same model.

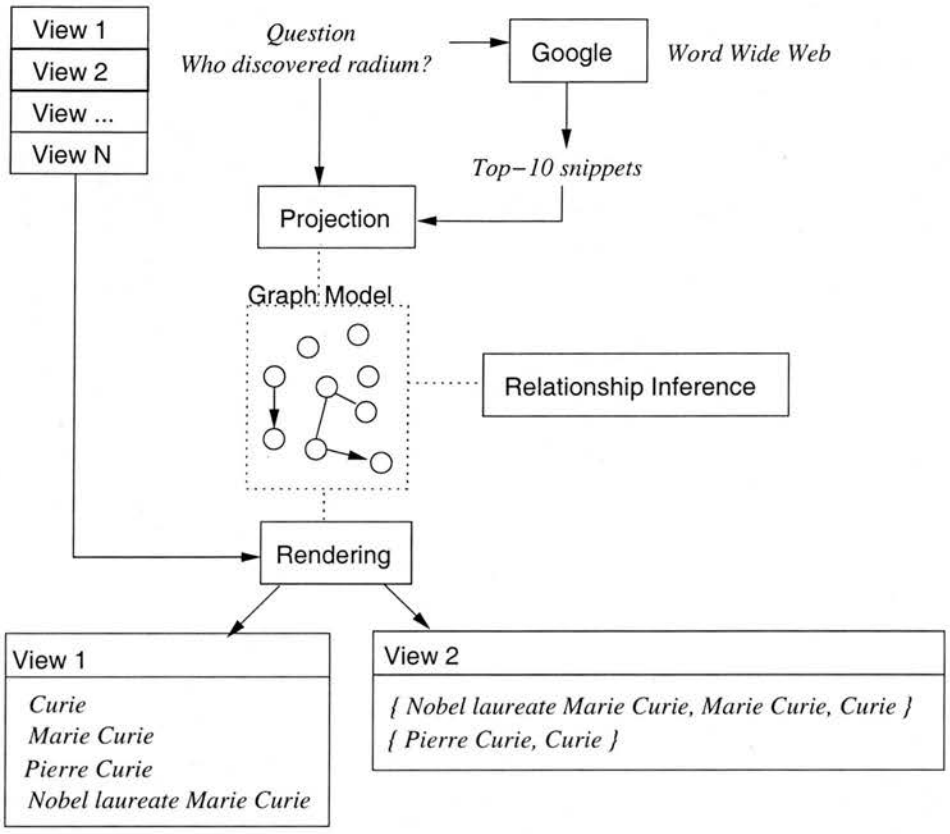


Figure 6.1: QAAM architecture for web-based experiments.

For evaluation purposes, snippets and answer candidates extracted from snippets by QAAM were locally stored and manually evaluated. In this supervised pipeline, QAAM takes as input a list of question, the 10 stored snippets for each question, the type of view to be rendered, and the evaluation file specifying answer exactness and evidence ratings. For each question, a model is generated from the 10 snippets and the requested view is output along with its evaluation scores.

The three following subsections now describe the projection process (mapping of snippets into graph nodes), the relationships and inference techniques used to draw edges between nodes (relationship discovery), and finally the types of view produced from each graph model, which correspond to the tasks defined for evaluation.

6.1.2 Projection

The projection process consists in extracting from the 10 input snippets the answer candidates to be compared and fused in the model. As in Chapter 4 (Section 4.3), QAAM extracted answer candidates that corresponded to nominal keyword and phrase spans. An exception list for English was used to filter out stop words. Note that numbers were not extracted in the previous experiment, but they were in this experiment. There were two other notables differences in the projection algorithm used here.

In Chapter 4, question and candidates were tokenized and then POS-tagged using MXPOST [Ratnaparkhi, 1996]. Span extraction was then based on regular expressions over POS tags (Table 4.4). In the version of QAAM described here, question and snippets were tokenized, POS-tagged and chunked using the OpenNLP library [Baldridge et al., 2001], a set of parsing tools using maximum entropy models. I used the most recent trained models (OpenNLP v.1.3). The use of OpenNLP over MXPOST was not based on performance (both systems use maximum entropy, which has been reported to be satisfying for the parsing task [Hachey, 2002]), but rather to simplify implementation since OpenNLP offers a trained chunker.

The second difference is that the projection process described here extracted overlapping spans, generating redundancy among candidates (Subsection 5.2.3). For instance, the following candidates: *Italian natural philosopher*, *Italian*, *philosopher* and *astronomer*, were extracted from the same snippet found for the question *Who was Galileo?* The chunk *Italian natural philosopher* actually generated 4 candidates: the full span, as well as the relevant keywords in the span.

The reason for collecting phrases as well as their inner keywords is to increase the ability of the system to compare and distinguish between different aspects of a complex answer. For example, as mentioned earlier, Galileo, in one hand, was known as a *philosopher* and an *astronomer*, and, in the other, he happened to be *Italian*.

As shown in Section 5.2 (Table 5.8), on average each snippet generated about 10-11 candidates. There were about 106 candidates per question. These were the candidates manually rated for exactness and evidence.

QAAM took as input the question and its 10 associated web snippets, and extracted the answer candidates from the snippets into a list of answer candidates. While in the previous experiment on location questions, the candidates were directly mapped as nodes of the graph model, here stem-matching candidates were grouped together into clusters, so that one node of the graph model would actually refer to a list of stem-matching candidates. Each node contained the following information:

- the list of answer candidate strings
- a pointer to the snippets they were extracted from
- the tag for the chunk, among *CD* (number), *NN* (common name), *NNP* (proper name), *JJ* (adjective) and *O* (other, e.g. symbol).
- a bag of stems representing the candidates
- the total number of occurrences of the candidate strings (frequency)

For instance, the following answer candidates to the question *What is Teflon?*: *coatings* (occurred 3 times in snippet 7 and also 3 times in snippet 10), *coating* (occurred once in snippet 9), and *coated*(occurred once in snippet 9) were projected into one node containing the following information:

- Strings: *coatings*, *coating*, *coated*
- Snippets: 7, 9, 10
- Dominant chunk tag: NN (the tagging was actually sparse between *JJ*, *NN* and *NNP*, but the most frequent was *NN*).
- Bag of stems: $\{coat\}$
- Frequency: 8

The comparison process, the inference of relationships between candidates, was thus based on nodes that had already merged stem-matching candidates. I now describe the relationships (edges) that could be possibly drawn such nodes, and the techniques used to infer them.

6.1.3 Relationship Discovery

The inference process involves comparing each pair of nodes in the graph in order to draw the graph's edges. This time, I considered three types of connections:

1. Equivalence ($\leftrightarrow$)
2. Inclusion ($\rightarrow$) and its symmetric relation: entailment
3. Aggregation ($\leftrightsquigarrow$)

There were two levels of inference. One level consisted of shallow comparisons between nodes, essentially based on stem comparison and co-occurrences. A more advanced level made use of WordNet to lookup lexical and semantic relationships. Table 6.1 describes the techniques used to infer each relationship.

The first type of equivalence, based on stem-matching, was the comparison used to merge equivalent candidates into a single node. The other relationships were inferred by comparing nodes, i.e. a set of stem-matching candidates.

To infer inclusion and entailment between nodes using shallow computations, I considered two types of overlap between the bags of stems representing nodes to be compared. The most specific overlap considers whether a node is the lexical head of another node. It is approximated by checking whether the first node is a suffix of the other node. or instance, *healing system* would be connected to *ancient healing system* by an inclusive edge (*healing system* $\rightarrow$ *ancient healing system*) in the model generated about acupuncture.

The second technique based on overlap subsumed the first, and was used only if a given node was not a lexical head, but still represented a subset of the other node. For instance, *ancient* would be considered to be inclusive of *ancient healing system* (*ancient* $\rightarrow$ *ancient healing system*).

The two types of overlap (suffix as opposed to non-suffixed overlap) were distinguish in order to approximate two types of inclusion, one based on the lexical head, the other based on a qualifier. Depending on the type of node that is included, a "qualifier" parent may be less relevant than a "lexical head" parent.

Table 6.1: Techniques used to infer relationships in graph models.

Equivalence	Stem-matching
Two candidates are equivalent if their bag of stems match. Note: This is the technique used to generate graph nodes, and thus is based on single candidates. Example: *common salt* $\leftrightarrow$ *common salts*	
Equivalence	WordNet: synonym, derivation, participle, and similarity links
Two nodes are equivalent if a WordNet link has been found found between their two string values. Examples: *philosopher* $\leftrightarrow$ *philosophy* (derivation), *two* $\leftrightarrow$ *2* (synonym), *dark* $\leftrightarrow$ *tenebrous* (similar)	
Inclusion/Entailment	Stemmed lexical head matching
A node X includes a node Y if it represents the lexical head of node Y. Example: *Pacific* $\rightarrow$ *western Pacific*	
Inclusion/Entailment	Bag of stems overlap
A node X includes a node Y if its bag of stems is a subset of the Y's bag of stems. Example: *Chinese* $\rightarrow$ *Chinese therapy*	
Inclusion/Entailment	WordNet: transitive hypernym, holonym and attribute links
A node X includes a node Y if a WordNet inclusion link has been found from X's string value to Y's. Examples: *France* $\rightarrow$ *Paris* (holonym/meronym), *treatment* $\rightarrow$ *acupuncture* (hypernym/hyponym), *age* $\rightarrow$ *old* (attribute)	
Aggregation	Source snippet co-occurrences
Two nodes X and Y are aggregated if $\lvert S_x \cup S_y \rvert > 1$ and $\lvert C_x \cap C_y \rvert \,/\, \lvert C_x \cup C_y \rvert \geq 0.5$ where S_x represent the set of source snippets for the node X. (And S_y the set for Y.) Example: *anatomic* (snippets 1,3,5) $\leftrightsquigarrow$ *body* (snippets 3,5,6)	

In the case of nodes that are named entities, for instance, a node *Mount* including *Mount Lhotse* (*What is the fourth highest mountain in the world?*), or *Marie* including *Marie Curie* (*Who discovered radium?*), the parent acting as a "qualifier" is likely to be less relevant than the lexical head for answering. On the other hand, in the case of included nodes representing common names, especially when answering definition questions, all parent nodes may be relevant, especially when the goal is to identify different aspects of the same answer (e.g. *ancient* in *ancient healing system*, or *Chinese* as "qualifier" parent of *Chinese therapy*).

The aggregation relationship was a new relationship meant to measure contextual relatedness between two nodes. I will show later in experiments on answer flexibility that equivalence and inclusion are not sufficient on their own to cluster candidates that refer to the same "answer", i.e. a referential entity that can be described using paraphrases (equivalence), different levels of granularity (inclusion/entailment) but also that can present different aspects or attributes, that are not related in terms of equivalence or inclusion, but do refer to the same entity. For example, vertigo can be defined as *dizziness*, *lightheadedness* (equivalence between candidates), or as a *medical symptom* (entailment), but it can also be described as a *fear of heights* (aggregation). However, if one refers to Vertigo, the *movie*, *film* (equivalence), or more exactly a *thriller* (inclusion), then the candidate *director Alfred Hitchcock* is to be aggregated with nodes referring to the movie, not the symptom. Computation of term co-occurrences give an indication of such relatedness.

Edges drawn from these techniques had a general label as an indication of the relationship (equivalence, inclusion, entailment, aggregation), as well as a marker of the technique used to infer this relationship.

Finally, I developed on the training corpus different heuristics based on the graph's topology to select, re-rank and render nodes into distinct views of the same model.

6.1.4 View-based Task Definitions

A view defines both a controlled selection over the answer content, and the format, or rendering of the answer. It may depend on a user preferences or a task requirements.

For location questions, both evaluated views provided a ranked list of answer strings, i.e. the rendering was text-based, but the selection of the graph nodes representing answer candidates (whose string value was output) varied from one view to another: Different graph-based features were used to retrieve and rank nodes (answer content).

In this chapter, each strategy was also associated with a distinct view. All the views generated text only. While in the previous experiment, a node corresponded to a single answer candidate, in this version, QAAM generated nodes based on several, stem-matching, candidates. Two kinds of view were considered:

1. A ranked list of nodes, i.e. a cluster of stem-matching candidates.
2. A ranked list of clustered nodes, i.e. clusters of clusters of stem matching candidates.

A example of a ranked list of nodes would be for instance:

What is Teflon?
1. {*coatings, coating, coated*}
2. {*non-stick coatings*}
3. {*fluorinated ethylene*}
4. {*ethylene*}
5. {*registered trademark*}

While a ranked list of node clusters could be:

What is Teflon?
1. {{*coatings, coating, coated*}; {*non-stick coatings*}}
2. {{*fluorinated ethylene*}; {*ethylene*}}
3. {{*registered trademark*}}

Each view differed from the type of filtering and selection performed on the graph model. Each view represents a re-ranking strategy, derived from the same models, but using different features.

When considering node re-ranking (as opposed to node clusters), I compare the following views:

- An **optimal** view that optimizes the re-ranking of nodes referring EXACT-STRONG answer candidates. I use this view as the upper bound for re-ranking.
- A **Google** view, i.e. a view that ranks nodes according to the original Google ordering. This view can be considered as the lower bound for re-ranking, the performance bound under which a re-ranking strategy is not worth using since the original Google ranking performs better.
- A **frequency-based** view: A view producing a list of nodes sorted by frequency, i.e. the sum of the frequency of each candidate merged into a node.
- A **redundancy-based** view: A view producing a list of nodes sorted by redundancy, an extended version of frequency, based on graph connections between nodes.

At this stage, I will compare modeling using shallow inference against modeling using WordNet lookups as well as shallow inference. To facilitate error analysis, I will perform what I call "incremental modeling": Using the top 10 answer nodes re-ranked by redundancy, as well as question nodes, QAAM generates new models and renders new views. The process involves (1) generating a first, large, model (on average a bit more than 100 nodes per model), (2) selecting the top-10 nodes on the basis of their redundancy score, (3) generating a new model based on the question nodes and the top 10 nodes previously selected, and (4) rendering views from the new model (Figure 6.2).

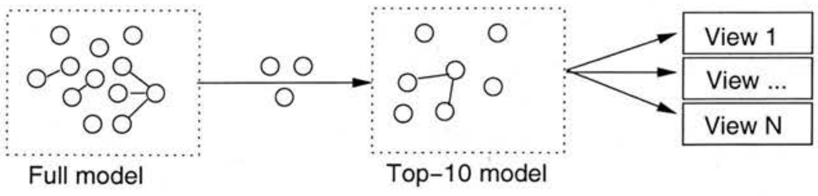


Figure 6.2: Incremental modeling.

From these models based on a subset of nodes, I compare the following views:

- A view producing a list of nodes sorted by redundancy, based on new models generated from shallow fusion (**top-10**).
- A view producing a list of nodes sorted by redundancy, based on new models generated from shallow fusion and WordNet (**top-10 WordNet**).
- A view producing a list of node clusters, where clusters are groups of nodes connected by entailment and equivalence, hence representing granularity between co-referring candidates (**entailment clusters**).
- A view producing a list of node clusters, where clusters are groups of nodes connected by inclusion, entailment, equivalence and aggregation (maximal subgraphs or connected components of the model), aiming at a representation of alternative answers (**connected components**).

Table 6.2 lists each view, the type of of inference performed, the number of models generated (incremental modeling or not) and the entry type of the output ranked list.

To evaluate views, I will focus on the following scoring measures (Section 5.1.3):

- Ranking measures are given by the MRR, indicating the mean rank of the first satisfying list entry, the question recall at rank, and the answer recall at rank, respectively measuring at a given rank the percentage of questions answered, and the percentage of answer candidates found.
- The accuracy of each output list is given by specificity, precision, NPV, F-Measure and overall accuracy measures.
- Clustering is assessed with clustering precision for each level (EXACT-STRONG, EXACT and LENIENT) and general clustering exactness scores.

Scores are computed from the list of answer candidates found at a given rank, whether the ranked entry is simply a node (merging of stem-matching candidates) or a cluster of nodes. For instance, if, for a given question, answer candidates *A* and *B* are satisfying on an EXACT-STRONG level (100% on evidence, and 100% on exactness), and *C* is a LENIENT answer (for instance, 50% evidence, 50% on exactness), a node entry such as $\{A,B\}$, and a cluster entry such as $\{\{A,B\}, \{C\}\}$ will both be considered satisfying entries on an EXACT-STRONG level, as they both contain at least one EXACT-STRONG answer. They will respectively obtain an EXACT-STRONG cluster precision of: 2/2 = 100% for the node, and 2/3 = 66.6% for the cluster, and a general cluster exactness of: (100+100)/2 = 100% for the node, and (100+100+50)/2 = 83.3% for the cluster.

Table 6.2: List of views

View	**# Models**	**Inference**	**Output**
Optimal	1	Shallow	Nodes
Google	1	Shallow	Nodes
Frequency	1	Shallow	Nodes
Redundancy	1	Shallow	Nodes
Top-10	2	Shallow	Nodes
Top-10 WordNet	2	Shallow/WordNet	Nodes
Entailment clusters	2	Shallow	Node clusters
Connected components	2	Shallow	Node clusters

The following two sections focus on, first, node re-ranking, and then cluster re-ranking.

6.2 Node Re-ranking

This section is concerned with the performance of fusion for node re-ranking. I propose to first review the upper and lower bounds I defined for node re-ranking: a strategy that optimizes the ranking of EXACT-STRONG candidates, and then a baseline based on the original Google rank. I then review different strategies comparing frequency as opposed to redundancy (an extended version of frequency, based on fusion). I show that graph model provide more flexibility when it comes to sort answers on the basis of popularity. Finally I assess the value of WordNet for re-ranking a subset of 10 nodes.

6.2.1 Optimal Node Re-ranking

An optimal re-ranking strategy would be provided by a system that rank all the nodes containing a satisfying answer candidate at the top of the list provided as a view. It basically provides an indication of what the best scores can be for a view that re-ranks nodes, knowing that a single node represent a list of stem-matching candidates. In this subsection, I first provide figures that gives an indication of the value, in terms exactness and precision, of stem merging to produce answer nodes. I then present the results of optimal re-ranking of answer candidates. The satisfaction level for which the ranking is optimized for is EXACT-STRONG, meaning that nodes containing containing EXACT-STRONG candidates are privileged (higher ranked) than nodes containing EXACT or LENIENT answers.

Table 6.3 first presents figures characterizing nodes based on stem-matching, in terms of precision and exactness, for all question and for all nodes. For the 50 questions, QAAM produced overall 4963 nodes of stem matching candidates, an average of 99.26 nodes (against an average of 106 answer candidates) per question, with a minimum of 65 nodes and a maximum of 138 nodes. 672 nodes contained at least one LENIENT answer candidate (an average of 13.44 per question). 385 nodes contained at least one EXACT answer candidate (7.7 per question). 359 nodes contained at least one EXACT-STRONG answer candidate (7.18 per question). Because of the merging, there were a bit less nodes than actual distinct answer candidate string (about 13.54% of satisfying nodes on the LENIENT level, against 14 for distinct strings).But merging did not create large nodes. On average, a node contained barely more than a single answer candidate (minimum: 1 and maximum: 3 answer candidates per node).

I used the clustering precision and clustering exactness scores defined in Subsection 5.1.3 to assess the quality of the stem-based merging of answer candidates into nodes, i.e. scores for merging precision and merging exactness (i.e. clustering precision and clustering exactness at node level). It is important to distinguish between merging precision and exactness, characterizing the stem-based merging that produces nodes, and clustering precision and exactness, characterizing the fusion-based clustering that produces clusters of nodes. Later in the chapter (Section 6.3), I will show QAAM generating views listing clusters of nodes, i.e. clusters of previously merged candidates. Then clustering precision and exactness will be provided to characterize *node clustering*, as opposed to the *stem-based merging* used to generate nodes that is evaluated here.

Table 6.3: Figures for nodes merging stem-matching answer candidates, measured from the full list of ranked nodes per question.

ES: EXACT-STRONG, E: EXACT, L:LENIENT

Group	ES	E	L
Merging precision	7.52%	8.09%	14.18%
Positive merging precision	99.02%	99.22%	99.40%
Merging exactness		11.14%	
Positive merging exactness		78.12%	
# nodes	4963	4963	4963
# satisfying nodes	359	385	672
% satisfying nodes (N = 4963)	7.23	7.75	13.54

In terms of positive merging precision, i.e. the percentage of satisfying answer candidates per satisfying node is more than 99%, demonstrating that what I call a "satisfying" node is a very precise entity: Most of the candidates members of such a node are satisfying. In other words, stem-matching merging produces very little noise.

The average node precision (including satisfying and non-satisfying nodes) is low, between 7 and 14 %, which is expected because the figures are computed on the full list of nodes (an average size of 99.26 nodes per question), and such a list contains around 86% of irrelevant nodes.

Positive merging exactness is slightly lower, about 78%. Note again that this exactness score is independent from a satisfying group. It is measured overall from the manual evaluation rating which rated supported candidates as exact (100%) or inexact (50%). Irrelevant answers had an exactness of 0%. If a node contained 2 satisfying candidates with respectively an exactness rating of 50 and 100%, the positive exactness would be 75%. Thus the positive exactness indicates that stem-based merging may induce the merging of candidates that do not have the same level of exactness. This is somehow unsurprising because the human exactness rating was very strict. For example, the node that merged together $\{coatings, coating, coated\}$ had an exactness of only 83.3% because *coated* was not judged as exact as *coating* to answer the question *What is Teflon?* (Teflon is a coating, not coated).

Table 6.4: Figures for nodes merging stem-matching answer candidates, measured from the top-10 nodes per question ranked by a strategy optimizing answer recall for the EXACT-STRONG group.

ES: EXACT-STRONG, E: EXACT, L:LENIENT

Group	ES	E	L
Merging precision	57.30%	57.40%	62.50%
Positive merging precision	98.79%	98.96%	99.20%
Merging exactness	59.95%		
Positive merging exactness	95.15%		
# nodes	500	500	500
# satisfying nodes	290	290	315
% satisfying nodes (N = 500)	58	58	63

The merging exactness for nodes, as the overall merging precision was low because computed on the full list of ranked nodes per question. As a reference point, Table 6.4 provides the same scores, based this time on the top-10 nodes ranked by a strategy that optimizes answer recall for the EXACT-STRONG group.

Exactness overall increases, since only the top-10 nodes are considered. Precision, on the hand, is slightly lowered: Very precise nodes have been eliminated for the same reason. Also, because exactly 10 nodes are selected, the list may contain unsatisfying nodes as well (for instance if the question had very few EXACT-STRONG answers).

Figure 6.3 shows both question and answer recalls at rank for the same optimal strategy. Because the optimal strategy ranks first all the satisfying nodes, the question recall is a flat curve topping at 100% from rank 1, in other words, with an optimal strategy, all the questions can be answered satisfyingly with the first ranked node.

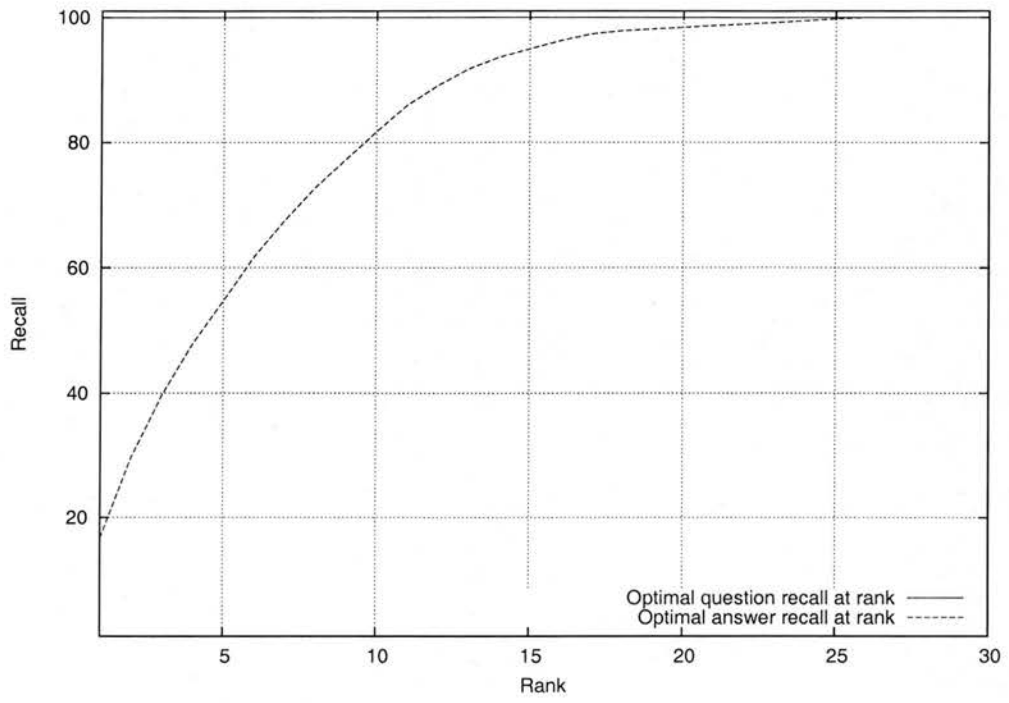


Figure 6.3: Answer recall (N = 378) and question recall (N = 50) at rank for answer candidate strings following an optimal order. Satisfaction group: EXACT-STRONG. The maximum rank is actually 138. A rank entry corresponds to a single node.

The optimal answer recall at rank 1 is 16.66% for the EXACT-STRONG group. The optimal strategy provided a satisfying node at rank 1 for all the questions. 38 of those nodes were singletons, i.e. referred to a single EXACT-STRONG answer candidate. 10 nodes actually merged two EXACT-STRONG candidates each, 1 node merged 3 EXACT-STRONG candidates, and 1 node merged 2 EXACT-STRONG with an an EXACT candidates. Hence, at rank 1, the best recall was 63 EXACT-STRONG answer candidates over 378 satisfying answer candidates for the whole corpus (16.66%).

Table 6.5: Re-ranking scores (at rank 10) for a strategy optimizing answer recall for EXACT-STRONG answers. Figures are computed over distinct answer candidate strings.

ES: EXACT-STRONG, E: EXACT, L:LENIENT

AR: answer recall, QR: question recall

NPV: Negative Predictive Value

System	**Optimal**		
Group	ES	E	L
MRR	1	1	1
AR - rank 1	16.66	15.55	9.16
AR - rank 10	81.74	76.54	48.71
Min rank for 100% AR	26	98	121
# Satisfying Answers Found	309 (N = 378)	310 (N = 405)	340 (N = 698)
QR - rank 1	100	100	100
QR - rank 10	100	100	100
Min rank for 100% QR	1	1	1
# Questions	50	50	50
Specificity	94.62	94.61	94.94
Precision	54.98	55.16	60.49
NPV	98.47	97.90	92.08
F-Measure	65.74	64.11	53.96
Accuracy	93.66	93.17	88.59
# Satisfying Nodes Found	290 (N = 359)	290 (N = 385)	315 (N = 672)

On average, from rank 1 to 10, nodes (whether satisfying or not) merged 1.12 answer candidates. Satisfying nodes within the top 10 had a mean size of 1.09.

Table 6.5 summarizes optimal scores. Note that the answer recall at rank tends to decrease from the EXACT-STRONG to the LENIENT group. This is because there are many more LENIENT answers than STRONG-EXACT one. A larger number of satisfying answers (larger N) tends to lead to a lower answer recall. Also, because this strategy optimizes answer recall for EXACT-STRONG answers, ranking them at the top, EXACT and LENIENT answers are necessarily ranked later, and not all of them appear within the top-10.

Note also that this optimal re-ranking does not discard unsatisfying nodes. Thus, if a question had only one possible satisfying node as answer (optimally ranked first), the 9 others lowered the results, especially the precision score. Also, because some questions actually had many more than 10 satisfying nodes, this optimal cannot achieve an answer recall of 100% within the top 10 nodes. Consequently, this "optimal" strategy, because it does not classify answers as satisfying or not (this is a somewhat different task) but simply re-ranks them to produce a list of the top 10, cannot achieve an accuracy of 100%. Also, accuracy is not actually the best indicator for QA performance. Because the percentage of irrelevant candidates is extremely high, a list of top 10 candidates (over about 100 candidates per question) will produce a high specificity, i.e. the percentage of appropriately discarded answers will be high. A better indicator is the F-Measure, based on answer recall and precision. The optimal F-Measure for EXACT-STRONG answers is 65.74%.

In the following subsections, I will compare the evaluation results of different re-ranking strategies with optimal figures at rank 10.

6.2.2 Node Re-ranking Based on Google Rank

I now consider the lower bound for re-ranking, sorting nodes on the basis of the original Google rank. For each question, a view was generated that ranks each node according to its best Google rank. Since a node merges several answer candidates, appearing in snippets at different ranks and in a different order, the best Google rank of a node is the best rank in the list of its merged answer candidates. This view basically provides the re-ranking bottom-line, under which any other re-ranking strategy is simply worse than the search engine original order. The final output list was limited to 10 nodes.

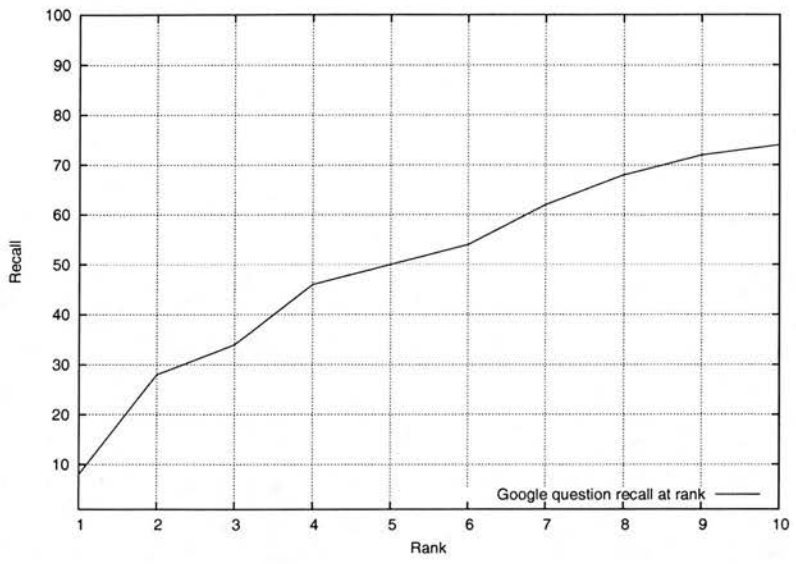


Figure 6.4: Question recall at rank for answer candidates. Node re-ranking following Google ordering. Satisfaction group: EXACT-STRONG.

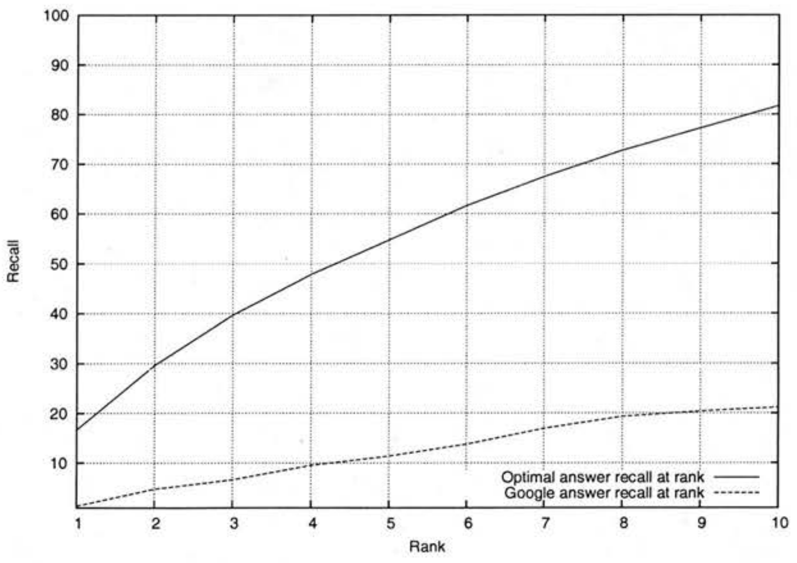


Figure 6.5: Answer recalls at rank for answer candidates. Node re-ranking following Google ordering vs optimal re-ranking. Satisfaction group: EXACT-STRONG.

Table 6.6: Google re-ranking scores at rank 10.

ES: EXACT-STRONG, E: EXACT, L:LENIENT

AR: answer recall, QR: question recall

NPV: Negative Predictive Value

System	**Google**		
Group	ES	E	L
MRR	0.27	0.28	0.35
AR - rank 1	1.32	1.48	1.43
AR - rank 10	21.16	20.49	17.90
# Satisfying Answers Found	80 (N = 378)	83 (N = 405)	125 (N = 698)
QR - rank 1	8	10	18
QR - rank 10	74	76	78
# Questions	50	50	50
Merging precision	14.2	14.8	22.89
Positive merging precision	100	100	99.56
Merging exactness	18.85		
Positive merging exactness	81.95		
# Satisfying nodes Found	71 (N = 359)	74 (N = 385)	115 (N = 672)
Specificity	90.50	90.51	90.83
Precision	15.18	15.74	23.71
NPV	93.46	92.93	87.43
F-Measure	17.67	17.81	20.40
Accuracy	85.35	84.93	80.82

When preserving the Google order by which answer candidates appear, the MRR at rank 10 of the output ranked list was 0.27 for the group of EXACT-STRONG answers. As shown on Figure 6.4, 8% of the questions were answered at rank 1, and 74% at rank 10. Figure 6.5 shows the answer recall at rank for the EXACT-STRONG group on the same baseline. The curve for the optimal recall at rank is shown on the same figure for comparison. At rank 10, 21.16% of the satisfying answer candidates were found (against 81.74% in optimal re-ranking).

Table 6.6 summarizes the different scores for a re-ranking based on Google ordering. Looking at merging precision and exactness, it appears that that satisfying nodes ranked at the top had a high precision (100% for EXACT-STRONG answers, against 98.79% for the optimal strategy). This is possible because the Google strategy only found 71 satisfying nodes (overall merging 80 satisfying answer candidates), as opposed to 290 for the optimal strategy. It is thus not surprising that Google ranking achieves a higher merging positive precision, since fewer satisfying nodes were found. On the other hand, the merging precision (computed over both satisfying and unsatisfying nodes) reflects the low performance of the Google strategy: the mean node precision is 14.2%, against 57.30% for the optimal strategy.

The overall exactness of nodes is also low (18.85% against an optimal score of 59.95%). The mean positive exactness is 81.95% (vs an optimal score of 95.15%). Note again that exactness is computed independently: It is not specific to a satisfaction group. Positive exactness measures the mean exactness of nodes that have at least one candidate for which exactness is greater than 0%, thus including EXACT and LENIENT answers. Thus a positive exactness of 81.95% shows that although Google found (at rank 10) very few nodes that were satisfying on the EXACT-STRONG level, it also top-ranked nodes that merged EXACT or LENIENT answer candidates.

Overall, we can consider that, for EXACT-STRONG answers, an optimal node re-ranking strategy achieves a F-Measure at rank 10 of 65.74%, while the bottom F-Measure is 17.67%. Similarly, the MRR at rank 10 will evolve between 0.27 (Google bottom-line for EXACT-STRONG) and 1. Answer recall at rank 10 covers at best 81.74% of the EXACT-STRONG answers, or only 21.16% of them on the basis of Google ordering.

The next subsection describes new re-reranking strategies, comparing re-ranking based on candidate frequency against re-ranking based on a larger set of fusion-based features.

6.2.3 Frequency versus Redundancy

This section examines the value of candidate frequency for re-ranking when considering a small corpus. On a large corpus (TREC 9 AQUAINT corpus), Clarke et al. reported a drop of 12% in performance when omitting "redundancy" as a feature [Clarke et al., 2001]. In this thesis, I distinguish between "frequency" and "redundancy". Frequency ("redundancy" in Clarke et al.) refers to the total number of occurrences of *an answer string* in a given corpus, e.g. how many times *Agra* occurs in the dataset retrieved for the question *Where is the Taj Mahal?* I define "redundancy" as the total number of occurrences of *an answer* in a given corpus, e.g. how many times the answer *Agra* is referred to in the corpus. As an answer, it can be expressed as *Agra*, but also as *Uttar Pradesh* or *India*. The three terms refer to the *same* answer. Thus, the popularity of an answer, if we take that popularity is approximately indicated by how often the given answer is mentioned in a corpus does not necessarily correlate with the frequency of a single answer string, but rather the sum of the frequency counts of answer strings referring to the same answer. While frequency is based on trivial string matching, redundancy, as I define it, may involve more complex relationships, for instance granularity in the case of *Agra, Uttar Pradesh, India*.

Using graph-based fusion, it is possible to approximate a frequency score (number of candidates merged into a node, i.e. stem-based frequency), as opposed to a redundancy score (sum of the frequency score for nodes connected by inclusion, entailment and aggregation as well). While frequency is based on an approximation of equivalence between answer candidates, redundancy involves comparing candidates not only for equivalence but also inclusion, entailment and aggregation.

In this section, I assess three re-ranking strategies based on shallow fusion (WordNet lookups were not considered). From a model that had fused the question and its answer candidates, QAAM produced a ranked list of nodes based on the three following sorts:

1. Answer nodes are sorted according to their frequency score: the number of answer candidates merged in the node by stem-matching (**frequency** view).

2. Answer nodes that directly overlap with a question node are penalized and score

0, other nodes are sorted by frequency (**frequency+** view). The overlap with a question node is defined as an inclusion or entailment edge (constructed from bag of stems overlap, and lexical head matching, see Table 6.1). For instance, *Chinese currency*, as well as *currency exchange rates*, overlap with *currency* (*What is the currency used in China?*). Those nodes simply scored 0. On the training corpus, it appeared that most answer nodes overlapping with a question nodes were irrelevant.

3. As in the second strategy, answer nodes that overlapped with a question node scored 0. If a node was a singleton, i.e. had no connection by inclusion, entailment or aggregation in the graph, its frequency was used as a score. Otherwise, the score of the node was its frequency times the frequency of each member of its entailment branch. For instance, *Pierre Curie* (frequency=3) entailed *Curie* (frequency=16), as well as *Pierre* (frequency=4). The score for the node *Pierre Curie* was 3*(3+16+4)=69. This heuristic was close to the one used in Chapter 4 taking into account the number of children by transitive inclusion (feature (e), Section 4.4). The more specific a node (i.e. the biggest its set of entailed neighbours), the better. The more redundant the entailment tree, the more likely the node was to be ranked high. This scoring boosted nodes that were very specific (longer spans), and often had a low frequency but had an entailment parent with a high frequency (**redundancy** view).

Tables 6.7, 6.8 and 6.9 summarizes the scores obtained for each strategy. Frequency on its own was sufficient to multiply Google MRR by more than 2, and to contribute about 12% more satisfying answers to the top 10 list (answer recall of 32.80% at rank 10 on EXACT-STRONG). But, overall, the third strategy, cumulating frequency, redundancy and specificity based on graph connections, was the best performing strategy with, at rank 10, a MRR of 0.72, and an answer recall of 40.47% for EXACT-STRONG answers (against an optimal answer recall of 81.74%). Overall, the top 10 nodes had a high positive merging precision (98.14% against 98.79% in optimal re-ranking) and a decent positive merging exactness (81.62%, against 95.15% in optimal re-ranking). The list precision score for answer candidates was 27.71% (against an optimal 54.98%).

Table 6.7: Re-ranking scores at rank 10 for a strategy based on frequency only (strategy 1). Figures are computed over distinct answer candidate strings.

ES: EXACT-STRONG, E: EXACT, L:LENIENT

AR: answer recall, QR: question recall

NPV: Negative Predictive Value

Group	ES	E	L
MRR	0.59	0.59	0.68
AR - rank 1	7.14	6.66	4.85
AR - rank 10	32.80	30.86	26.93
# Satisfying Answers Found	124 (N = 378)	125 (N = 405)	188 (N = 698)
QR - rank 1	44	44	54
QR - rank 10	92	92	98
# Questions	50	50	50
Merging precision	21.19	21.29	32.89
Positive merging precision	98.14	98.61	99.09
Merging exactness	27.1		
Positive merging exactness	81.62		
# Satisfying Nodes Found	108 (N = 359)	108 (N = 385)	166 (N = 672)
Specificity	90.22	90.19	90.97
Precision	21.23	21.40	32.19
NPV	94.35	93.78	88.67
F-Measure	25.77	25.27	29.32
Accuracy	85.96	85.46	82.18

Table 6.8: Re-ranking scores at rank 10 for a strategy based on frequency and question overlap filtering (strategy 2). Figures are computed over distinct answer candidate strings.

ES: EXACT-STRONG, E: EXACT, L:LENIENT

AR: answer recall, QR: question recall

NPV: Negative Predictive Value

Group	ES	E	L
MRR	0.63	0.63	0.72
AR - rank 1	7.93	7.40	5.01
AR - rank 10	32.01	30.12	26.07
# Satisfying Answers Found	121 (N = 378)	122 (N = 405)	182 (N = 698)
QR - rank 1	50	50	60
QR - rank 10	92	92	98
# Questions	50	50	50
Merging precision	20.63	20.73	31.99
Positive merging precision	98.25	98.73	98.76
Merging exactness	26.36		
Positive merging exactness	81.37		
# Satisfying Nodes Found	105 (N = 359)	105 (N = 385)	162 (N = 672)
Specificity	90.33	90.30	91.02
Precision	21	21.18	31.59
NPV	94.30	93.72	88.55
F-Measure	25.36	24.87	28.57
Accuracy	86	85.5	82.10

Table 6.9: Re-ranking scores at rank 10 for a strategy based on shallow fusion expressing redundancy (strategy 3). Figures are computed over distinct answer candidate strings.

ES: EXACT-STRONG, E: EXACT, L:LENIENT

AR: answer recall, QR: question recall

NPV: Negative Predictive Value

Group	ES	E	L
MRR	0.72	0.72	0.78
AR - rank 1	9.52	8.88	5.73
AR - rank 10	40.47	38.51	32.52
# Satisfying Answers Found	153 (N = 378)	156 (N = 405)	227 (N = 698)
QR - rank 1	62	62	68
QR - rank 10	92	92	96
# Questions	50	50	50
Merging precision	27.53	28.13	41.59
Positive merging precision	99.75	99.76	99.52
Merging exactness	34.86		
Positive merging exactness	83.41		
# Satisfying Nodes Found	138 (N = 359)	141 (N = 385)	209 (N = 672)
Specificity	91.52	91.54	92.59
Precision	27.71	28.26	41.12
NPV	95.03	94.50	89.61
F-Measure	32.90	32.60	36.31
Accuracy	87.73	87.31	84.34

Figures 6.6 and 6.7 show respectively the question recall at rank, and the answer recall at rank, for each strategy, compared to the Google baseline and the optimal strategy (The optimal question recall is 100% and is not drawn on Figure 6.6).

The question recall at rank for the three strategies tend to converge at rank 5-6 (about 90% of the questions are covered with at least one satisfying answer within the top 5-6 nodes). On the other hand, there is a distinct increase in answer recall at rank 10: The third strategy found about 7% more answers than the strategy based on frequency only.

For significance testing, I used a two-sided Wilcoxon-Rank-test for pairwise ascertained samples to compare the node precision scores of each output list on the EXACT-STRONG level. The precision score for each node (between 0 and 100%) reflects the level of satisfaction for the node (if the score is 0, the node is irrelevant, a score of 100% indicates that all the candidates merged into the node are satisfying). A ranked list of precision scores reflects the value of the reranking strategy as well as the quality of the merging in each node. After two-sided tests, the performance of the third strategy proved to be significantly higher ($p < 0.01$) than the Google based re-ranking, and also significantly higher ($p < 0.01$) than both the first strategy, based on frequency only, and the second strategy (frequency and question overlap).

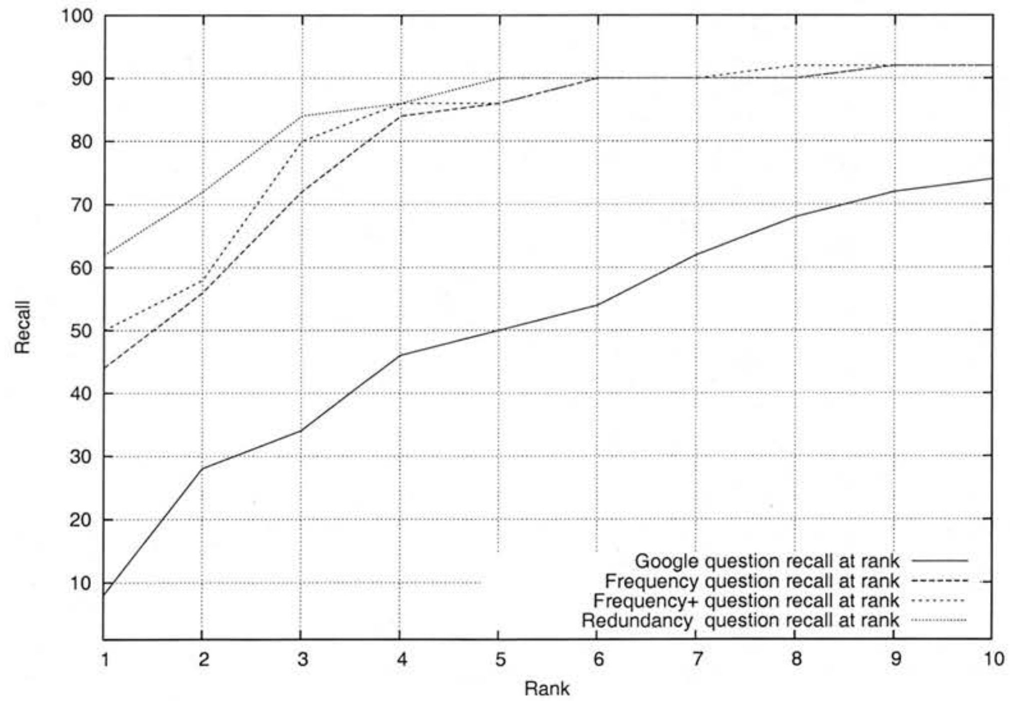


Figure 6.6: Question recall at rank for answer candidates. Satisfaction group: EXACT-STRONG.

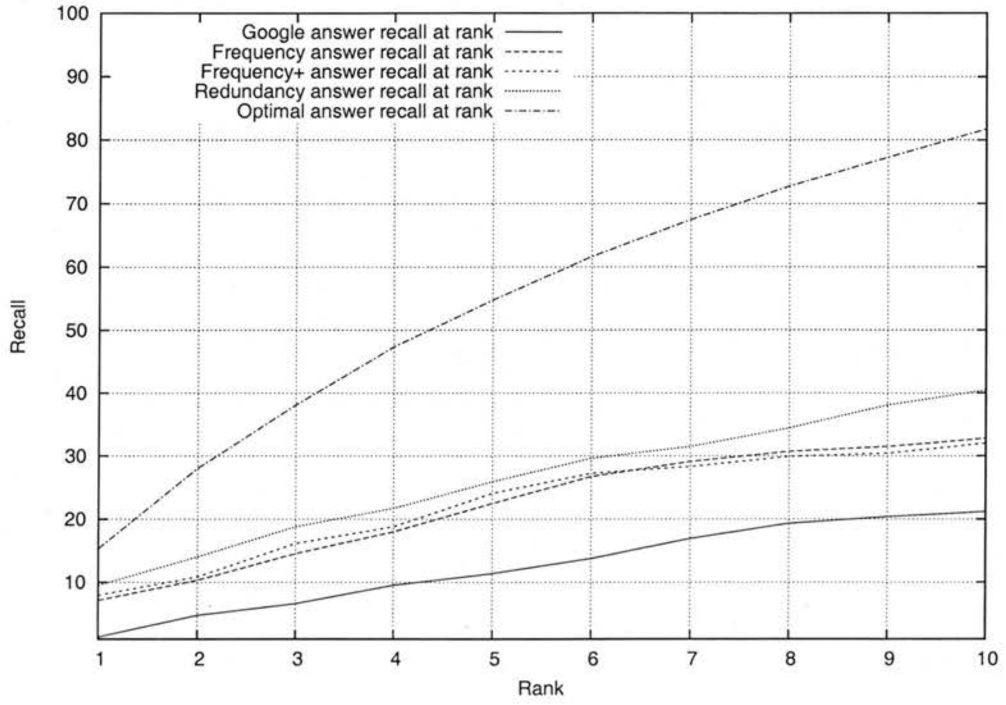


Figure 6.7: Answer recall at rank for answer candidates. Satisfaction group: EXACT-STRONG.

From error analysis, it appeared that the third strategy was efficient at ranking satisfying nodes which originally had a low frequency (and thus were neglected by the frequency-based strategy), but entailed a high frequency node. At rank 10 for LENIENT answers, this was the case for 47 nodes. The nodes provided at rank 1 by the third strategy were also more specific than the other strategies, which was expected because entailment was based on stem overlap. The third strategy simply tended to provide nodes representing candidates with a longer span, e.g. *Who invented the instant Polaroid camera? Edwin Land* (third strategy) vs *Land* (frequency-based strategies). Overlap has proved to be useful in previous work [Brill et al., 2001]. However, here, all the relationships were checked to verify that the considered node was not a singleton, including the new relationship, aggregation, based on term co-occurrences. When omitting aggregation inference, the third strategy drops from a MRR of 0.72 to 0.68 (at rank 10, for EXACT-STRONG answers). This is difference is not statistically significant, although, without aggregation, the performance of the third strategy is not so significantly higher ($p < 0.02$) than the first strategy based on frequency only.

Table 6.10 represents the answer recall at rank 10 for EXACT-STRONG answer candidates per question type. Overall, whether re-ranking was based on frequency only or more fusion-based features, factoid questions had a better coverage than definition questions.

Table 6.10: EXACT-STRONG answer recalls at rank 10 per question type for strategy 1 (S1) and 3 (S3).

Type (# questions)	**Total #**	**S3 - #**	**S3 - %**	**S1 - #**	**S1 - %**
Total (50)	378	153	40.47	124	32.80
FACT-PERS (9)	61	34	55.73	26	42.62
FACT-LOC (13)	45	24	53.33	20	44.44
FACT-TIME (5)	11	5	45.45	4	36.36
DEF-BIO (3)	13	5	38.46	4	30.76
FACT (5)	47	17	36.17	14	29.78
DEF (15)	201	68	33.83	56	27.86

To summarize, fusion-based re-ranking, taking into account not only frequency, but also overlap-based specificity and aggregation relationships, significantly outperforms a re-ranking strategy based on frequency only. I described a heuristic that was designed on the training corpus, and applied to the test corpus. In the next subsections, I present results for the same heuristic applied this time to incremental modeling, and also assess the value of relationships inferred from WordNet. In Section 6.4, in order to go beyond the heuristic devised here, I report an experiment using fusion-based features for machine learning.

6.2.4 Value of WordNet for Open Domain Fusion-based QA

In order to assess the value of WordNet-based inference, for each question, and thus each graph model, I selected the top 10 answer nodes produced by the previous strategy based on a fusion-based heuristic (the best-performing third strategy), as well as the question nodes available, to generate a new model where inference this time used not only shallow computations of entailment, inclusion and aggregation but also WordNet lookups to draw graph edges (Table 6.1). The main motivation for this incremental modeling was to reduce the number of candidate nodes per graph (from about 100 to 10) in order to facilitate human error analysis.

On these new models, I applied the third re-ranking strategy based on frequency, redundancy and specificity scoring of nodes. I compare two views for this heuristic:

1. A first view produces a ranked list of nodes that were scored on the basis of shallow and WordNet inferred relationships (**shallow+WordNet** fusion). Thus, this time entailment relationships known from WordNet were included to compute a score for the entailment branch, and a WordNet equivalence with a question node, as well as simple overlap, was penalizing.

2. The second view produces a ranked list of nodes that were scored on the basis of shallow relationships only, i.e. edges inferred from WordNet were not taken into account (**shallow** fusion).

It is important to note that because this time models were generated from 10 candidate nodes only, the topology of the graphs changed slightly. Fewer edges could be drawn overall. For all questions, graph models fusing all the answer candidates generated 10312 edges representing relationships. For all questions, graph models based on the previous top-10 candidate nodes generated 757 edges only, hence affecting the computation of redundancy and specificity scores. For all questions (50 graph models), WordNet generated 191 (20 equivalence edges, 171 inclusion/entailment edges) new relationships, i.e. about a quarter of all the relationships (N=757).

Table 6.11 compares ranking scores for these two new views. Because both were generated from the top 10 nodes of the same re-ranking strategy previously described, scores at rank 10 are the same. Merging precision and exactness are also the same as given in Table 6.9 listing the previous strategy's scores.

The view produced from both shallow and WordNet relationships (**shallow+WordNet** fusion) obtained a MRR of 0.72, an answer recall at rank 1 of 9.78% and a question recall at rank 1 of 64%. These scores are very close to the scores of the strategy based on all candidates and shallow fusion (strategy 3, with respective scores 0.72, 9.52% and 62%). On the other hand, applying exactly the same heuristic, based on **shallow** fusion only, actually increased the scores slightly.

Table 6.11: Re-ranking scores for strategies based on shallow versus shallow and WordNet fusion at rank 10 after re-modeling. Figures are computed over distinct answer candidate strings.

ES: EXACT-STRONG, E: EXACT, L:LENIENT

AR: answer recall, QR: question recall

Strategy	Shallow+WordNet			Shallow		
Group	ES	E	L	ES	E	L
MRR	0.72	0.72	0.79	0.75	0.76	0.79
AR - rank 1	9.78	9.13	5.73	10.58	9.87	6.01
AR - rank 10	40.47	38.51	32.52	40.47	38.51	32.52
QR - rank 1	64	64	68	68	68	70
QR - rank 10	92	92	96	92	92	96

The second view obtained a MRR of 0.75, an answer recall at rank 1 of 10.58% and a question recall at rank 1 of 68%. However, after testing, the difference between this re-ranking from top-10 shallow modeling, and the original re-ranking based on a full model, was not statistically significant ($p > 0.05$). Figures 6.8 and 6.9 respectively compare question recalls and answer recalls, and reflects this conclusion. Apart from a few variations at ranks 1-2, using WordNet relationships does not make a significant difference for re-ranking using this specific heuristic.

Because this specific heuristic was manually devised on a training corpus for which models were generated from shallow fusion, and then applied to the test corpus, it is not clear whether this heuristic was simply just specific to shallow fusion.

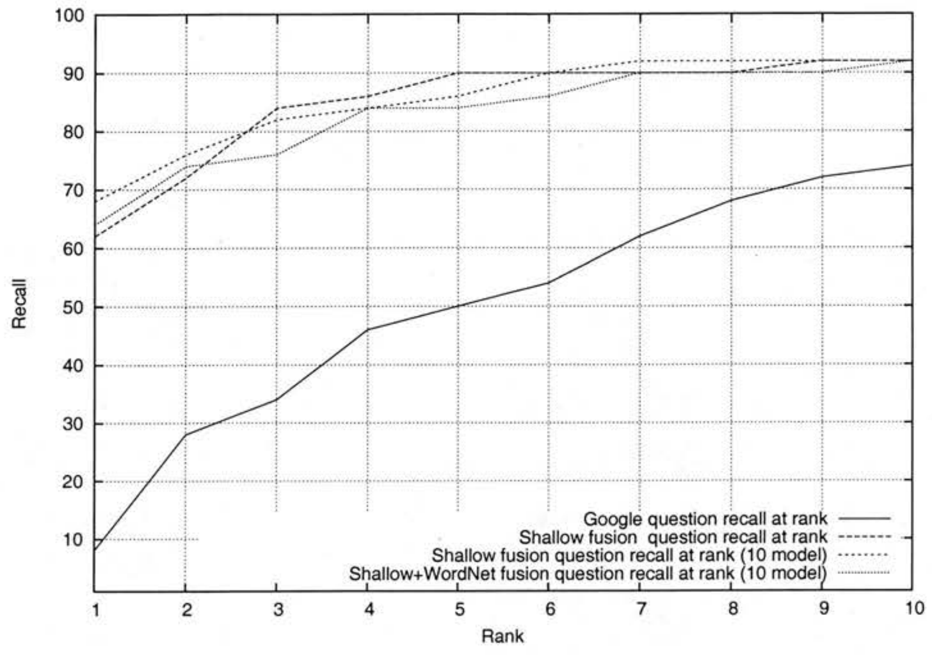


Figure 6.8: Question recalls at rank for EXACT-STRONG answer candidates.

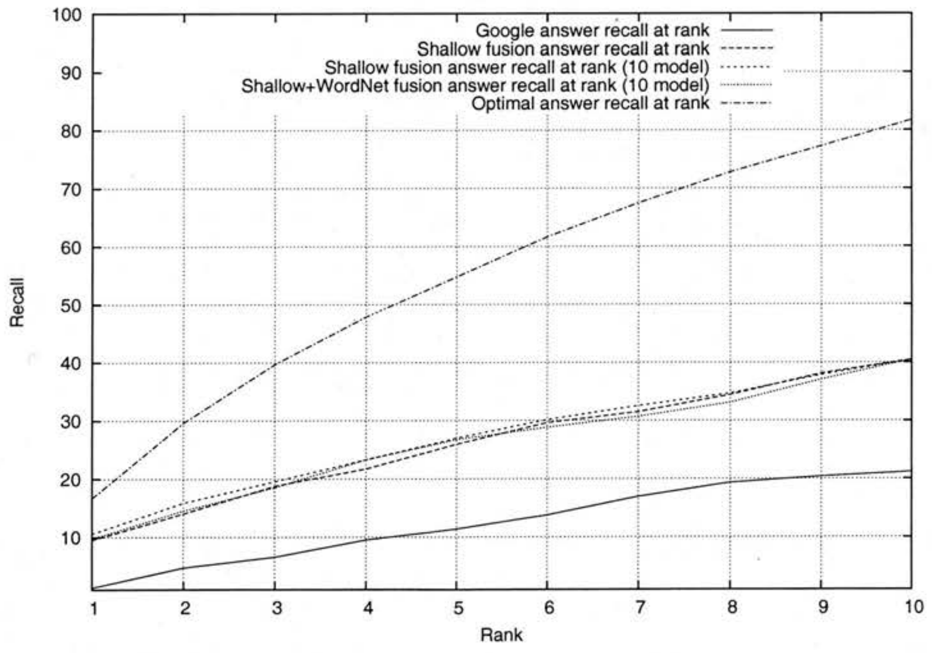


Figure 6.9: Answer recalls at rank for EXACT-STRONG answer candidates.

Also, the idea of manually devising a heuristic from models is first not very appealing, and second actually difficult on large graph models. In the previous subsections, I showed that a simple heuristic based on a fusion-based approximation of redundancy and specificity outperformed frequency-based re-ranking, hence demonstrating that further comparison between answer candidates significantly improves QA re-ranking performance. I will show in Section 6.4 that graph modeling provides relevant features to generate reliable answer classifiers using machine learning on a training corpus.

6.2.5 Discussion

Figure 6.12 summarizes the main EXACT-STRONG scores for each view assessed in this section.

Table 6.12: EXACT-STRONG score comparison for views producing list of re-ranked nodes.

AR: answer recall, QR: question recall

PMP: positive merging precision, PME: positive merging exactness

	MRR	QR - rank 1	AR - rank 1	F-Measure	PMP	PME
Google	0.27	8	1.32	17.67	100	81.95
Frequency	0.59	44	7.14	25.77	98.14	81.62
Frequency+	0.63	50	7.93	25.36	98.25	81.37
Redundancy	0.72	62	9.52	32.90	99.75	83.41
WordNet	0.72	64	9.78	32.90	99.75	83.41
Shallow	0.75	68	9.87	32.90	99.75	83.41
Optimal	1	100	16.66	65.74	98.79	95.15

The merging of answer candidates for satisfying nodes is overall precise (close to 100%) and exact (83.41% positive exactness for the redundancy-base strategy). The small variations are essentially due to differences in re-ranking, with some strategies top-ranking more precise nodes than others (e.g. Google).

More interestingly, the table shows that answer recall tends to progress much more slowly than the MRR or the question recall (both based on the rank of the *first* satisfying answer). While state of the art evaluations such as TREC QA, especially for factoids, essentially focus on the MRR or the question recall at rank 1, the results presented here show that a very high MRR does not reflect the actual performance for *multiple* answers. For instance, the shallow view, based on a subset of 10 nodes, has a MRR of 0.75, however barely more than a third (32.90%) of all the answer candidates were found at rank 10. This is half of what the optimal strategy could find.

When assessing and comparing QA systems, it would be essential to distinguish what is considered the central task: to answer the question, or not only to answer the question but also to provide a good overview of what all the possible answers can be in the given corpus. In TREC QA for factoids, only the first answer is considered, therefore privileging a quiz-like task, in which multiple answers are not so important: what matters is to correctly answer the question. However, there is also an audience for systems that are concerned with multiple answers. In the medical domain for instance, professional users are more interested in a wide panel of possible answers that are supported by good evidence (thus abandoning the binary scale of correctness), rather than a single answer. In the perspective of such users, it is essential to have systems that can handle multiple answers, and that are evaluated with measures, such as answer recall, taking into account answer multiplicity.

Table 6.12 shows that fusion, even based on shallow computations of equivalence, inclusion, entailment and aggregation, not only improves the MRR and the question recall at rank, but also answer recall.

In the next section, I focus on re-ranking clusters of nodes. Clustering allows the grouping of nodes that are considered to relate to the same answer (e.g. as *Agra* and *India*), thus providing the user with not only a more structured but also a more answer-efficient output than a flat list of single nodes.

6.3 Re-ranking Clusters of Nodes

This section covers the views that produce a ranked list of clusters of nodes, rather than single nodes. The objective of node clustering is to group together nodes that considered to be related in some way. I explore two types of clustering. The first strategy produces a list of clusters in which nodes vary in granularity. The second strategy generates clusters of nodes that are related by entailment, inclusion or aggregation, and basically represent the connected components of a graph model, i.e. its maximal subgraphs.

Both views are based on incremental modeling. The first model is generated with shallow fusion and the redundancy-based strategy (strategy 3, Subsection 6.2.3) is used to select a subset of 10 answer nodes. Those answer nodes are then fused again in a new model, with the question nodes, using shallow fusion. Each view extracts different types of clusters and re-rank them into the final list.

For each view, I provide the measures previously used, to assess the re-ranking strategy and the ability to recognize multiple answers, and characterize clustering in terms of precision and exactness.

6.3.1 Entailment Clusters

This subsection reports evaluation results on a view producing *entailment clusters*. As mentioned earlier, a node represents the merging of answer candidates. For instance, the node *Pierre Curie* (*Who discovered radium?*) actually merged 3 candidates (3 occurrences of *Pierre Curie*). An *entailment cluster* is built starting from a singleton containing a leaf node (a leaf in that it does not include any other node) to which the leaf's parent nodes are added. Only parent nodes by direct entailment and transitive entailment, as well as their neighbours connected by equivalence, are considered.

Figure 6.10 represents the shallow graph model generated from the top-10 nodes previously re-ranked on the basis of redundancy using incremental modeling (Subsection 6.2.4). Taking the example of the leaf node *Pierre Curie*, the entailed parents are *Curie* and *Pierre*. The entailment cluster corresponding to Pierre Curie is thus $\{\{$*Pierre Curie*$\}$, $\{$*Pierre*$\}$, $\{$*Curie*$\}\}$ (string-matching answer candidates merged into the same node are skipped to ease reading).

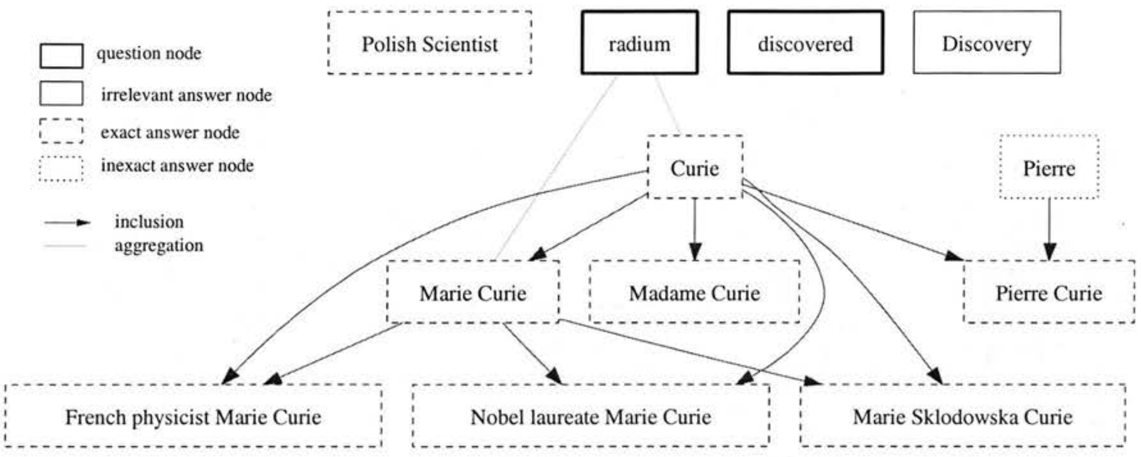


Figure 6.10: *Who discovered radium?* - Shallow graph model based on the top-10 nodes from fusion-based re-ranking.

The algorithm starts with an empty list of clusters *CL*, a list *NL* of 10 re-ranked nodes and the corresponding graph model. A empty set is also created to keep track of seen nodes.

For each answer node N_i of the ranked list *NL*, such that N_i has not been seen before, a new cluster C_k is created with N_i as single member. Each answer node N_j connected directly or transitively to N_i by inclusion or equivalence (i.e. N_i either entails or is equivalent to N_j) is added to C_k, as well as to the set of seen nodes. The new cluster C_k is then added to the list of clusters *CL*.

The list of clusters *CL* is finally sorted so as to respect the original node re-ranking strategy based on redundancy: The rank of a cluster is the best rank found among its members. Note that members of each cluster are sorted by granularity, i.e. the most specific node will appear first.

For the question *Who discovered radium?*, the re-ranked list of nodes was:

1. {Curie}
2. {Marie Curie}
3. {Pierre Curie}
4. {French physicist Marie Curie}
5. {Nobel laureate Marie Curie}
6. {Marie Sklodowska Curie}
7. {Madame Curie}
8. {Pierre}
9. {Discovery}
10. {Polish Scientist}

The following list of entailment clusters was produced:

1. { {French physicist Marie Curie}; {Marie Curie}; {Curie} }
2. { {Nobel laureate Marie Curie}; {Marie Curie}; {Curie} }
3. { {Marie Sklodowska Curie}; {Curie} }
4. { {Pierre Curie}; {Curie}; {Pierre} }
5. { {Madame Curie}; {Curie} }
6. { {Discovery} }
7. { {Polish Scientist} }

The motivation behind entailment clustering is to group nodes by granularity, from specific to general. Clusters may overlap as different leaf nodes can share the same parents, as the *Curie* example above. The advantage of starting from leaf nodes (as opposed to root nodes in inclusion-based clustering) is to immediately specify potentially ambiguous parent nodes. For instance, *Curie* is ambiguous because the surname refers to two distinct people. The obvious limitation is that if two leaves were not identified as equivalent in the model, they will generate distinct clusters. This is actually a difficult case since *French physicist Marie Curie*, *Nobel laureate Marie Curie*, *Marie Sklodowska Curie*, *Madame Curie* and *Polish Scientist* are not strictly equivalent. They do, however, co-refer to the same person.

Table 6.13 presents the results of the evaluation of entailment clusters.

Table 6.13: Re-ranking scores for entailment clusters produced from a list of 10 nodes previously ranked using shallow fusion. Figures are computed over distinct answer candidate strings. The maximum number of clusters per question was 9.

ES: EXACT-STRONG, E: EXACT, L:LENIENT

AR: answer recall, QR: question recall

NPV: Negative Predictive Value

Group	ES	E	L
MRR	0.79	0.80	0.84
AR - rank 1	17.98	17.03	14.18
AR - rank 9	40.47	38.51	32.52
# Satisfying Answers Found	153 (N = 378)	156 (N = 405)	227 (N = 698)
QR - rank 1	72	74	78
QR - rank 10	92	92	96
# Questions	50	50	50
Clustering precision	26.16	26.66	36.97
Positive clustering precision	75.26	74.90	89.82
Clustering exactness	31.82		
Positive clustering exactness	77.40		
# Satisfying Clusters Found	115	118	136

Note that the truth table measures, e.g. the F-Measure, are the same as previously reported in Table 6.9 since they are computed from the exact same list of nodes (top-10). Only entailment clustering increased the answer recall at rank 1 (17.98% against 10.58% for the unclustered list, Table 6.11), as well as question recall at rank 1 (72% against 68%) for EXACT-STRONG answers. The MRR for the same group also increased from 0.75 to 0.79.

Figures 6.11 and 6.12 present the question recall at rank, and the answer recall at rank for EXACT-STRONG answers from entailment clusters. On average per question, from a list 10 nodes, 6.2 clusters were generated.

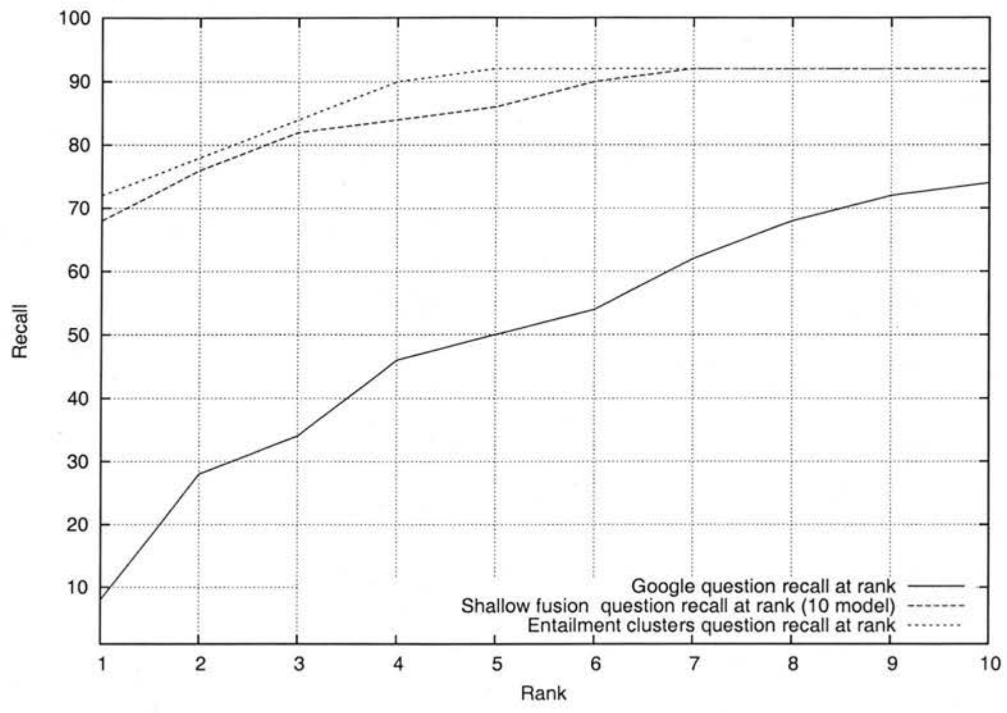


Figure 6.11: Question recall at rank for entailment clusters against re-ranking from a shallow 10-node model and Google ordering. Satisfaction group: EXACT-STRONG.

The maximum rank (i.e. number of clusters) was 9, which is why the answer rank in Table 6.13 stops at 9 rather than 10. At rank 1, entailment clusters achieve a better answer recall on EXACT-STRONG answers than the optimal strategy for single nodes.

The positive clustering precision decreased from 99.75% to 75.26% for EXACT-STRONG answers, meaning that the clusters are slightly less precise than nodes: On average, a quarter of all the answer candidates merged in the nodes members of a cluster are unsatisfying on an EXACT-STRONG level. This score increased slightly (89.82%) when taking into account LENIENT answers, meaning that 10% of the answer candidates in a cluster are really irrelevant. In terms of exactness, satisfying clusters were also on average less exact: 77.40% exactness against 83.41% for nodes (Note that the approximate optimal exactness for a node based on stem-matching is 95.15%.) Those figures actually apply to rather small clusters. On average, a cluster consisted of 1.85 answer candidates, and 2.34 for satisfying clusters. It is not surprising that satisfying clusters contain more candidates than unsatisfying clusters since the basic hypothesis behind fusion is that questions may expect multiple answers and thus satisfying answers may be supported by a larger network in graph models.

It is, overall, expected that clusters, presenting more content than nodes, would naturally improve question recall and answer recall. However, as indicated by the average size of a cluster, as well as the positive clustering precision and exactness, clusters offer a level of answering that is precise and exact, while attempting to organize together related answers that were found in different documents.

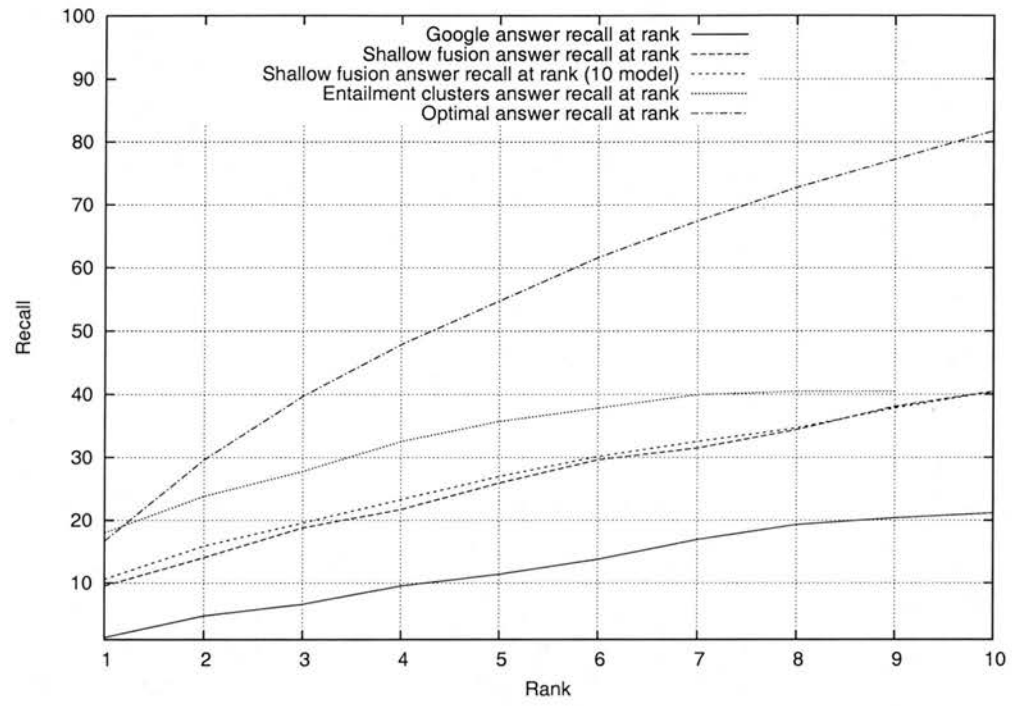


Figure 6.12: Answer recall at rank on answer candidates for entailment clusters against re-ranking from a shallow 10-node model, Google ordering and optimal node re-ranking. Satisfaction group: EXACT-STRONG.

In terms of error analysis, I looked at two types of errors: (1) the lack of clustering between nodes that should have been on the same entailment branch, and (2) the presence of irrelevant nodes in entailment clusters, which penalized positive precision clustering.

With respect to inner sorting, from specific to general, because entailment in graph models was essentially based on stem overlap, clusters were effectively dealt with (e.g. *When was Hiroshima bombed?* {*August 6, 1945*; *1945*; *August 6*}).

However, a strong limitation was the absence of connections between nodes that should have been on the same entailment branch, due to the lack of recognition of :

- synonyms (*What is myopia? nearsightedness* versus *short-sightedness*).
- co-referring clauses, as seen above with the example of *French physicist Marie Curie* and *Nobel laureate Marie Curie*.
- inclusive relationships such as hypernymy (*mosquitoes* and *insects*) or meronym (*Tharsis plateau* on *Mars*).

While some of these relationships could have been covered by WordNet, a run producing entailment clusters extracted from a model using WordNet lookups did not improve the quality of views. Overall WordNet views generated less precise and less exact clusters (69.11% positive clustering precision against 75.26% without WordNet for the EXACT-STRONG group, 71.85% positive clustering exactness against 77.40% without WordNet). More connections were made and thus clusters were larger indeed (average size of 2.9 per satisfying cluster on EXACT-STRONG level). But increasing the size of the clusters did not improve the MRR (0.78 against 0.79 without WordNet), although it did increase answer recall at rank 1 (18.51% against 17.98%) but not significantly.

Although WordNet helped identifying relationships such as *insect* $\leftrightarrow$ *mosquito*, it is not exhaustive enough to cover for instance *Mars* $\leftrightarrow$ *Tharsis plateau*. Also, complex descriptives clauses, such as *small flying night bugs* (*What do bats eat?*) that appeared in the top 10 nodes but for which inner keywords projected into nodes (e.g. *bugs*) did not make in the top-10, were difficult to connect. If the node *bugs* had been present in the top-10 list of nodes, it would have been connected to *insect*, and, consequently, forming a well organized subgraph to answer *What do bats eat?*. Figure 6.13 shows the graph model generated for that question using WordNet relationships.

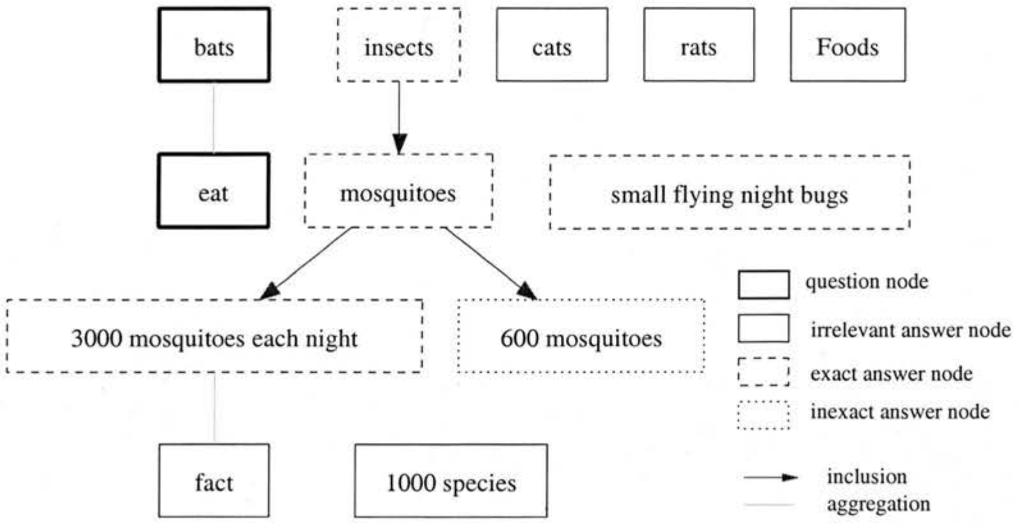


Figure 6.13: *What do bats eat?* - Graph model using WordNet relationships.

The output clusters were as follows:

1. { {3000 mosquitoes each night}; {mosquitoes}; {insects} }
2. { {6 00 mosquitoes}; {mosquitoes}; {insects} }
3. { {cats} }
4. { {rats} }
5. { {food, foods} }
6. { {facts, fact} }
6. { {small flying night bugs} }
6. { {1000 species} }

On the other hand, entailment clusters based on overlap could also contain somewhat inexact and unnecessary information, such as first names for person names (*Pierre Curie* and *Pierre*; or simply irrelevant nodes due to erroneous projection of chunk parts (*Ulan Bator*; *Bator*); or irrelevant nodes that were somewhat related but inadequate to serve as an answer (*What is amoxicillin?* {*drug class*; *drugs, drug*; *class*}, cluster for which only the node *drugs, drug* was considered exact and well supported).

On average, each question was answered with 2.38 satisfying clusters on the STRONG-EXACT level, with a minimum of 1 satisfying cluster for 19 questions, and a maximum of 7 satisfying clusters for a single question. 31 questions had more than one satisfying cluster. Among those questions with multiple satisfying clusters, multiplicity represented:

- alternative answers (10 questions: 6 FACT-PERS, 2 FACT, 2 DEF):
  Examples: *What is peyote?* a cactus, or a beading term (peyote stitch). *Name a German philosopher.* Nietzsche, Kant, Heidegger.

- aggregated answers, coordination or illustration of different aspects of an answer (11 questions: 7 DEF, 1 DEF-BIO, 1 FACT, 1 FACT-PERS, 1 FACT-TIME):
  Examples: *What are capers?* Capparis Spinosa, unopened green flower buds. *Who was Galileo?* astronomer, Italian physicist.

- answers varying in granularity not clustered together (8 questions: 3 DEF, 3 FACT-LOC, 1 FACT, 1 FACT-PERS):
  Example: *Where is the volcano Olympus Mons located?.* Tharsis Plateau and Mars were not clustered.

- equivalent answers that had not been properly clustered (9 questions, 6 DEF, 1 FACT-LOC, 1 FACT-PERS, 1 FACT):
  Examples: *What is sodium chloride?* common table salt, regular table salt. *Who discovered radium?* Marie Sklodowska Curie, Nobel laureate Marie Curie.

- one or more inexact answers entailing an exact answer listed in another cluster generated distinct clusters that were actually redundant considering the exact answer (11 questions: 5 DEF, 3 FACT-LOC, 2 FACT-PERS, 1 FACT):
  Examples: *What is naproxen?* drug guide $\leftrightarrow$ drug, drug administration $\leftrightarrow$ drug. *What is the currency used in China?* leftover yuan $\leftrightarrow$ yuan.

Note that questions could have distinct cases of multiplicity, for instance equivalent answers not properly clustered, as well as alternative answers.

6.3.2 Connected Components

While the previous view focused on granularity by generating entailment clusters, I now describe a view that produces clusters mapping to the *connected components* of the graph model. A *connected component* is a maximal subgraph of the graph, i.e. a set of nodes that are connected by equivalence, inclusion, entailment, and/or aggregation. Each connected component represents a partition of the graph, as shown on Figure 6.14. The subgraph can be a clique (a complete subgraph), a singleton (a single member partition) or any maximal subgraph such that a path can be found from any node member to any other node member by following one or more edges.

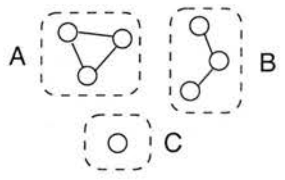


Figure 6.14: Three connected components of a graph: a clique (A), a maximal subgraph (B), a singleton (C).

The algorithm starts with an empty list of clusters *CL*, a list *NL* of 10 re-ranked nodes and the corresponding graph model. A empty set is also created to keep track of seen nodes.

For each answer node N_i of the ranked list *NL*, such that N_i has not been seen before, a new cluster C_k is created. The connected component of N_i is retrieved from the graph model, such that each answer node member of the connected component is connected directly or transitively to N_i by equivalence, inclusion, entailment or aggregation. The component members are added to C_k, as well as to the set of seen nodes. The new cluster C_k is then added to the list of clusters *CL*. (Note that question nodes are not considered for clustering.)

As for entailment clusters, the list of clusters *CL* is then sorted so as to respect the original node re-ranking, i.e. the rank of the cluster is the best rank found among its members. Members of each cluster are also listed following the original ranking order.

Using the same graph model as example (Figure 6.10, the following list of connected components was produced:

1. { {Curie}; {Marie Curie}; {Pierre Curie}; {French physicist Marie Curie}; {Nobel laureate Marie Curie}; {Marie Sklodowska Curie}; {Madame Curie}; {Pierre} }
2. { {Discovery} }
3. { {Polish Scientist} }

The objective of such clustering is to group together related answers, not only answers that vary in phrasing or in granularity (entailment-based clustering), but also answers that could be aggregated, thereby mapping each cluster to an alternative answer. However, the task of identifying alternative answers is not trivial.

For instance, snippets primarily indicate that Marie Curie was the main discoverer of radium. A few other snippets indicate that it was a joint discovery with her husband. Should *Marie Curie* and *Pierre Curie* be considered as aggregated answers or alternative answers? Also, the answer candidate *Polish scientist* is so unspecific that is not clear whether it applies to the wife or the husband. Contextually it refers to *Marie Curie*, now in another snippet she is referred to as a *French physicist*. Marie Curie was born Polish, but became a naturalized French citizen later in her life. However, this is not mentioned in any of the snippets found for the question. Should we expect QA systems to identify a local alternative, *Polish* vs *French*, that is not actually a real contradiction, or should this local difference be seen as minor if not irrelevant to the question?

Another such difficulty occurs with answer candidates varying in granularity. For example, should we consider *3000 mosquitoes each night* and *600 mosquitoes an hour* to be alternative answers to *What do bats eat?*, or should we group them altogether under a common denominator, for instance *mosquitoes* or *insects*?

Clusters add some flexibility to question answering by organizing related answers together. However, it seems to me difficult to assess in further details this "relatedness". While there exist some clear-cut examples, for instance, *What is vertigo?* may have obvious alternative answers, e.g. the medical symptom as opposed to the movie, the diversity of answers is such that identifying very precisely the type of relationships between each candidate can be a hard task, even for human assessors.

For this experiment, I will first review the evaluation scores assessing the exactness and precision of clustering, as well as the improvement in terms of answer re-ranking. I will then characterize cluster multiplicity, and how it can be mapped to what I call multiple answer referents. For this evaluation of multiple answer referents, I will not consider the relationships intra-clusters, because of the difficulties mentioned above. I will instead assess that candidates in the same cluster indeed co-refer to the same answer referent.

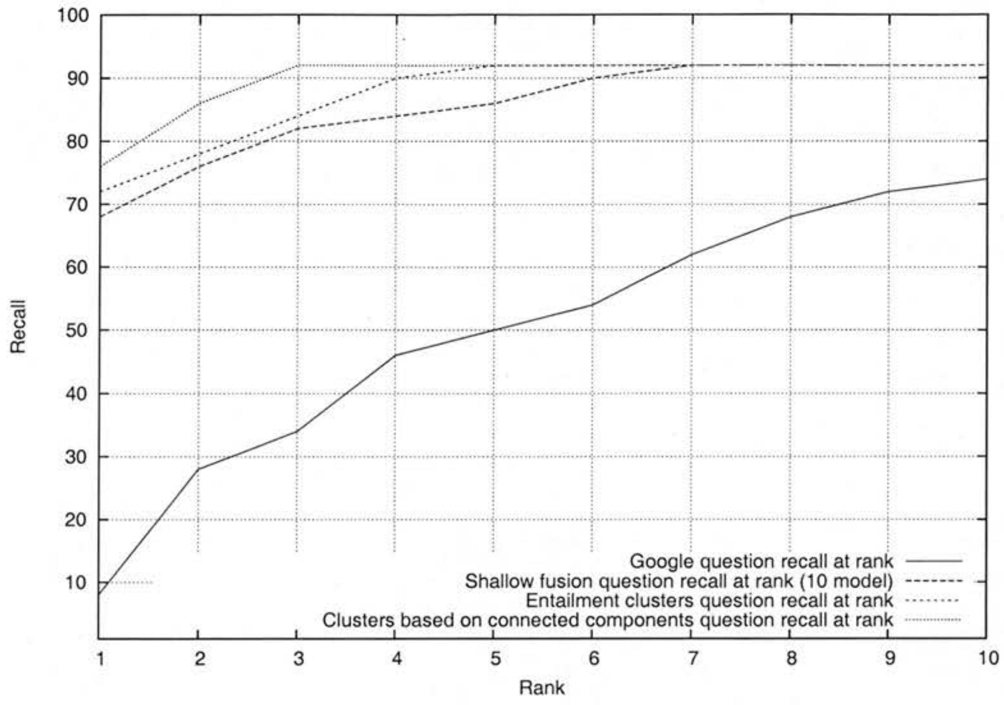


Figure 6.15: Question recall at rank for clusters based on connected components against entailment clusters, re-ranking from a shallow 10-node model and Google ordering. Satisfaction group: EXACT-STRONG.

Figures 6.15 and 6.16 present the question recall at rank, and the answer recall at rank for EXACT-STRONG answers from clusters based on connected components. On average per question, from a list of 10 nodes, 4.2 clusters were generated. The maximum rank (i.e. number of clusters) was 7. At rank 1, clusters based on connected components, as entailment clusters, achieve a better answer recall on EXACT-STRONG answers than the optimal strategy for nodes. (This is a normal result since clusters are groups of nodes, thus providing more answer candidates at rank 1 than a strategy providing a single node at rank.)

As expected, the average cluster size was higher than the average size for entailment clusters, with 2.6 answer candidates per cluster. Satisfying (EXACT-STRONG group) clusters were also larger, and contained 4.5 answer candidates on average (4.3 for satisfying EXACT clusters, 4.2 for satisfying LENIENT clusters).

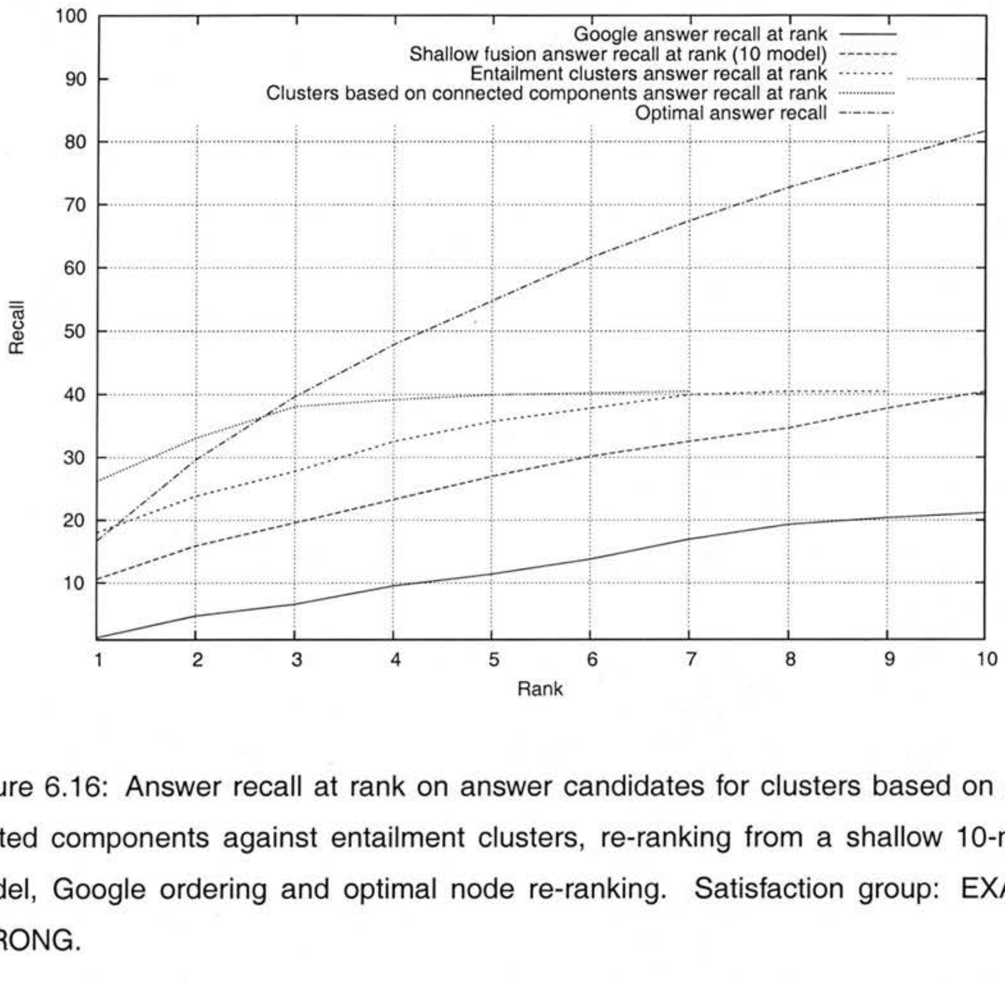


Figure 6.16: Answer recall at rank on answer candidates for clusters based on connected components against entailment clusters, re-ranking from a shallow 10-node model, Google ordering and optimal node re-ranking. Satisfaction group: EXACT-STRONG.

Table 6.14 reports the evaluation results for clusters based on connected components. Because connected components produced larger clusters than entailment clusters, it was also expected that the MRR (0.83 on EXACT-STRONG answers, against 0.79 for entailment clusters), the answer recall at rank 1 (26.19% against 17.98%), and the question recall at rank 1 (76% against 72%) would increase.

Positive clustering precision dropped down from 75.26% to 59.23% for EXACT-STRONG answers, meaning that in an average satisfying cluster, containing about 4-5 candidates, about 3 of these candidates were satisfying on EXACT-STRONG level, but about 2 were not satisfying. The EXACT positive clustering precision shows a similar distribution. The clustering precision for the LENIENT group (76.01%) indicates that, on average, a bit less than a quarter of a cluster is actually irrelevant. Positive clustering exactness also dropped down from 77.0% to 65.62%.

This is an expected trade-off that larger clusters improve answer recall but are, on average, less precise and less exact. Still, most of the members of a cluster judged satisfying are relevant, although possibly inexact.

Table 6.14: Re-ranking scores for clusters based on connected components produced from a list of 10 nodes previously ranked using shallow fusion. Figures are computed over distinct answer candidate strings.

ES: EXACT-STRONG, E: EXACT, L:LENIENT

AR: answer recall, QR: question recall

NPV: Negative Predictive Value

Group	ES	E	L
MRR	0.83	0.84	0.88
AR - rank 1	26.19	24.69	21.63
AR - rank 7	40.47	38.51	32.52
# Satisfying Answers	378	405	698
QR - rank 1	76	78	82
QR - rank 10	92	92	96
# Questions	50	50	50
Clustering precision	21.46	22.15	30.06
Positive clustering precision	59.23	59.77	76.01
Clustering exactness	26.11		
Positive clustering exactness	65.62		

On average, each question was answered with 1.42 satisfying clusters on the STRONG-EXACT level, with 1 satisfying cluster for 27 questions, 2 satisfying clusters for 13 questions, and 3 satisfying clusters for 6 questions (4 questions did not have a satisfying answer within the top 10 nodes previously selected). 19 questions had thus more than one satisfying cluster, and among those questions, multiplicity represented:

- alternative answers (7 questions: 4 FACT-PERS, 2 DEF, 1 FACT).
- aggregated answers, coordination or illustration of different aspects of an answer (5 questions: 4 DEF, 1 FACT-TIME).
- answers varying in granularity not clustered together (6 questions: 4 DEF, 1 FACT-LOC, 1 FACT).
- equivalent answers that had not been properly clustered (6 questions, 2 DEF, 2 FACT-LOC, 1 FACT-PERS, 1 FACT).

(Note that several error types could occur within the same question.) Over the 50 questions, within the top 10 nodes, 4 questions had no answer on the EXACT-STRONG level. 39 questions expected only one answer referent, although this referent could be expressed by different answer candidates. For instance, *manic-depression*, *manic depressive illness* and *diagnosis* are distinct EXACT-STRONG answer candidates, characterizing a single answer referent for the question *What is bipolar disorder?* 6 questions expected 2 alternative answer referents, and 1 question expected 3: *Name a German philosopher.* had 3 answer referents: (1) *Friedrich Nietzsche, Nietzsche*, (2) *Kant* and (3) *Martin Heidegger*. I considered multiple answer candidates such as *Pierre Curie* and *Marie Curie*, or *3000 mosquitoes each night* and *600 mosquitoes an hour*, as pointing to a single referent: the collective answer *Curie* for *Who discovered radium?*, and the referent *insects* for *What do bats eat?*

Table 6.15 shows for how many questions satisfying clusters exactly mapped to alternative answer referents. Two questions had a match and a mismatch at the same time. *What does the word fortnight mean?* expected two alternatives: {*two weeks*, *fourteen days*} and {*10 days*} (a snippet explicitly mentioned that *fortnight* could be understood as 10 days). Two satisfying clusters were found, one for *two weeks*, another for *10 days*, but *fourteen days* was incorrectly clustered with *10 days* (failure to identify *fourteen* as an alternative to *10*). The second case was the question *What is peyote?*, which expected two answer referents, one for the *cactus* and another for the *beading term*. The view provided 3 satisfying clusters, with an extra singleton, *stitch*, which modeling failed to identify as related to *beading* terminology.

Table 6.15: Mapping of multiple satisfying clusters to multiple answer referents (within the top-10 nodes) with clustering based on connected components. Level: EXACT-STRONG.

# expected answer referents	3	2	1	Total	%
# questions	1	6	39	46	100
# questions with matched referents	1	3	26	30	65.2
# questions with mismatched referents	0	1	13	14	30.4
# questions with mis/matched referents	0	2	0	2	4.4

Most errors (16 questions) were due to a lack of connectivity (lack of inference of inclusion, equivalence and/or aggregation) rather than an excessive connectivity due to ambiguity. (In a single case: *Who is the governor of Colorado? Bill Ritter* and *Bill Owens* were clustered together because of the firstname overlap.)

However, overall, looking at both satisfying and unsatisfying clusters, the number of output clusters per question does not correlate with the actual number of distinct referents. There are still too many unsatisfying clusters output present in the final list of clusters (139 over a total 210 leading to a precision of 33.8% only for clusters, and 27.71% for candidates on EXACT-STRONG level, Table 6.9). This is also a limitation of re-ranking which does not systematically attempt to discard unsatisfying clusters, but rather rank them lower.

6.3.3 Discussion

Figure 6.16 summarizes the scores by clustering views for each group of answer.

It is, again, not surprising that larger clusters (connected components) increase scores based on the rank of the first satisfying cluster (MRR and question recall at rank 1). It indicates the limitation of such measures when attempting to handle answer multiplicity. Positive clustering precision for entailment clusters on the LENIENT level indicates that approximately 90% of the answer candidates merged into a cluster are relevant, although not necessarily exact (77.40% in positive clustering exactness). As expected connected components are much less precise and exact, but still about 2/3 of a cluster consists of relevant information.

However, with a view based on connected components, for 30 questions out of 46, the satisfying clusters that were generated did map to the number of distinct answer referents to a given question (Table 6.16).

Table 6.16: Score comparison for views producing list of re-ranked clusters.

AR: answer recall, QR: question recall

PCP: positive clustering precision, PCE: positive clustering exactness

EXACT-STRONG answers					
	MRR	QR - rank 1	AR - rank 1	PCP	PCE
Entailment	0.79	72	17.98	75.26	77.40
Connected Comp.	0.83	76	26.19	59.23	65.62
EXACT answers					
Entailment	0.80	74	17.03	74.90	77.40
Connected Comp.	0.84	78	24.69	59.77	65.62
LENIENT answers					
Entailment	0.84	78	14.18	89.82	77.40
Connected Comp.	0.88	82	21.63	76.01	65.62

There is currently no such task in QA as the automated recognition of distinct answer referents for a given question, which would indicate for instance the degree of answer heterogeneity for a given corpus. (The number of distinct referents could vary depending on the corpus.)

The later view based on connected components approximates such a task, but it would benefit from more inner and outer filtering of the output clusters, i.e. by discarding systematically unsatisfying candidates inside a cluster (inner filtering to increase clustering exactness and answer precision), as well as unsatisfying clusters (outer filtering to increase the precision over clusters). In the next chapter, Subsection 7.2.2, I propose to develop a node typing system in order to more accurately identify the different types of nodes, and proceed to such a filtering.

Finally, these two experiments, on entailment clusters and connected components, are based on two distinct clustering algorithms to map graph content into answer clusters. The points to be discussed in comparison with traditional document clustering are the issues of (1) overlapping clusters,(2) the halting criterion, and (3) the similarity function that is used for general text clustering.

Entailment clusters allow overlapping for parent nodes. For example, both clusters starting from *Marie Sklodowska Curie*, and *Pierre Curie*, entail *Curie*. Nodes belonging to different clusters, i.e. parent nodes including distinct children (distinct as in 'not connected by any relationship'), are typically ambiguous nodes, having different instantiations. *Curie* in this context is ambiguous because it can refer to the wife, as well as to the husband. Connected components on the other hand are non-overlapping. *Curie* is then not considered as ambiguous but as a common denominator. For this specific question, it is acceptable. But it is very likely to be a source of mistaken clustering in the case of ambiguous nodes that may not stand for a collective answer, such as *Curie*. In Chapter 7, Subsection 7.2.1, I will argue in further details that this is a problem of incompatible transitivity between some nodes. If we take as an example the following three nodes in the context of the question *Where is Perth?*, *Perth*, entailing *Australia* as well as *Scotland*, entailment clusters would adequately distinguish between two alternative answer referents: *Scotland* versus *Australia*.Connected components would also distinguish between the referents because *Perth* is a question node, and question nodes are not considered during clustering. Now, if the question had been *Name a city that starts with the letter P*, then *Perth*, *Australia* and *Scotland* would have been all clustered together. The issue is that the node *Perth* would stand at the same time for two distinct cities (ambiguity). There are two possible solutions, either (1) disambiguate answer candidate extractions before merging them into nodes, so that, for instance, two nodes are generated for *Perth*, or (2) allow overlapping clustering and disambiguate during clustering.

With respect to the halting criterion, which is usually a self-measure of cluster cohesion in text clustering, the two algorithms described simply enumerate leaf answer nodes, in the case of entailment clusters, and connected components in the second case. Halting is made on the basis of relationship connectivity, i.e. subgraphs of the models, rather than an assessment of the cohesion of the answer cluster.

Similarly, while text clustering is usually based on an approximation of similarity, the algorithms presented here are based on relationships identified between nodes (inclusion, entailment, aggregation).

The clustering methods used here are not directly comparable with Clustering By Committee (CBC) used by [Pantel and Lin, 2002] for QA. Indeed, Pantel et al. used clustering to identify semantic class label, specifically denoting hypernymy in order to filter candidates to definition questions, but they did not generate "answer clusters". Fusion-based clusters are closer to Suffix Tree Clustering [Zamir and Etzioni, 1999] (STC), in that they aim at improving the readability of answers by grouping related candidates together. Now, STC is based on a measure of similarity and aim at improving IR by organizing results into salient topics. The paradigm is slightly different in QA, at least in the task I proposed, where clusters aim at representing alternative answers to a question

Before closing this chapter on answer flexibility, I review a first experiment using machine learning techniques to generate an answer classifier (as opposed to a reranker) trained on graph models.

6.4 Features of Graph Models and Machine Learning

A difficulty encountered with machine learning techniques applied to QA is the low density of answer candidates versus irrelevant candidates. When attempting to build a QA classifier, whose task would be to specify whether a given candidate is a satisfying answer or not, one has to deal with the problem of *sparse data*: The amount of relevant candidates is so small, that, often, the optimal accuracy on training data is achieved when candidates are all classified as irrelevant.

However, this corpus has the particularity of (1) being relatively small, and (2) to have been annotated with an exhaustive list of answers. Table 5.9 showed that EXACT-STRONG answer candidates represented 11.7% of the total number of candidates, and up to 20.5% for LENIENT answers. When merging candidates into nodes, the distribution of relevant nodes is 7.2% (359 relevant nodes over 4963 generated nodes) for the EXACT-STRONG group, and 13.5% for the LENIENT group: The number of positive instances (satisfying nodes) is low but manageable.

I decided to experiment with machine learning to obtain a more general indication of the performance level of fusion-based features, i.e. to assess whether such features could be good enough to be exploited by machine learning techniques. If the results obtained using automated training techniques were comparable or better than those of the heuristic-based re-ranking strategy used earlier, then it would mean that the performance reported in the previous subsections would be expected to generalize well on new questions answered with Google web snippets, or if another type of dataset, it would be possible to use machine learning techniques to build a decent classifier from a small set of annotated data, as it was done here.

The task for machine learning was defined as a binary classification task over the nodes produced for graph models. The classification outcome indicates whether the node is satisfying or not. I performed training and testing for the three satisfaction groups: EXACT-STRONG, EXACT and LENIENT, and on the full testing corpus (50 questions and 4963 output nodes). I compare the trained classifiers' results to the results obtained by re-ranking using shallow fusion, i.e. the third strategy based on redundancy defined in Subsection 6.2.3, which outputs a list of 10 nodes.

The task of classification is a slightly different task from the one of re-ranking. A classifier filters out the nodes classified as unsatisfying, while a re-ranking machine will rank them lower. For comparison, I considered that the top-10 nodes sorted by the re-ranker were considered as satisfying, while the rest of the list had been discarded as irrelevant.

Also, in the previous subsections, I computed answer recall on the basis of distinct answer candidate strings, i.e. how many satisfying answer candidates had been found in the list of 10-top nodes (knowing that a node may have merged several candidates). Because a trained classifier classifies nodes, using features based on their position in the graph (number of neighbours, types of neighbours, incoming and outcoming edges), the answer recall reported here is on the basis of satisfying nodes correctly classified over the total number of output nodes. For comparison, I provide a node-based answer recall and precision as well for the re-ranking system.

6.4.1 Features Associated with Graph Models

For machine learning, three groups of features were computed for each node, with an overall total of 50 features per nodes.

The first group encodes information about the node itself:

- The tag for the node: *CD*, *NN*, *NNP*, *JJ* or *O* (other).
- The size of the bag of stems.
- The frequency of the node, i.e. the number of merged candidates into the node.
- The best candidate rank based on Google order of apparition, varying between 1 and 138 (maximum number of distinct candidates strings for a question).
- The best snippet rank from Google, varying between 1 and 10 (number of snippets per question).

The second group characterizes the question, and encodes relational information based on the node's incoming and outcoming relationships with question nodes:

- The WH-word of the question, among *what/which*, *when*, *where* and *who*, was added as a feature to characterize the question.
- A measure of the overlap with question nodes: The number of question nodes equivalent to, included or entailed by the node.
- The number of question nodes connected by aggregation.

Finally, the third group of features encodes relational information based on the node's incoming and outcoming relationships with other answer nodes. For each node, the following graph partitions were computed:

- The connected component of answer nodes to which the node belongs (by aggregation, inclusion, entailment and equivalence).
- The entailment partition of answer nodes to which the node belongs (the node and its entailed parents).
- The set of answer nodes connected by direct inclusion via lexical head matching.
- The set of answer nodes connected by direct inclusion based on overlap.
- The set of answer nodes connected by direct entailment via lexical head matching.
- The set of answer nodes connected by direct entailment based on overlap.
- The set of answer nodes connected by direct aggregation.

For each set of nodes, the following features were computed (representing overall 7*6 = 42 features for the fourth group):

- Size: the number of nodes in the set.
- Frequency sum: the sum of the frequency of each node member of the set.
- Mean frequency: the mean frequency of each node member of the set.
- Maximum frequency: the highest frequency among node members.
- Ratio node frequency over mean frequency: the frequency of the node considered over the mean frequency of the set.
- Ratio node frequency over maximum frequency: the frequency of the node considered over the highest frequency in the set.

Those last features describe the node's neighbours in terms of connectivity (number of neighbours) and frequencies, as well as an indication of how the node compares to its neighbours. For instance, the node considered can be a low frequency node that entails a set of nodes with a high mean frequency (small ratio node frequency over mean frequency).

Each node is thus considered as a distinct unit, with its own features (e.g. frequency, tag, Google rank), as well as a contextualized unit, encoded by features characterizing its neighborhood, and the relationships to its neighbours.

6.4.2 A Fusion-based Naive-Bayes Answer Classifier

I used the machine learning and data mining toolkit Weka[1] [Witten and Frank, 2005] to experiment with the training of a Naive Bayes classifier. Table 6.17 presents the results obtained with a 10-fold cross validation for the 50 questions of the test corpus.

Table 6.17: Machine learning scores (Naive Bayes classifier) on 10-fold cross-validation over the test corpus against shallow fusion based reranking scores at rank 10. Figures are computed over satisfying nodes.

ES: EXACT-STRONG, E: EXACT, L:LENIENT

System	Naive Bayes			Re-ranking		
Group	ES	E	L	ES	E	L
# nodes	4963	4963	4963	4963	4963	4963
# satisfying nodes	359	385	672	359	385	672
# output nodes	528	532	639	500	500	500
# satisfying output nodes	126	127	250	138	141	209
Recall	35.1	33.0	37.2	38.4	36.6	31.1
Precision	23.9	23.9	39.1	27.6	28.2	41.8
F-Measure	28.4	27.7	38.1	32.1	31.8	35.7
Accuracy	87.2	86.6	83.6	88.25	87.85	84.80

[1]http://sourceforge.net/projects/weka

Overall, for the 50 questions, 4963 nodes were generated, with 359 satisfying nodes on the EXACT-STRONG level, 385 satisfying nodes on the EXACT level, and 672 satisfying nodes on the LENIENT level. Note that recall, precision, F-Measure and accuracy are scored over satisfying nodes, not answer candidates, since the classifier computes outcomes for nodes, not candidates. The classifier on 10-fold cross validation predicts a F-Measure similar to the re-ranking view based on shallow redundancy: A bit more than a third of the satisfying nodes are automatically identified, with a similar precision.

To examine more precisely the value of each feature group, I performed 2 tests using 10-fold cross-validation:

1. A first test checked the value of the first group of features, characterizing the node itself (i.e. node tag, bag size, frequency, best Google phrase rank and best Google snippet rank).

2. The second test checked the value of the first group of features combined with features characterizing the question (WH-word), and the relationships between the node and question nodes (inclusion, entailment, equivalence, aggregation).

Table 6.18 compares the results of the two tests with the first results, combining the three groups of features. (The tested level is EXACT-STRONG.) For reference, scores over satisfying nodes for both the Google view and the optimal view have been provided. (Note again that, in Table 6.6 for Google, and in Table 6.5 for the optimal strategy, scores are given for satisfying answer candidates, while figures in Table 6.18 are for satisfying nodes.)

Table 6.18: Evaluation of graph-based feature groups for machine learning (10-fold cross validation), and comparison with bottom-line and optimal re-rankers. Scores computed on the EXACT-STRONG level.

1: node features; 2: question related features; 3: answer comparison features

	1	**1, 2**	**1, 2, 3**	**Google**	**Optimal**
# nodes	4963	4963	4963	4963	4963
# satisfying nodes	359	359	359	359	359
# output nodes	114	168	528	500	500
# satisfying output nodes	36	38	126	71	290
Recall	10	10.6	35.1	19.7	80.7
Precision	31.6	23	23.9	14.2	58.0
F-Measure	15.2	14.5	28.4	16.5	67.5
Accuracy	91.9	90.9	87.2	85.5	94.3

The first two groups of features performed very poorly in comparison to a combination of the three groups. The two tests actually performed below the view based on Google ranks (10% against 10.7% recall of satisfying nodes). The high accuracy is characteristic of the *sparse data* problem: A good accuracy can be obtained by discarding most nodes, thus increasing the number of true negatives (unsatisfying nodes identified as such by the classifier) , which is still very high for the corpus. Recall, precision and F-Measure are more meaningful scores, and show that the features in group 1 and 2 are not extremely discriminating for QA classification. On the other hand, features of group 3, which make use of answer comparison, are discriminating enough to perform at a level similar to the best view experimented with in this chapter.

From both machine learning and re-ranking experiments, it appears that the best strategies are the ones involving not only individual characteristics of each node with respect to the question, but also relational features making use of comparison between node candidates.

6.5 Summary

In this chapter, I presented a wide range of experiments, from node re-ranking to node clustering, on a set of 50 open domain questions with an average of 106 web answer candidates per question.

For node re-ranking, I report a significant improvement when using a strategy based on redundancy as opposed to frequency. I defined redundancy as an extended version of frequency, that takes into not just trivial equivalence (stem matching), but more complex relationship between answer candidates (inclusion, entailment and aggregation).

Redundancy provided a significant improvment of the MRR compared to frequency (from 0.63 to 0.72, Tables 6.8 and 6.9), and a 12% improvment of the question recall at rank 1 (from 50% to 62%) for EXACT-STRONG answer candidates. At rank 10, redundancy also provided a 8.5% improvement of the EXACT-STRONG answer candidate recall, and an increased precision (F-Measure for redundancy: 32.90 against 25.36 for frequency).

The best scores (0.75 MRR and 68% question recall at rank 1, Table 6.11) were obtained using incremental modeling, which consists in generating a first model, extracting the top 10 best answer nodes as well as the question nodes from the first model, and generating a new model from which the final re-ranked list of answer nodes is output. Further testing would be needed to assess exactly the impact of the number of nodes in a model on fusion: It is not clear whether incremental modeling showed an improvement because fewer nodes were involved and/or a pre-selection of the best nodes for re-modeling affected fusion. On the other hand, on 10-node modeling, although WordNet helped to find more relationships (and thus increased connectivity), it did not improve re-ranking.

When experimenting with node clustering, I showed that model's subgraphs could be mapped into clusters of related answers. I assessed two types of clusters. Entailment clusters grouped together answer nodes varying in granularity, each member was sorted from specific to general. Clusters based on the graph model's connected components aimed at representing distinct answer referents, or alternative answers, to a question.

Because a cluster (a group of nodes) consists of more answer candidates than a node (group of stem-matching candidates), it was expected that clusters would obtain a better MRR (0.79 for entailment clusters and 0.83 for connected components), as well as a better question recall at rank 1 (72% for entailment clusters, and 76% for connected components, Tables 6.13 and 6.14). As a group, clusters are less precise than nodes (positive merging precision for nodes: 99.75% for EXACT-STRONG candidates, against 75.26% for entailment clusters, and only 59.23 for connected components). However, when looking at the precision over LENIENT answers, both types of clusters prove to be relatively precise entities. The LENIENT positive clustering precisions were respectively 89.92% and 76.01% for entailment clusters and connected components, meaning that between 10 and 24% of the nodes members of a clusters are not satisfying as answers. Interestingly, some of those unsatisfying nodes were still relevant to the question. For instance, the first connected component, with which the question *Who invented the telephone?* was answered with, contained variant phrasings of *Alexander Graham Bell*, as well as the node *1876*, which is the date when Bell allegedly invented the telephone (as explicitly stated by 3 snippets). The presence of such nodes is due to the fact that QAAM does not proceed to any form of question and answer typing as such. For instance, it does not specifically look for person names as opposed to dates for *who* questions. While the date does not answer the question, and more inner filtering of the output clusters would be needed to increase their precision, it helps to increase the connectivity around the answer *Alexander Graham Bell*, thus increasing its redundancy score. With the preliminary results obtained with answer comparison, I would argue in favour of more analysis of the types of errors found in QA results. While there are definitely off-topic answer candidates, as I have shown in this chapter, as well as in Chapter 4 regarding location questions, there are candidates that are not answers but still related enough to provide a supporting network around a satisfying answer candidate.

Contextual information, as provided by fusion-based features, also proved to be useful when training a classifier on graph models. While this thesis mainly describes results that evaluate heuristics developed on training data, Section 6.4 reports similar results when using machine learning techniques (F-Measure of 28.4% and 38.1% considering respectively EXACT-STRONG and LENIENT nodes, against 32.1% and

35.7% for the re-ranking heuristic based on redundancy). Although, I only experimented with a limited set of features and inference techniques for training, I hope to have generated enough interest for research to be pursued in that area.

Finally, in this chapter, I showed that, when concerned with multiple answers, traditional measures such as MRR and question recall provide a relevant description of the performance, but answer recall and precision provide a more accurate indication of a system's ability to handle multiple answers. Moreover, when generating clusters of answers, it is necessary to provide a measure of the inner precision and exactness of each cluster.

Recall and precision are used in TREC QA to evaluate the results to list and definition questions, but there are not available for factoid questions. Factoid scores at TREC QA are usually better than non-factoid scores, e.g. at TREC 13, the best score for factoids was 0.770, 0.622 for list questions and 0.460 for other questions. From my experiments, I argue that this distinction between factoids versus non-factoids is somewhat artificial. As shown from this web corpus, answer multiplicity is not infrequent, and does not necessarily depend on the question type. It is more likely to be dependent on the corpus used, especially how heterogeneous it is in terms of data available for a given question.

Chapter 7

Conclusion and Avenues for Future Work

While experimenting with information fusion, on the basis that the phenomenon of answer multiplicity could be used to improve automated QA performance, several issues were raised relating to core aspects of QA.

One is terminology. For instance, when considering single shot, extraction-based factoid QA, it is perhaps acceptable to use the term "answer" to refer to the string extracted for a given question. However, the term "answer" is generally confusing, especially when dealing multiple extractions. An "answer" could refer to a single extraction, as well as to a list of extractions referring to the same entity. Consequently, the evaluation of multiple "answers" raises issues such as the meaning of answer redundancy, or answer alternatives. While it is trivial to penalize redundancy for string matching extractions, identifying that *What is Vertigo? symptom, movie* corresponds to an ambiguous question with alternative answers, while *What is bipolar disorder? manic-depression, diagnosis* does not, is a harder task.

In this last chapter, I review the different issues that I have come across during this thesis while assessing the value of information fusion for QA. I summarize the results obtained with fusion for traditional re-ranking, as well as more novel experiments, making use of the graph-based representation originating from fusion, and involving the generation of a list of answer clusters, rather than single answers. The second section is dedicated to suggested research areas derived from this thesis.

7.1 Contributions of the Thesis

This section reviews the core contributions of the thesis. With the aim of making use of multiple answers, I had to clarify, at least from an engineering perspective, what I mean by an answer, and I found the MVC paradigm helpful to deal not only with the process of information fusion for QA, but also with the consequent issues fusion raises with respect to terminology. The first subsection summarizes distinctions I made along the thesis regarding answer terminology. The second subsection is dedicated to the data sets I collected and/or annotated to experiment with multiple answers. Finally, I review the main improvements achieved by fusion in my experiments.

7.1.1 QA Terminology

When dealing with multiple "answers" and fusion, the current terminology used in QA when referring to an "answer" can be confusing. While the MVC belongs to the category of design patterns in software engineering, I believe it can help to clarify the terminology used in QA.

The principal aim of the MVC is to separate model and user interface (view) so that data handling is not affected by interface changes. In QA, interfaces changes would involve having a range of views available to the user for a given question. Currently, most QA systems provide a list of strings, with eventually pointers to the source document. But as mentioned in the QA Roadmap [Burger et al., 2002], it would be desirable to *generate* a richer output, and possibly allow users to choose between different outputs depending on their preferences. It would also be desirable to have dynamic interfaces, in which the users can interact with the view. An interaction could include for instance asking for additional media, or change in medium (e.g. from text to pictures), for more content (e.g. clarification), which could translate into either the view rendering more content from the model, or a request for additional data to be modeled and rendered.

However, the most important feature of the MVC, when applied to QA, is the distinguish between data (corpus), representation (graph model) and rendering (views of the model). Previously, the term *answer* could apply to *answer candidate extractions* found in the corpus as well as the output of a QA system, consisting of a list of *answer strings*. When performing fusion, there is not a necessarily direct mapping between found extractions and output strings, because answer candidates are merged and normalized. The final output, although based on extractions, actually corresponds to a process of answer generation or rendering, in MVC terms. This has an impact on evaluation, since the scores then assess group of candidates rather than single candidates. In order to clarify the terminology around the term *answer*, I propose the following definitions.

Answer candidate extraction. In text-based QA, an answer candidate extraction corresponds to a string that will be used for answering. (The extraction may vary in size (e.g. sentences, phrases.) This material may include strings that represent relevant information for the question, or simply related information although not directly relevant, and, expectedly, irrelevant information. It is basically the raw, unprocessed data used for answering. "Used for answering" means that the data will be processed in the QA pipeline, although it may not necessarily be shown to the end user after processing and filtering.

Normalized answer candidate. When projecting answer candidate extractions into a model, operations of normalization may be performed. In this thesis, I opted for a representation based on pairs of attribute-value indicating linguistic or statistical information about an extraction. In Chapter 6, the projection into nodes consisted in merging stem-matching extractions. Thus each node (answer candidate normalization) actually represented a set of answer candidate extractions. This may or not be the case in other representations. But, in order to compare extractions, whether it be through fusion or simple filtering, some level of annotation is usually required. For instance, most QA systems include frequency scores for a given extraction ("answer frequency"), this is a form of normalization assuming that string matching extractions are equivalent. (They may not be.) The normalized extraction is a enriched version of the extraction.

Answer candidate rendering. Rendering is the process of representing an answer candidate to the user. It involves two major aspects: (1) the medium chosen for rendering (e.g. text, sound, picture), and (2) the content to be rendered. While, in traditional

QA, the view or final output presented to the end user consists of a list of answer candidate extractions, a view could actually consist of generated text merging several extractions. For instance, in Chapter 6 of this thesis, I experimented with clusters grouping normalized answer candidates.

I have used the term "candidate" so far to indicate that each corresponding entity is the result of an automated process, which could include satisfying answer candidates, relevant but unsatisfying ones and off-topic candidates. I propose to use the term answer only when referring to what has been judged to be indeed satisfying.

Now, when handling multiple "answers", for evaluation purposes, one may want to know how many "answers" have been found by the system, and similar statistics. This is where the definition of "answer" is non trivial, because the "answer" may be instantiated in different manners (both in terms of content and rendering). The same "answer" could be rendered in various ways (e.g. *Where is the Taj Mahal?* could be answered with just *"Agra"*, or *"Agra, India"*, or a map). When I started assessing answer clusters, I required each cluster to be evaluated, i.e. the relevance of each cluster member, but also the list of clusters, i.e. the "answer" as a *whole*.

In this thesis, I considered an *answer* to be a structured object contained in the *model* and retrieved by a given strategy to build a *view*, i.e. there is only one answer per question, and it has two core attributes: content and rendering. An answer is a structured object, and can eventually be divided into one or more **answer referents**.

In Subsection 6.3.2 (Chapter 6), I introduced the notion of **answer referent** to mean the abstraction answer candidates extractions could be co-referring to. Earlier, I provided the example of *What is Vertigo? symptom, movie* as opposed to *What is bipolar disorder? manic-depression, diagnosis. Symptom* and *movie* address two distinct answer referents, *vertigo* as a medical symptom, versus *Vertigo* as the Hitchcock's thriller. On the other hand, *manic-depression* and *diagnosis* are referring to the same answer referent: *Bipolar disorder*, also previously known as *manic-depression*, is a psychiatric *diagnosis*.

A question may have only one answer referent or several answer referents, in which case they are necessarily alternative answer referents. An answer referent can be described using different media. If text is used, an answer referent can be characterized by multiple answer candidate extractions. The relationships between those extractions can be, locally, equivalence (paraphrases), inclusion/entailment (granularity), aggregations (different aspects) and/or alternatives (contradictions within a referent). Indeed, there could be local alternatives for a given answer referent. In Subsection 6.3.2, I mentioned the case of *3000 mosquitoes each night* and *600 mosquitoes an hour* (*What do bats eat?*). The alternative is only local, because overall, both extractions agree on the fact that bats eat *mosquitoes*, only the quantities are subject to alternatives[1].

While, so far, the type of QA output has been simple enough to be referred to with expressions such as *list of answers*, I consider an answer to refer to the whole output, and to be composed of one or more answer referent. I found the task of assessing answer referents to be feasible on a set of 10 nodes per question. But, to be more rigorous and propose concrete guidelines for such a task, further investigation would be required on a corpus of questions annotated with multiple answer candidate extractions. For instance, in Subsection 6.3.2, for 46 questions having at least one satisfying answer candidate extraction (on EXACT-STRONG level) within the top 10 nodes, only 7 questions had multiple answer referents, although, on average, there were about 3 distinct satisfying answer candidate extractions per question ("distinct" meaning they did not string-match). Although there are many multiple answer candidate extractions, the actual amount of answer referents appears to be much lower. This thesis was concerned with multiple answer candidate extractions, that had been found satisfying for a given question. But there is currently very little information about the actual amount of multiple answer referents per question. This would require a new level of annotation, taking into account answer complexity.

In the next subsection, I remind the reader of the two corpora oriented towards multiple answer extractions I produced during this thesis, and that would be available for further investigation.

[1]One may argue that both expressions are equivalent if the duration of a night is considered to be 5 hours: the evaluation of inner relationships is not trivial.

7.1.2 Corpora of Multiple Answers

In this thesis, I experimented on two corpora for which I collected data. In Section 4.1 (Chapter 4), I introduced a corpus based on TREC QA questions and system's judgments for the tracks 8 to 11. At TREC QA 11, a new criterion of answer exactness was introduced, aiming at a better pin-pointing of answer extractions (meaning shorter answer strings). Earlier judgments were composed of slightly longer strings. To homogenize the dataset, I derived exact candidates from the earlier judgments.

The main corpus is composed of all systems' judgments with their TREC evaluation flag ("correct" versus "incorrect" answer), and their source document from the AQUAINT corpus. Linguistic annotations are also available for each answer candidate extraction (stems, lemma, POS-tags and chunk tags). From this master corpus, I derived a smaller corpus focused on location questions with a list of 10 judgments' per question. More than half of the questions have several correct answers.

The advantage of this corpus is that it is derived from a state-of-the-art evaluation, and the extractions, whether correct or not, are the judgments provided by real QA systems. The inconvenient, on the other hand, is that extractions come from different systems, and they reflect different processing choices for the task. This may be a source of bias in the collection.

In Chapter 5, I introduced a new corpus where extractions have been collected from Google web snippets. The answering material is available in an unprocessed form (first Google screen page), and a tokenized form (extracted phrases). For each question, answer candidate extractions have been fully annotated, only by a single rater so far, although it is planned to obtain judgments from other annotators in order to validate the current ratings.

This data collection only represents a small subset of the web data available for 50 questions, but it is provided with an exhaustive annotation of multiple answers for the retrieved snippets.

Instead of the traditional binary annotation, correct versus incorrect answer, I described a scaled rating for answer candidate strings, based on two core assessments:

1. the *exactness* of the answer string, in term of pin-pointing and level of informativity.

2. the value of the *evidence* for the answer string, i.e. a score for the textual context (in this corpus, the snippet the answer string was extracted from) as a basis for belief.

On the basis of evidence and exactness scaled ratings, I distinguished between three categories of answers. EXACT-STRONG answers represent a core set of candidates that are both supported by strong evidence as well as exact answers. The group of EXACT answer considers candidates that are exact answers but could be supported by either strong or weak evidence.

I argued in favour of a more lenient evaluation, accepting, for instance, opinions as long as the evidence is explicit enough to support the interpretation, instead of a strict annotation of answer correctness. Indeed, when an answer is annotated as "incorrect" it is not clear whether it is because it is unsupported, inexact, or because the expressed opinion defies common knowledge. I argue that, in the later case, systems should not be penalized. As in automated summarization, I believe systems should be first assessed on their ability of reporting accurately the information provided by a given corpus, whose quality may vary independently of the system's actual performance. The task of assessing how authoritative an answer is requires to first identify the range of available answers, and, then, to compare them.

Finally, both corpora show that it is not rare for factoid questions to have multiple satisfying answer extractions. In the corpus of location questions, more than half of the questions had multiple acceptable answers. For the web corpus, on average, 10 snippets per question provided 7.5 EXACT-STRONG answer candidates, 8 EXACT and 14 LENIENT answer candidates (counting distinct answer strings only). Although the process of candidate extractions induced some redundancy, the amount of multiple satisfying extractions is non-negligible. While definition questions (36% of the corpus) have many distinct multiple answers, all question types, including factoids, actually expect multiple answers, showing that answer complexity is not necessarily a matter of question complexity nor question type.

7.1.3 Information Fusion for Traditional Re-ranking

Compared to traditional re-ranking, in which the output consists of a list of answer candidate extractions, answer comparison appears to improve significantly the task of answer re-ranking (Chapter 4).

On a restricted set of question types (85 location questions), a baseline only using relationships between question nodes and answer nodes found an acceptable answer at rank 1 in 49% of the questions (MRR of 0.63). A strategy based on fusion, using not only relationships between question nodes and answer nodes, but also between answer nodes themselves, top-ranked an acceptable answer for 72% of the questions, achieving a MRR of 0.82 (Table 4.7).

In this experiment, fusion also proved to be more resistant to irrelevant answers (marked incorrect in TREC data). Instead of considering answer candidates as competitors only, fusion allows to make use of multiple candidates by generating a supportive network around a satisfying answer. Interestingly, this network does not only consists of nodes representing "correct" answers, but it also includes "incorrect" but related candidates. In the next section (Subsection 7.2.2), I propose what I call "node typing" in order to identify more precisely in the graph models which nodes correspond (1) to satisfying candidates, (2) to unsatisfying but related candidates which seem to play a role in the performance of fusion, and (3) to candidates that irrelevant or off-topics and are a source of noise in the graph models.

7.1.4 Information Fusion for Answer Clustering

In Chapter 6, I experimented with the rich and flexible structure available from graph models. I evaluated fusion on the web corpus, containing fewer questions (50 against questions) but a larger input, with an average of 106 web answer candidates per question, as well as a greater variety of question types.

I first compared again lists of re-ranked nodes (although nodes this time represented stem-matching candidates instead of a single extraction), and opposed two strategies, one based on the frequency of an answer node (the number of stem-matching candidates referred to), and the other based on redundancy, which I defined as an extended fusion-based version of frequency. While frequency gives an indication about strings judged equivalent, and merged into a single node, redundancy takes into account a large range of relationships between nodes (equivalence, inclusion, entailment and aggregation). Thus, redundancy indicates the number of occurrences relating to a network of nodes.

The view based on redundancy achieved a MRR of 0.72 and a question recall at rank 1 of 62%, against respectively 0.63 and 50% for the view based on frequency (EXACT-STRONG scores, 6.8 and 6.9). At rank 10, redundancy also provided a 8.5% improvement of the EXACT-STRONG answer candidate recall, and an increased precision (F-Measure for redundancy: 32.90 against 25.36 for frequency).

Frequency (equivalence) has been previously shown to improve answer re-ranking [Abney et al., 2000, Clarke et al., 2001, Brill et al., 2001]. The scores of a redundancy-based strategy prove that other relationships, such as inclusion/entailment and aggregation, are also helpful, especially when relevant answer candidates have a very low frequency, but are surrounded by frequent contextual nodes (i.e. nodes representing candidates that are not satisfying, but still related).

In the corpus based on TREC judgments (location questions), answer frequency was low. Inclusion based on overlap, as well as non trivial (WordNet based) inclusions and equivalences, were the most exploited sources of connectivity between answer candidates. In the web corpus, 42% of the relevant answers (LENIENT group) only occured once. Although frequency can help to improve measures based on the first ranked satisfying answer (MRR and question recall), measures assessing the strategy for multiple answers (answer recall and precision) are difficult to improve on the basis of frequency alone.

When experimenting with node clustering, I showed that model's subgraphs could be directly mapped into clusters of related answers. I assessed two types of clusters: (1) Entailment clusters, grouping together answer nodes varying in granularity, and (2) clusters based on the graph model's connected components, with the aim of representing distinct answer referents, or alternative answers, to a question.

Because a cluster of nodes consists of more answer candidates than a single node, it was expected that clusters would obtain a better MRR (0.79 for entailment clusters and 0.83 for connected components) and a better question recall at rank 1 (72% for entailment clusters, and 76% for connected components, Tables 6.13 and 6.14). To assess the precision of clustering itself, I provided an evaluation measure based on the number of distinct satisfying candidates in each cluster (clustering precision), as well as a score based on the mean exactness of answer candidates members of the cluster (clustering exactness). Clusters are less precise than nodes (positive merging precision for nodes merging stem-matching candidates: 99.75% for EXACT-STRONG candidates). Entailment clusters had a positive clustering precision of 75.26%, connected components obtained a precision of 59.23%. However, when looking at the precision over LENIENT answers (including inexact and weakly supported candidates), both types of clusters prove to be relatively precise entities: Between 10 and 24% of the nodes members of a cluster are not satisfying as answers, for a mean cluster size varying between 2.34 (entailment clusters) and 4.5 (connected components).

Fusion also proved to be useful when training a classifier on graph models. In Section 6.4, I reported preliminary results with machine learning. A Naive Bayes classifier trained on graph models achieved a score similar to the heuristic manually devised on the training set. (F-Measure of 28.4% and 38.1% for the classifier on 10-fold cross-validation considering respectively EXACT-STRONG and LENIENT nodes, against 32.1% and 35.7% for the re-ranking heuristic based on redundancy).

Although information fusion proved to improve QA performance quantitatively (increased question and answer recalls), as well as qualitatively (through generation of more structured answers), there are limitations to the work presented here. The following section reviews open issues encountered with graph models, as well as suggested improvements and research questions deriving from this work.

7.2 Avenues for Future Work

This section reviews limitations and possible improvements of the graph models presented as means for fusion in this thesis. The first issue concerns transitivity for the relationships inferred between nodes. The second concerns a system of node typing to differentiate between nodes that can act as satisfying answers, as opposed to nodes that are only related to the question or answer topic, but cannot serve as answers themselves. Finally, I introduce two new tasks, one based on multiple choice questions to investigate multiple answers, the second concerned with more elaborate rendering of answers, involving the generation of illustrated answer summaries.

7.2.1 Transitivity in Graph Models

Transitivity is a useful property that can be used to discover new relationships to compensate for the lack of knowledge about unknown expressions. However, depending on the techniques, especially those based on similarity rather than strict equivalence, transitivity does not always insure correct inferences for the broad set of relations that I subsume under equivalence (noted $\leftrightarrow$), because similarity is not transitive. Figure 7.1 shows examples of such non-transitivity.

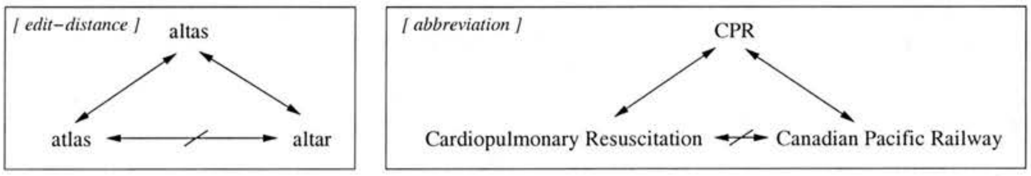


Figure 7.1: Non-transitivity cases over the equivalence relationship

Configurations such as in Figure 7.1 signal that a parent node is ambiguous. For instance *altas* could be a misspelling for *atlas* or *altar*, and *CPR* stands for both *Cardiopulmonary Resuscitation* and *Canadian Pacific Railway*. Both parents, *atlas* and *CPR* are ambiguous.

Transitivity in inclusion (noted $\rightarrow$) is also problematic because the relationship subsumes heterogeneous relations as well. Let us consider the following inclusive chain:

$$\textit{continent} \rightarrow \textit{Europe} \rightarrow \textit{Scotland} \rightarrow \textit{Edinburgh}$$

While *continent* $\rightarrow$ *Europe* has been detected by the hyponym pointer in WordNet, the three others have been discovered by the spatial inclusion (partonym/meronym) relation (in WordNet as well).

Heterogeneous relationships have been studied in [Chaffin et al., 1988] who proposed a taxonomy of part-whole relations that he extended later to a more general analysis of semantic relations [Chaffin, 1992]. Chaffin distinguishes three elements in an inclusive relation: the <inclusion> itself, the <connection> between the two entities and their <similarity>. For instance, spatial inclusion is made of one element only, <inclusion>, the meronymy relation is based on <inclusion> and <connection> (engine/car) while the hyponymy relation is the most complex relationship with the additional element of <similarity>.

Chaffin uses syllogisms to illustrate the validity of his relationships. Consider the following consistent syllogism using inclusion:

Edinburgh is in Scotland.	<inclusion>
Scotland is in Europe.	<inclusion>
Edinburgh is in Europe.	<inclusion>

Chaffin argues that *a syllogism containing two different kinds of inclusion relations in its premises is valid if and only if the relation in the conclusion is the simpler of the two relations*. The latter syllogism is valid because the conclusion *Edinburgh is in Europe* is not composed of more elements than the premises.

The following syllogism is not valid because its conclusion involves hyponymy which is a more complex relation that the one used in the premise (spatial inclusion). Chaffin's elements that make the conclusion more complex than the premises are printed in bold face.

Scotland is in Europe.	<inclusion>
Europe is a continent.	<inclusion> <connection> <similarity>
*Scotland is a continent.	<inclusion> **<connection>** **<similarity>**

In this case, transitivity is inconsistent. In automatically generated models, transitivity can result into even worse results because of equivalence ambiguities mentioned above (e.g. word or world referent ambiguity). For instance, Edinburgh can refer to either Edinburgh, Scotland or Edinburgh, Indiana. Also, inference techniques are not always consistent with each other. For instance, contextual inclusion based on co-occurrences count is not always consistent with overlap inclusion (see Subsection 3.2.2).

One way around inferring incorrect relationships by transitivity (besides dropping the computation of the graph's transitive closure) is to preserve which technique has been used to infer the relationship, for instance, a hypernym lookup in WordNet as opposed to a partonym lookup, or Edit-Distance as opposed to synonym lookup, and block transitivity on incompatible relationships.

However, transitivity should still be considered carefully when clustering a graph model. If nodes are grouped on the basis of being connected by a broad, heterogeneous, type of relationship, then the resulting cluster may include nodes that are connected through unreliable transitivity. For instance, if equivalence serves as the basis for clustering, *CPR*, *Canadian Pacific Railway* and *Cardiopulmonary Resuscitation* will end up in the same cluster. The answer nodes in this cluster are actually alternative answers. Another example for inclusion is shown in figure 7.2 : Since both *Agra* and *Atlantic City* are *touristic* places, clustering based on inclusion will again wrongly collect alternative answers in the same graph partition.

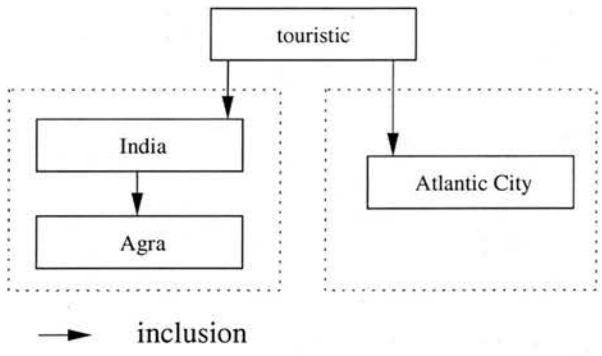


Figure 7.2: Clustering with inclusion

On the other hand, nodes may not be directly connected despite the fact that they are referring to the same answer. For instance to the question *Who was Galileo?*, *astronomer* and *inventor of the telescope* may not be directly connected because the relationship between the two candidates is not strictly speaking an equivalence, but rather a co-reference. Both qualifications refer to Galileo indeed, but the system is unable to resolve the anaphora. A similar problem occurs with collective answers. To the question *What were Christopher Columbus' three ships?*, the candidates *Nina*, *Pinta* and *Santa Maria* cannot be directly connected through inclusion or equivalence. [Webber et al., 2002] define such a relationship as *answers that are mutually consistent but not entailing can be replaced by their conjunction (aggregation)* (see Table 2.6).

To conclude, inclusion and equivalence are useful to identify relevant groups of nodes, but they are not sufficient to recognize alternative answers. One needs to generate a graph model based not only on ontological relationships such as equivalence and inclusion, which denote paraphrases or granularity, but also more pragmatic, contextual relationships (aggregation and alternative) to recognize that two nodes are, or not, referring to the same answer, whether because they actually co-refer or because they serve as a collective answer.

7.2.2 Improving Modeling with Node Typing

In a perspective based on the recognition of multiple answers, it appears that a *relevant* expression for QA modeling is not necessarily an answer. From my previous experiments, I discovered that some candidates – themselves unsatisfying as answers but related to satisfying ones – were playing a role in connecting nodes in answer models, and thereby telling us something about the answer.

In the first experiment on location questions, the ratio of answers among the candidates was 16.6% (i.e., one answer for each five candidates). Because answer candidates were judgments from TREC QA systems, QAAM benefited from the filtering the systems had performed, and the input was somewhat homogeneous. The presence of related expressions might have been a bias from the collection I used. However, in the second set of experiments, using the web as raw input, the system had to deal with a larger unfiltered and heterogeneous input. The ratio of satisfying answer nodes (EXACT-STRONG level) dropped to 7.2%, but despite the important amount of candidates that were unsatisfying as answers, contextually related nodes still helped to generate large clusters around correct answers.

Although in my experiments I only considered two types of nodes, question nodes versus answer nodes, looking at the nodes of graph models, one can actually observe four types of nodes, which I classify as follows:

- *question nodes* projected from the question.
- answer nodes that are correct answers. I call them *nuclear* nodes, i.e. nodes that contain a core information, that fills the gap expressed by the question.
- *satellite* nodes are the nodes that are relevant to a *nuclear* node or a *question* node, but are not answers. They provide background information.
- *blank* nodes are either off-topic or irrelevant nodes.

A standard top-down QA pipe-line usually approaches the problem of answer recognition by using filters and weighting for re-ranking. The goal is to identify candidates that are correct answers (*nuclear* nodes), and discard everything else (*satellite* and *blank* nodes). In machine learning approaches to answer classification, this translates into a binary classification. For instance, [Ittycheriah and Roukos, 2002] (IBM's statistical system) base their system as follows: *We model the distribution p(c|a, q), which attempts to measure the c, "correctness", of the answer and question. c can take on values of either 0 and 1 indicating either an incorrect or correct answer respectively.* These two sentences very well sum up the attitude that is currently dominant in QA. Although the frequency of an answer *a* is used as a statistical feature to induce its potential correctness, the likelihood remains centrally based on checking that the pair question-answer is fitting. What I have introduced in the thesis is that a "good" answer is not only determined by the question but also by its context and the other possible answers to the same question. Absolute *correctness* is not the only relevant criterion when assessing answers, and that a good evidence or justification is as important as the answer itself. To this end, *satellite* nodes not only help building a supporting network around a *nuclear* nodes, they can also be used in the final output as means of justifying the answer, as shown in my mixed-media experiment. For instance, pin-pointing *Mughal architecture* can help justify the Taj Mahal in Agra, while pin-pointing *casino* and *Trump Taj Mahal* can help justify the Taj Mahal in Atlantic City.

I believe that a ternary classification of candidate nodes (*nuclear*, *satellite*, *blank* nodes) may benefit answer recognition more than a binary one (*correct*, *incorrect* nodes). While there has been extensive research on characterizing *nuclear* nodes, i.e. answer typing through syntactic and semantic type checking, little has been done about what I call *satellite* nodes. The closest type of research to *satellite* characterization is perhaps the field of query expansion. In query expansion, trigger terms, i.e. terms that are assumed to strongly relate to a potential answer (e.g. measure units for a length related question, for example *feet*, *meters*, *elevation*, *high* for *How tall is Mount Everest?*) are collected and used to generate a set of keywords to extend the description of the initial question.

In answer models, it would be worth investigating whether *satellite* nodes have typical relationships with a question node or their nuclear neighbours, or if they can be automatically identified by the number of in-coming and out-coming edges they generate during relationship inference, since they play an important role in clustering. Some *satellites* might be identified as more significant than others. For instance to *Where is the Taj Mahal?*, the terms *location* and *tourism* are satellites that recurrently appear in the IR output for questions related to famous places. *Mughal architecture* and *casino* are not so frequent across TREC-like questions; however they do have an important frequency count for the specific *Taj Mahal* question. While the terms *location* and *tourism* are good generic indicators of answer nodes, *Mughal architecture* and *casino* could play a discriminating role in identifying alternative answers.

7.2.3 Multiple Choice Questions

Rather than open-ended "wh" questions, the field of QA could also advance through the use of multiple choice questions. The advantage of multiple choice questions is the time and effort usually spent on assessing answers would be greatly reduced because the number of answers would be fixed and their validity assessed beforehand. Questions could be of any type including yes/no questions which are often discarded in state-of-the-art QA evaluations.

In such an evaluation, systems would be allowed to use any kind of resource, but they would be limited in time, which is another criterion that is currently not taken into account during state-of-the-art evaluations. The time factor expresses a trade-off: Would you prefer using a system that has an average to low performance but answers quickly, or a system whose accuracy is a bit higher but takes a long time to answer? Overall, you might get a correct answer more quickly with the first system, by modifying your query in repeated interactions. This usability aspect should be a criterion when assessing QA systems.

As for multiple choice questions, I imagine two levels of answering:

1. Level 1 consists in selecting the one or more satisfying answers to the question. This selection can be assessed in a score based on precision and recall.

2. Level 2 includes, besides selecting the right answers, providing a satisfying justification for the selection. This justification would not be assessed in terms of correctness. Rather, its qualities would be to be a good justification of the answer (judged on a quality scale), to be easy to understand, self-sufficient, and not too long to read (i.e. no full documents).

A similar exercise has recently been started (April 2006) in the context of the cross-language QA evaluation (CLEF): the Answer Validation Exercise (AVE)[2], which combines aspects of QA and Textual Entailment[3]. In this exercise, the input consists of (1) a question, (2) a piece of text (one or more sentences) and the name of the document it has been extracted from, and (3) the hypothesis (a sentence). The system's goal is to validate or invalidate the hypothesis (i.e. is the hypothesis true or false given the text and the question). A training corpus for English and Spanish is available, but the English corpus does not provide item (2), the piece of text extracted from the document: Systems must refer to the full supporting document to validate the hypothesis.

This exercise is slightly different from the one I propose. First, I assume systems that can use fusion, i.e. that can base their answer on more than one document. Also, each AVE triple forms a pair question/answer, rather than a question and a set of answers. While AVE indeed could improve systems' self-assessment, it does not provide a framework to assess answer *justification*. The *validation* is essentially based on finding an entailment relationship. It does not provide the user with some text that can explain the explain the validation of an answer, with respect not only to (1) the question, but also (2) to the other answers, e.g. (1) there are written indications of a Taj

[2]http://nlp.uned.es/QA/ave

[3]http://www.pascal-network.org/Challenges/RTE2/

Mahal in *Agra*, and a Taj Mahal in *Atlantic City*, and (2) the Taj Mahal in Agra is distinct from the Taj Mahal in Atlantic City (as opposed, for instance, to two different opinions). In short, a validation is a binary decision, while a justification requires both a validation and some output that explains the validation with respect to the question and the other answers. This may look more elaborate but it may actually simplify the validation of some of the hypotheses in the AVE training corpus. For instance, one hypothesis to the question *Where is the Brandenburg Gate?* is *Berlin*. Further down, another triple is found with the same question and the answer *East Berlin*. If *Berlin* has been validated, then *East Berlin* is likely to be valid too. By comparing answers, one can simplify and improve the validation process.

AVE and the first level of answering I propose for multiple-choice questions are equivalent (validation systems). In AVE, hypotheses have been generated from a question and answer candidate pairs, for instance *Where will the Olympic Games take place in 2000?* and the candidates *Sydney*, *Greece*, *U.S.* generated the following hypotheses: *The Olympic Games take place in Sydney in 2000*, *The Olympic Games take place in Greece in 2000*, *The Olympic Games take place in U.S. in 2000*. Notice that because hypotheses are automatically generated, they may not be syntactically or semantically sound (e.g. *What is UNITA?*, one of the generated hypotheses is *UNITA is the Angolan army continued to attack*). Hypotheses are meant to simplify the validation process. However, for a task oriented towards justification, I would propose to keep the original answer (e.g. *Sydney*, *Greece*, *U.S.*), or an actual extraction rather than a generated statement, because the generation process introduces unnecessary mistakes, and it should be up to each system whether the answer should be apprehended as a correct/incorrect statement in the form chosen for AVE, or something else. Also, the form of the answer is a rendering choice. If the answer is produced using natural language techniques, then it should be evaluated separately as such.

The second level of answering would require the community to reflect on what a good justification is. In TREC 11, a justification was optional, and most participants included the string surrounding the answer as a justification. However these justifications were not evaluated, and there was no discussion of whether the surrounding context is enough to justify an answer with respect to a (possibly ambiguous) question and other answers. Moreover, systems using fusion (whose answers are derived from multiple documents) would have to choose the best surrounding context from these documents, or generate a justification. Also, it would be useful to require a justification to contain elements such as the author's name and the authoring date to facilitate answer understanding.

In terms of answer modeling, I mentioned that *satellite* nodes, the nodes that are not answers, but do relate to them, could be used to provide answer justifications, especially in the context of multiple answers. This is an aspect I have not been able to evaluate separately, because it is not clear yet what makes a good justification.

Such evaluation would assess the data mining power of each system, its accuracy and its ability to produced well-justified answers, which is currently an aspect of answering that is not taken into account in current evaluations.

7.2.4 Mixed-Media Rendering

While, in this thesis, I mainly experimented with extractions and clusters of extractions, it is also possible to imagine answers that have been reconstructed or generated, such as answer summaries, using either extraction-based techniques or language generation. My experiments focused on text-based rendering, but it is also possible to envisage the rendering of answers using different media, such as pictures. In this section, I present a prototype experiment to articulate the type of answers one could generate, and the research questions relating to answer rendering.

For this prototype experiment, a view consisted of a list of clusters computed by QAAM and rendered into an HTML page using a template. The template structure consists of 3 elements for each cluster: (1) a **summary** with a title, (2) a main **picture** and (3) a **reference** section providing thumbnail pictures – if more than one picture was judged relevant to the topic – and a list of hyper-links pointing to the web sites where the pieces of information were originally found for both the summary and the pictures. This preliminary view was generated for 230 TREC QA questions of various types, using Google as back-end for data collection. (This set was different from the corpus described in Chapter 5.)

I will now take the question *What is amitriptyline?* and a sample answer cluster: {*depression, treat depression, tricyclic antidepressant, chronic pain, Elavil*} as an example of how the template was instantiated from the subgraph given in Figure 7.3.

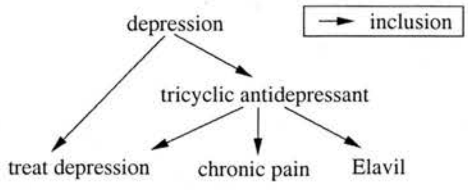


Figure 7.3: First answer cluster for *What is amitriptyline?*

The summary consists of a title, meant to be a general description of the topic addressed the cluster, and of a brief sentence-based description of the cluster.

In a graph, the title maps well to the most inclusive node of the subgraph, i.e. the root node that has the largest number of children, in this case *depression*. The summary was constructed from the "sentences" (they are not always grammatically sound sentences) available in the set of source snippets for the cluster. Using graph-based features, sentences were ranked by informativity (defined by the number of leaf nodes mentioned in the sentence), and the sentences that do not contain any new information (i.e. information already provided in a higher ranked sentence) were discarded. For example, the leaf nodes (*treat depression, chronic pain, Elavil*) lead to the following sentences:

(a) *Amitriptyline, **Elavil**, Endep is a tricyclic antidepressant used to **treat depression** and **chronic pain***

(b) *Amitriptyline, an antidepressant (mood elevator), is used to **treat depression***

(c) *Amitriptyline is useful in reducing the symptoms of **chronic pain** in the following skin conditions*

The leaf nodes mentioned in (b) and (c) are already present in (a). The last two sentences are thus discarded as redundant. Title and summary generated for such an answer topic would be then:

depression

Amitriptyline, Elavil, Endep is a tricyclic antidepressant used to treat depression and chronic pain.

Pictures were retrieved independently, from Google Images. Google Images indexes pictures based on the text of the page that includes the picture. To match these index terms, a query to Google Images consisted in the concatenation of the main keyword (focus) of the question (*amitriptyline*) and the title of the summary (*depression*).

The 20 first pictures provided by Google were re-ranked by the number of keywords present in their URLs. The regular expression matching was performed on tokenized URLs and weighted depending on the keyword matched. Answer keywords were more valued than question keywords so that the picture reflect the answer cluster rather than the question. Finally, the top 5 re-ranked pictures were downloaded, the first one becoming the main picture and the others included as thumbnails in the template. In the amitriptyline example, only one picture was considered relevant and output as an illustration of the summary (Figure 7.4).

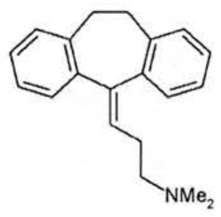


Figure 7.4: Main picture for *What is amitriptyline?*

The last section of the template, the reference section, was built from the list of source URLs of the summary and pictures to orient the user interested in further search.

Although sentence-based summaries provided a wider context to explain answer multiplicity, e.g. the two answer clusters created in response to the question *What is the capital city of Ethiopia?* distinguished between *Addis Ababa* (current capital city) and *Gonder* (17th century capital), questions did not always benefit of summaries. One characteristic of QA is that an answer might be found incidentally in a document whose topic is actually not related to the question at all. [Lin and Katz, 2005] give an example of this phenomenon regarding the answer to *How many floors are in the Empire State Building?*, which could be found in a document about the Great Depression. Segmentation for fusion in QA may have to be finer grain than paragraph level because the immediate context is not always relevant and may need to filtered out. Although a paragraph is easy to read as a unit, in QA, it may introduce totally unrelated context that will make the answer harder to understand. For instance, *What is the Ohio state bird?*, from a readability point of view, was very loosely answered with *My travel buddy is a red cardinal beanie baby that represents the Ohio state bird.* In such cases, a simple phrase (the node value) would have been more accurate.

Also, ideally, the summary should be produced using more sophisticated language generation techniques, as proposed in the past literature on automated summarization from multi-documents [Barzilay et al., 1999]. However, the final text should not summarize the sources sentences themselves but focus on the information relevant to the formulation of an answer summary.

Pictures, on the other hand, helped to identify visually alternative answers. For instance, *What is vertigo?* returned a template with a logo from the Vertigo Software company, a picture of nurses illustrating the topic about the medical symptom, a logo for the Vertigo comics, and a picture of Hitchcock when the topic related to the movie. Also, definitely wrong answers are more easily identified straight away.

Even though there is no formal picture annotation on Google Images, location questions returned answer topics illustrated with maps, and the question *What is the national anthem in England?* directly led to a image representing the music sheet for God Saves the Queen.

Overall, pictures were useful by providing either background information or a direct representation of the answer topic, often much better than text, especially when the question expected a world referent as an answer. Richer annotation of the pictures will eventually make use of them in creating illustrated views more effective. But overall, images were surprisingly good, even though the retrieval was performed on non-annotated pictures.

Interestingly, Google has recently started to provide pictures for some queries performed on their text search engine. For instance, *Ohio state bird* asked to Google web (not Google Images) is responded with a list of documents preceded by pictures of the Ohio state bird (Figure 7.5). I could not find references regarding this new Google feature. It seems to be automated and subject to errors. For instance a search on *New Jersey state bird* provides two pictures of the Eastern Goldfinch, but the third picture is a map of the state of New Hampshire.

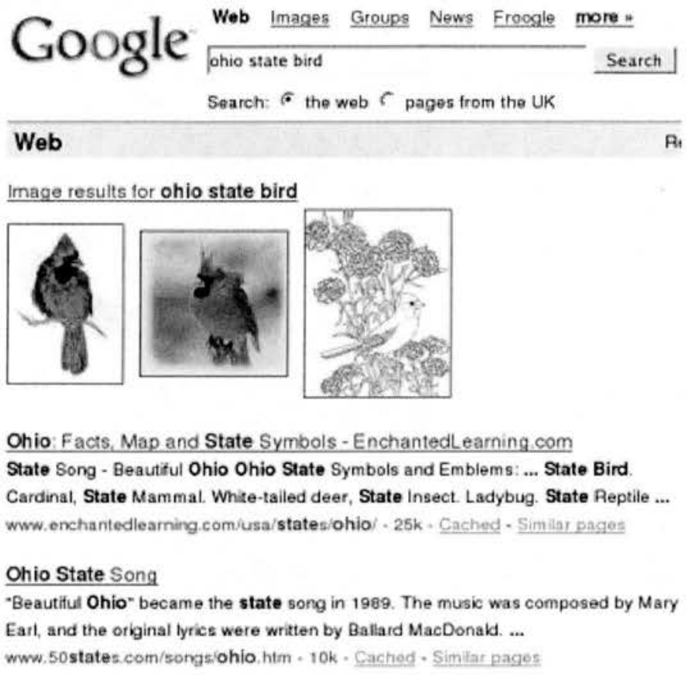


Image results for **ohio state bird**

Ohio: Facts, Map and **State** Symbols - EnchantedLearning.com
State Song - Beautiful **Ohio Ohio State** Symbols and Emblems: ... **State Bird**. Cardinal, **State** Mammal. White-tailed deer, **State** Insect. Ladybug. **State** Reptile ...
www.enchantedlearning.com/usa/**states**/**ohio**/ - 25k - Cached - Similar pages

Ohio State Song
"Beautiful **Ohio**" became the **state** song in 1989. The music was composed by Mary Earl, and the original lyrics were written by Ballard MacDonald. ...
www.50**states**.com/songs/**ohio**.htm - 10k - Cached - Similar pages

Figure 7.5: Google text results for *Ohio state bird* now include pictures (September 2006)

This mixed-media experiment was a prototype for further investigation of the value of pictures and summarization for QA. The following table presents a proposal for an evaluation grid that would be filled by a human judge to assess the results for such a view.

Question 3 (correctness) is asking about the user's perception rather the actual answer correctness. This is because an answer might indeed be correct but presented in such way (e.g. no context, no justification) that the user may class it as a system error, and vice-versa.

The first group of questions (1-2) is meant to be asked while only the question is shown. The second group (3-4) evaluates how quickly a user assesses a system's results. The user is then redirected to the third group's questions (5-12) and provides a score assessing its perception of the quality of each answer by evaluating answer redundancy (5-6) and answer comprehension (7-12).

Table 7.1: Mixed-media answer metrics

Question Assessment (before viewing the results)		
1	Do you know the answer to the question?	Yes/No
2	How difficult do you think the question is?	Easy/Medium/Difficult
Assessment per question (viewing the results)		
3	Does one or more of these answers seem correct to you?	Yes/Maybe/No
4	Automated assessment: time spent to answer (3)	
Assessment per answer		
5	Does this textual answer or a similar one appear elsewhere?	Yes/Not sure/No
6	Has this picture or a similar one been shown before?	Yes/Not sure/No
7	Do you think the textual answer is correct?	Yes/Maybe/No
8	Is the textual answer written in proper English?	Yes/Acceptable/No
9	Does the picture represent the answer?	Yes/Maybe/No
10	Does the picture indicate the context of the answer?	Yes/Maybe/No
11	Is the picture relevant to the question?	Yes/Do not know/No
12	Is the picture relevant to the given answer?	Yes/Do not know/No

Although such comparison could be performed on systems that do not perform answer modeling, I suspect that a fusion-based system that uses relationships such as inclusion or that performs answer co-reference resolutions might be less redundant and provide a greater variety of pictures.

7.3 Summary and Contributions

One of the main challenges of open domain QA on free text is that systems must produce a response to a question by processing *unstructured* data. This lack of structure and typing, which earlier database QA took advantage of, is reflected in the type of answers that state-of-the-art QA systems provide and that form the basis for their evaluation: That is, a list of strings which have each been extracted from a supporting document. Because strings only map poorly to answers (e.g. distinct strings may actually refer to the same answer), the end user of a QA system may have difficulty to interpret the variety of these answer strings. Also, because of the lack of structure and typing (some strings can refer to the same answer, while other are the witnesses of alternative answers), the results of current QA systems are not directly reusable by other applications for further processing. The two core motivations for analyzing answer complexity are stepping towards more user-oriented answers, as well as re-usability of QA results.

Despite the fact that more and more systems are using data-driven approaches that take account of such answer characteristics as redundancy, and despite recent work on *answer complexity*, the trend in open domain QA is still to find a "correct" answer to more and more "complex" questions. For instance, TREC QA requires systems to provide only a single answer to factoid questions. The new TREC ciQA track is meant to move away from these factoid questions in order to "address complex information needs". That is, complexity in QA is mainly seen as a feature of the question. There are "simple" questions, such as factoids, and "complex" questions, usually longer and more elaborate, for which the answer type is more difficult to assess beforehand. This is a view that I strongly disagree with, especially for open domain QA on free text. The answer to a very elaborate question can still be found through simple keyword matching, while answering a "simple" question like *How old is Bill Gates?* from the web is actually hard, because there are multiple divergent claims. And for me, complexity also depends on characteristics of the corpus used for answering (e.g., heterogeneity in web data), not on the type of question alone.

In this thesis, I have shown that, regardless of their complexity, all questions, including factoids, can have multiple answers (Subsection 2.1.2). The current rule in

TREC QA that requires that systems find one and only answer to factoids is not realistic, and worse the training data that such a large evaluation produces is biased by this rule and not representative of the actual complexity among answers. There is an understandable reason for cutting off the number of answers: Evaluating QA answers involves costly human assessment. My hypothesis is that the evaluation of *correctness*, i.e. the truth value of an answer, may actually make the assessment harder than it could be. Judging currently involves checking the supporting document and following guidelines such as if the answer was *true* at the time the document was written, then the answer should be accepted. This is a lengthy task that also relies on the background knowledge of judges or eventually checking external resources.

If one would consider QA as a task similar to summarization, in which the truth value of the summarized facts is not essential as long as the report makes it explicit that facts have been reported differently by distinct sources, then QA evaluation could be reduced to assess the quality of the evidence for a given answer rather the truth value of the answer itself. Providing not only answers but also a justification for each answer would improve QA answers in terms of self-sufficiency: The user should not have to browse a full document (or several documents if the answer comes from fusion) to verify that an answer is justified. Current evaluations are mainly concerned with the correctness of a question-answer pair. As soon one considers multiple answers, it becomes more obvious that multiple sources are involved in the answer retrieval process, and that answers need to be more appropriately justified for the user to understand a system's response.

More generally, processing multiple answers raises core issues in QA that have been ignored in considering each answer independently. Evaluating multiple answers immediately raise the issue of the possible relationships between answers, and how to organize them. Answers appear in many places and in different forms, but they can also be presented in different forms depending on the user's profile. (This again makes answer correctness a very relative notion, because a given user might be more interested in some types answers than others). This also immediately raises the problem of distinguishing between answer content and answer rendering, not only to be able to organize answers by comparing extractions, but also to improve the quality of the final response (e.g. providing a map for location questions could be more informative than just text).

In this thesis, I proposed a first approach to using multiple answers, by fusing answer candidates into a graph model. Fusion consists in inferring two broad relationships: inclusion and equivalence. Whereas equivalence and near-equivalence have been used successfully in frequency counts as a discriminating feature to re-rank answers, I showed that granularity, i.e. inclusion, can serve as a stronger criterion for this purpose. In my first experiment, I showed an improvement of 23% in first ranked answer accuracy (Table 4.7) when using fusion for answering. Because questions often have multiple answers, a strategy that considers answers as potential allies rather than competitors improves the average performance by building a supporting network around an answer. Analyzing the graph models, it appeared that some candidates that are not actual answers could help (1) by linking correct answers to each other and (2) by providing background information that can be used to explain answer multiplicity.

In Chapter 5, I described a new corpus based on TREC QA questions for which I collected and annotated web data. I propose new scoring measures to evaluate candidates on the basis of evidence and exactness, rather than correctness. The ratings also permitted the introduction of clustering measures to assess the value of structured answers, such as answer clusters.

In Chapter 6, I first demonstrated a significant improvement between a view based on frequency as opposed to a view based on redundancy, i.e. making use of more complex answer comparison than trivial equivalence (frequency score). Especially, fusion proved to increase not only question recall (+12%) and MRR (+9%), but also answer recall at rank 10 (+8.5%). I then more specifically worked on answer clustering and rendering to start generating a response that acknowledges answer multiplicity. Instead of ranking a list of extractions, the system generated clusters grouping related answer nodes. I experimented with clusters expressing answer granularity, as well as clusters aiming at distinguishing between the possible alternative answer referents available for a given question. Using evidence and exactness based ratings, I showed that the clusters were relatively precise and exact entities, and fusion was flexible enough to allow the generation of different sorts of clusters.

In the same chapter, I also reported preliminary results with machine learning techniques in order to show, that although I mainly worked with heuristics developed on training data, graph-based features could be used to train an answer classifier with a similar performance.

I conclude this thesis with a call for attention to answer complexity in parallel with the new attention being paid to question complexity. It will equally serve the joint goals of user understanding and user satisfaction.

Appendix A

QA Taxonomies

A.1 The 13 conceptual question categories used by Wendy Lehnert

Reproduced from [Burger et al., 2002].

	Question Categories	Examples
1.	Causal Antecedent	Why did John go to New York?
		What resulted in John's leaving?
		How did the glass break?
2.	Goal Orientation	For what purposes did John take the book?
		Why did Mary drop the book?
		Mary left for what reason?
3.	Enablement	How was John able to eat?
		What did John need to do in order to leave?
4.	Causal Consequent	What happened when John left?
		What if I don't leave?
		What did John do after Mary left?
5.	Verification	Did John leave?
		Did John anything to keep Mary from leaving?
		Does John think that Mary left?

6.	Disjunctive	Was John or Mary here?
		Is John coming or going?
7.	Instrumental/Procedural	How did John go to New York?
		What did John use to eat?
		How do I get to your house?
8.	Concept Completion	What did John eat?
		Who gave Mary the book?
		When did John leave Paris?
9.	Expectational	Why didn't John go to New York?
		Why isn't John eating?
10.	Judgmental	What should John do to keep Mary from leaving?
		What should John do now?
11.	Quantification	How many people are there?
		How ill was John?
		How many dogs does John have?
12.	Feature Specification	What color are John's eyes?
		What breed of dog is Rover?
		How much does that rug cost?
13.	Request	Would you pass the salt?
		Can you get me my coat?
		Will you take out the garbage?

A.2 Arthur Graesser's Taxonomy of Inquiries

Reproduced from [Burger et al., 2002].

	Question	Abstract Specification and Examples
1.	Verification	Is a fact true? Did an event occur?
		Is an F-test a type of statistic? Did it rain yesterday?
2.	Comparison	How is X similar to Y? How is X different from Y?
		In what way is Florida similar to China? How is an F-test different from a t-test?

3.	Disjunctive	Is X or Y the case? Is X, Y, or Z the case? Do the mountains increase or decrease the rain in Oregon? Did he order chicken, beef, lamb of fish?
4.	Concept completion	Who? What? When? Where? What is the referent of a noun argument slot? Where are the large population densities in North America? Who wrote the song? What did the child steal?
5.	Definition	What does X mean? What is the superordinate category and some properties of X? What is a factorial design? What does interaction mean?
6.	Example	What is an example of X? What is a particular instance of the category? What is an example of an ordinal scale? What experiment supports this claim?
7.	Interpretation	How is a particular event interpreted or summarized? Does the graph show a main effect for "A"? What happened yesterday?
8.	Feature specification	What qualitative attributes does entity X have? What is the value of a qualitative variable? What is George like? What color is the dog?
9.	Quantification	What is the value of a quantitative variable? How much? How many? How many rooms are in the house? How much profit was made last year?
10.	Causal antecedent	What caused some event to occur? What state or event causally led to an event or state? How does warm air get to Ireland? Why is the kite going backwards?
11.	Causal consequence	What are the consequences of an event or state? What causally unfolds from an event or state? What happens to the warm winds when they reach the mountains? What are the consequences of double-digit inflation?
12.	Goal orientation	What are the motives behind an agent's actions? What goals inspired an agent to perform an action? Why did Roger move to Chicago? What was the purpose of the city's cutting taxes?
13.	Enablement	What object or resource enables an agent to perform an action? What device allows you to measure an earthquake? What do I need to bake this fish?

14.	Instrumental/Procedural	How does an agent accomplish a goal? What instrument or body part is used when an agent performs an action? What plan of action accomplishes an agent's goal? How does a person perform long division? How do you move a mouse on a computer?
15.	Expectational	Why did some expected event not occur? Why wasn't there a war in Iraq? Why doesn't this doll have a mouth?
16.	Judgmental	The questioner wants the answerer to judge an idea or to give advice on what to do. What do you think about the new taxes? What should I do to stop the fight?
17.	Assertion	The speaker expresses that he or she is missing some information. I don't understand what this message on the computer means. I need to know how to get to the Newark airport.
18.	Request/Directive	The speaker directly requests that the listener supply some information Please tell me how to get a printout of this file.

A.3 The Taxonomy of Forms of Expected Answers

Reconstruction taking into account variations around the LIBGIS scheme [Pomerantz, 2005].

Directional	Questions asking about the location of a specific information source.
Holdings	Questions about whether a specific information source or document is owned by the library.
Ready reference	Questions asking for simple, factual answers; the answer should be readily ascertainable from available information sources.
Exact reproduction	Questions asking for pictorial and textual materials, taken directly from an information source and unchanged.
Description	Questions asking for a description of something, briefer in length than the original thing (basically, an abstract).

Readers advisory	Questions asking for assistance in the choice of books or the gathering of data.
Bibliographic instruction	Questions asking for assistance in use of information source(s).
Research	Questions asking for involved answers; the answer should require some effort and wide use of information sources to formulate.
Citation list	Questions asking for a list of information sources on a particular subject.
Analysis	Questions asking for some form of data analysis, whatever that data might be scientific, social, financial, etc. Questions of this type might ask for trends, pro or con arguments, cause and effect, compare and contrast, etc.
Critique	Questions asking for an evaluative discussion of a particular subject. (E.g.: a movie review, Cliffs notes-like analyses of a book, etc.)

A.4 The Taxonomy of Answer Sources

The essential librarian task, according to [Richardson, 1995], is the classification of a question according to the following genres (overlap may occur):

1. Abstracts
2. Atlases
3. Bibliographies
4. Biographical sources
5. Dictionaries
6. Directories
7. Encyclopedias
8. Government publications

9. Handbooks / manuals
10. Indexes
11. Statistical sources
12. Yearbooks

A.5 Webclopedia Question-Answer Typology

This is a partial reproduction of question answer typology developed the USC Information Sciences Institute [Hovy et al., 2001, E. H. Hovy, 2002]. The full version is available on-line at *http://www.isi.edu/natural-language/projects/webclopedia/Taxonomy/taxonomy_toplevel.html*.

- Relational qtargets (question targets)
  - R-LOCATION
  - R-TIME
    - R-EVENTS
    - R-BIRTHDAY
  - R-POPULATION
  - R-PERSONS
  - R-ABBREVIATION
- Abstract qtargets
  - A-WHY-FAMOUS
  - A-DEFINITION
  - A-SYNONYM
  - A-CAUSE-OF-DEATH
- Semantic qtargets
  - C-PROPER-NAMED-ENTITY
    - C-ROPER-PERSON
    - C-PROPER-PLACE
      - C-CONTINENT
      - C-PROPER-UNIVERSITY
  - C-QUANTITY

        * C-MONETARY-QUANTITY
        * C-SPEED-QUANTITY

- Syntactic qtargets
    - S-NP, S-NOUN
    - S-VP
    - S-PROPER-NAME
- Role qtargets
    - ROLE REASON
    - ROLE MANNER
- Slot qtargets
    - SLOT TITLE
    - SLOT QUOTE
- Lexical qtargets: Lexical qtargets are used when the answer is already available from some external knowledge, and all the system still has to do is look for text supporting that answer.
- Combinations of qtargets

Appendix B

Answer Evaluation Guidelines

This document provides definitions and guidelines for manually assessing answer candidates to a given set of questions.

Answer candidates were automatically extracted from the 10 first snippets corresponding to the first screen page of Google (January 2007) queried with the question string. The evaluation is meant to assess (1) the quality of the evidence (snippet) supporting an answer candidate, and (2) the exactness of the answer string. The particularity of this evaluation is that the assessor does not evaluate the *correctness* of the answer, i.e. some universal truth value attached to the answer, but rather the level of *exactness* of the answer candidate and the *quality and strength of the evidence* supporting the answer candidate.

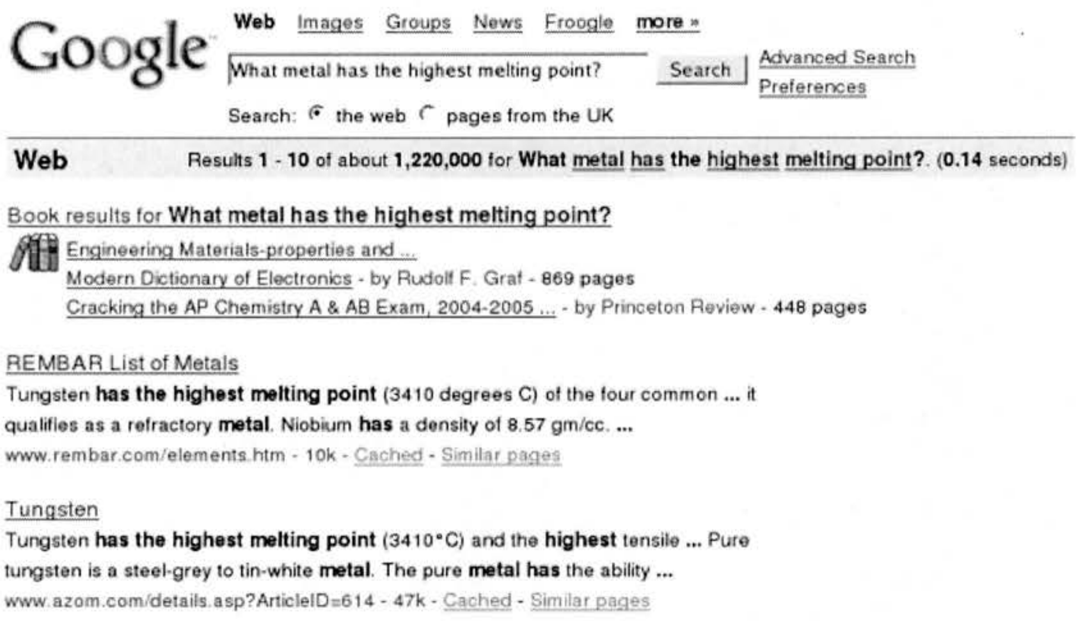
The first section provides several definitions and introductory notes to facilitate further references to the material. The second section specifically describes the assessment task to be performed and the corresponding marking tool. The third section provides examples and guidelines for the task. The last section list the 50 test questions and their exact and strongly supported answers.

B.1 Definitions

This section reviews definitions for three entities: questions, snippets and answer candidates, and the two aspects to be rated for this evaluation: the quality of the snippet as evidence and the exactness of an answer candidate.

Question - In the material to be evaluated, the question is a string expressing a knowledge gap to be filled with some information, e.g. *Who invented the telephone?* or *Name a food high in zinc.* A question can have zero, one or more satisfying answers. The notion of *satisfying* answer is described in the following section.

Snippet - A snippet is a short piece of text made of excerpts from a web document. When querying a web search engine, results are usually organized in a list of snippets pointing to actual documents. A snippet typically stands for the "summary" of a document. It usually consists of a title, a body and a link to the document, as shown on Figure B.1.


Google
Web Images Groups News Froogle more »
What metal has the highest melting point?
Search
Advanced Search
Preferences
Search: the web pages from the UK
Web

Results 1 - 10 of about 1,220,000 for What metal has the highest melting point?. (0.14 seconds)

Book results for What metal has the highest melting point?
Engineering Materials-properties and ...
Modern Dictionary of Electronics - by Rudolf F. Graf - 869 pages
Cracking the AP Chemistry A & AB Exam, 2004-2005 ... - by Princeton Review - 448 pages

REMBAR List of Metals
Tungsten **has the highest melting point** (3410 degrees C) of the four common ... it qualifies as a refractory **metal**. Niobium **has** a density of 8.57 gm/cc. ...
www.rembar.com/elements.htm - 10k - Cached - Similar pages

Tungsten
Tungsten **has the highest melting point** (3410°C) and the **highest** tensile ... Pure tungsten is a steel-grey to tin-white **metal**. The pure **metal has** the ability ...
www.azom.com/details.asp?ArticleID=614 - 47k - Cached - Similar pages

Figure B.1: Top 2 snippets provided by Google for the query "What metal has the highest melting point?".

Because a snippet is made of excerpts, the text is not necessarily well-formed, coherent or explicit. This will be an important characteristic when assessing the quality of the snippet.

Answer candidate - An answer candidate is a string extracted from a snippet. In the material described here, it usually corresponds to a nominal keyword or a phrase. An answer candidate is part of the pool of possible answers to a question, i.e. the list of different strings automatically extracted from all the snippets found for that question.

An answer candidate is always attached to a snippet. It is never evaluated on its own, but with respect to its source snippet, and only its source snippet. The same answer candidate may appear in another snippet, it should then be evaluated in the context of the other snippet.

Because the extraction of an answer candidate from a snippet relies on an automated process, the extracted string is not necessarily well-formed or exact. The extraction process may also miss keywords or phrases that could be relevant answers to the question. Thus it may happen that a snippet, as a whole, could be found to be relevant to a given question, but none of the answer candidates extracted from that snippet are actually relevant.

Also, it is important to note that the extraction process merges identical strings together. Thus, if the string *tungsten* appears twice in the same snippet, it will appear only once in the set of answer candidates for that snippet. The evaluation of the answer candidate *tungsten* will be based on its two occurrences in the snippet.

Evidence - Evidence is the textual basis for belief or disbelief. In this evaluation, the source **snippet** serves as evidence. It may be seen as a textual justification that would argue in favour of the textual content of the snippet as an actual answer to the question.

Assessors may have their own set of beliefs about what does and does not constitute a "good" answer, independently of the source of snippet. However, the goal of the evaluation is not to assess the snippets and the answer candidates with respect to such beliefs, but with respect to the evidence provided "as is" by the snippet. Hence the judgment is not on the correctness of the snippet content but on the quality of the evidence. Agreement may vary as to what degree of explicitness is required to admit that the evidence indeed supports an answer, but the judgment should not be made on the basis of one's beliefs about the answer itself. Thus, this evaluation allows positive ratings for beliefs or opinions that may be false, incorrect or inexact, but for which the textual evidence is strong.

For instance a snippet containing the following string *tungsten has the highest melt-*

ing point (in response to the question *What metal has the highest melting point?*) is explicit enough to strongly support the answer candidate *tungsten* as a satisfying answer. So is the string *niobium has the highest melting point* to support *niobium*, although one may know that this belief about niobium is mistaken. Similarly, a snippet that only describes tungsten without specifying it has indeed the highest melting point does not provide any strong basis for the belief that tungsten is the answer to the question. The rationale for a rating of evidence quality rather than answer correctness *per se* is that automated question answering processes are going to be evaluated on their mining quality rather their ability to decide whether a statement is universally true or false.

Evidence is evaluated for each snippet, as a whole, independently from the list of extracted answer candidates, and also for each answer candidate in relationship to its source snippet. The two evidence-based ratings are further detailed in the next section.

Exactness - Exactness is a rating that measures the quality of the automated process that extracts answer candidates from snippets. As such, it only applies to answer candidates, not to snippets.

This notion of exactness is close, in part, to the notion of exactness described in the TREC QA task [Voorhees, 2002]. For instance, in TREC QA, the following answers: *Mississippi, mississippi, the Mississippi River* to the question *What is the longest river in the United States?* are considered correct and exact answers while none of the following are considered exact: *At 2,348 miles the Mississippi River is the longest river in the US., 2,348 miles; Mississippi.* This string-based requirement assesses the quality of pin-pointing the precise span that corresponds to an answer. It is meant to avoid (1) extraneous, possibly redundant or irrelevant, information, be it punctuation or words, or (2) the opposite, which is not given as an example in TREC QA, i.e. cases when the extracted span is not complete, for instance *2,348* is not exact because the unit is missing.

This is the first aspect of exactness rating, and it is essentially a string-based criterion for which it is possible to set specific arbitrary rules specifically defining what an exact string is. (The rules are detailed in the last section.)

The second aspect evaluated under exactness is the level of specificity of a given answer candidate. For instance, the candidates *world* or *Earth* may not be considered informative enough to answer the question *Where is the Taj Mahal?* in a satisfying manner. Exactness here is taken in the sense of precise degree of informativity. Such exactness may vary depending on the question. For instance, *Mars* is an exact enough answer to the question *Where is the volcano Olympus Mons located?*.

B.2 Assessment Task

The task consists in performing two assessments:

1. The evaluation of each snippet found for a given question.
2. The evaluation of answer candidates found for each snippet.

The first evaluation is focused on each snippet only. The objective is to assess the value of the snippet as **evidence**, independently from the answer candidates that have automatically extracted from it. Because the extraction process sometimes misses out relevant candidates, this first assessment provides an indication of the value of the raw material (unprocessed snippet).

The judgment consists in assessing whether the snippet provides a well-supported answer to the question, i.e. a good quality evidence.

The second evaluation aims at assessing the answer candidates automatically extracted from each snippet. Judges are requested to evaluate two characteristics of each answer candidate in the context of its source snippet: its **exactness** as a string, and whether the candidate is well supported by the source snippet, i.e. a measure of the **evidence** for the candidate.

It is important to note that the later evidence rating (for answer candidate) is relational: It evaluates the quality of the snippet as evidence *for* a given answer candidate. Indeed, a snippet could actually contain several relevant answer candidates, but provide strong evidence for only one candidate. In other words, the textual content of the snippet may provide a good justification for one candidate, but a rather poor one for another candidate. Thus, the evidence rating may be different from one candidate to another, although both have been extracted from the same snippet. The candidate evidence rating may also be different from the snippet evidence rating, i.e. a rating for the value of the snippet on its own, independently from candidate extraction.

Table B.1 summarizes the ratings for snippets and answer candidates.

Table B.1: Ratings for snippets and answer candidates

	Evidence			**Exactness**	
Snippet	None	Weak	Strong	NA	
Answer candidate	None	Weak	Strong	Exact	Inexact

For both snippets and answer candidates, the value of the **evidence** can be rated as *strong* (good evidence) or *weak* (some evidence), or eventually set to *none* if the snippet does not support the answer. **Exactness** is not measured for snippets since they are evaluated as a whole, independently from the extraction process. The **exactness** of answer candidates only applies to those candidates that are supported by a weak or strong evidence. The string is judged either *exact* or *inexact*.

The following section provides guidelines and examples as to what may be considered a *weak* or a *strong* evidence, and an *exact* or *inexact* answer candidate.

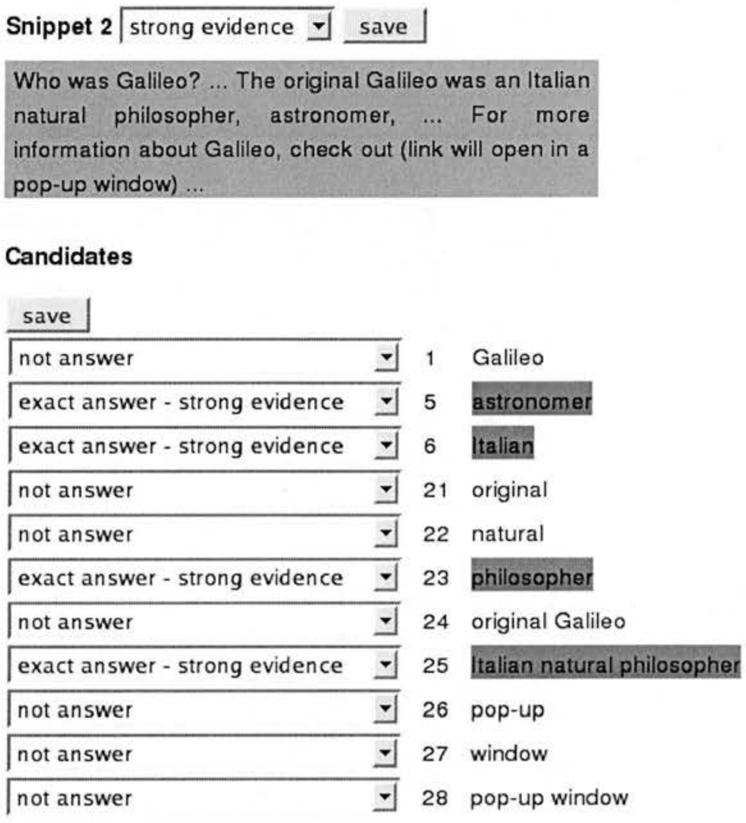
B.3 Guidelines and Examples

A web application is provided to rate snippets and answer candidates for each question. The index page provides the list of questions to be evaluated, each pointing to the specific rating page listing snippets and answer candidates.

It is advised to first rate the quality of the evidence provided by each snippet, as a whole, and then to assess each answer candidate in relationship to its source snippet.

The top part of Figure B.2 shows the rating form for the second snippet found to the question *Who was Galileo?*. The selection menu proposes three ratings: *no evidence*, *weak evidence* and *strong evidence*. Rating guidelines are provided in the first subsection below.

Once snippet have been evaluated, the list of answer candidates extracted for each snippet can be reviewed and rated. On Figure B.2 , four candidates have to be *exact* and well-supported (*strong* evidence) answer candidates.



Snippet 2 strong evidence | save

Who was Galileo? ... The original Galileo was an Italian natural philosopher, astronomer, ... For more information about Galileo, check out (link will open in a pop-up window) ...

Candidates

save

not answer	1	Galileo
exact answer - strong evidence	5	astronomer
exact answer - strong evidence	6	Italian
not answer	21	original
not answer	22	natural
exact answer - strong evidence	23	philosopher
not answer	24	original Galileo
exact answer - strong evidence	25	Italian natural philosopher
not answer	26	pop-up
not answer	27	window
not answer	28	pop-up window

Figure B.2: Evaluation web form for the second snippet and corresponding answer candidates found to *Who was Galileo?*

You may notice that the extraction process process may generate partly redundant spans (e.g. *Italian natural philosopher* and *philosopher*, or miss out some keywords or phrases. This is the reason why sometimes it may happen that a snippet represents strong evidence, but no answer candidates may actually be relevant answers.

Answer candidates can be rated as follows:

- no answer
- inexact answer - weak evidence

- inexact answer - strong evidence
- exact answer - weak evidence
- exact answer - strong evidence

Again, it is important to note that the answer candidate evidence rating is relative to the corresponding source snippet. The same answer candidate may be rated *exact answer - strong evidence* for a given snippet, but *exact answer - weak evidence* or *no answer* in other snippets because the evidence is unsatisfying. Guidelines for exactness and evidence ratings are provided in the second subsection.

B.3.1 Evidence Rating for Snippets and Answer Candidates

Table B.2 provides a commented example of evidence rating for snippets. As a general, a snippet must contain at least a satisfying answer, and provides explicit enough support for the answer, to be rated as strong.

A snippet can be considered weak evidence, when it contains a satisfying answer but the context is not explicit enough, requiring for instance the reader to make an assumption or to draw an inference from his/her own knowledge, or on the basis of the content of other snippets. Note that the rating assesses the evidence should assess the evidence, not the exactness of the answer considered.

Snippets that do not contain any answer at all, even if they are related to the topic, or provide some additional context, should be marked with a 'no evidence' flag.

For answer candidates, the evidence is assessed for the candidate in relationship to the snippet. A snippet can provide strong evidence for a candidate, but only weakly support another.

For instance, the following snippet found to the question *What are Quaaludes?*

Methaqualone - Wikipedia, the free encyclopedia ... In the sitcom The King of Queens, after doing a favor for Doug, Supervisor O'Boyle mentions, "I wouldn't mind if a handful of Mexican quaaludes found their ...

can be rated as strong evidence because it explicitly indicates that Quaaludes can be of Mexican origin, and this is a relevant piece of information to answer the question.

Table B.2: Ratings for snippets and answer candidates

What are Quaaludes?	
Rating	**Snippet/comment**
Strong	*interview - quaaludes ... Such is the mystery of methaqualone, the downer that stamped its mark on the 1970s under its altogether more sexy American trade name: Quaaludes. ...*
	The snippet explicitly refers to Quaaludes as a trade name for the substance known as methaqualone. It also indicates its depressant quality. The evidence can be rated as strong.
Weak	*methaqualone: Definition and Much More from Answers.com ... On cult television show Strangers With Candy, the main character Jerri Blank talks about 'the good old days' and how no one makes good quaaludes anymore. ...*
	One may infer that methaqualone is the term defined in the document and that the excerpt mentioning Quaaludes allows to associate both terms. The assessor may also know, from self-knowledge or other snippets that indeed the word Quaaludes is a name for methaqualone. However, because this requires making an assumption and involves some inference, the evidence is only rated as weak.
None	*Ask Erowid : ID 143 : What are the effects of quaaludes? ... Ask Erowid Question and Answer: What are the effects of quaaludes?*
	The snippet does not contain any relevant information for the given question. It is thus marked as providing no evidence.

The snippet also indicates methaqualone, however it is not explicit that Quaaludes are another name for the substance. Thus, among the following sample list of candidates: *Methaqualone* is assessed as an exact answer, based on weak evidence, *Mexican* is rated as an exact answer, based on strong evidence, and *sitcom* for instance would be marked as not an answer. An extraction such as *a handful of Mexican quaaludes* would be marked as inexact, based on strong evidence.

As a rule, if accepting the answer requires inference of some knowledge, either from other snippets, or personal knowledge, the evidence should be rated as weak, if the snippet does provide the basis for some inference. Otherwise, the answer should not be accepted for the given snippet.

Sometimes, candidates, as standalone extractions, may be ambiguous. For instance, to *Who is Duke Ellington?*, *black*, outside its context is a satisfying answer. The source snippet contains the following string: *The black community in Ellington's time and a tour of the Shaw neighborhood where he grew up (..).* In this context, it is possible to assume that the statement is made because Duke Ellington was a *black* man. However, this is an assumption, somewhat backed up by personal knowledge about jazz men, and the other snippets. Therefore the evidence for *black* should be at best *weak*. Similarly, the candidate *black community* is acceptable as an answer, but only on weak evidence. Now *black* can be considered exact, while *black community* is not specific enough. It should be indicated that Ellington belonged to the black community or that he was a black man, but the candidate as it is lacks of exactness. I review a few guidelines for exactness in the following section.

B.3.2 Exactness Rating for Answer Candidates

As a general rule, answer extractions should be rated exact only if they fit exactly and precisely as answers. They should not contain extra, noisy information. They should also contain all the necessary information to be used as answers outside their context. If they are not understandable without context, or if they require some inference to be understood, they should be marked as inexact.

For instance, to the question *What is acupuncture?*, the candidate *needles* is judged inexact, as opposed to *Chinese therapy*. This is because, although acupuncture indeed involves needles, the term *needles* does not exactly provide a definition. In TREC QA, *needles* was accepted as a "correct" answer. In this evaluation, it was judged as inexact but strongly supported (because the context snippet is explicit about the role of needles in acupuncture).

Biography questions have a similar ratio between EXACT-STRONG and lenient.

For instance, to *Who is Duke Ellington?*, the candidate *artist* can be judged exact, and strongly supported. On the other hand, *Jazz* is inexact (but strongly supported). A date, on its own, should be rated inexact, unless the extraction specifies that it is for instance a birth date, i.e. a date is only exact when the associated event is explicit.

For named entities, arbitrary rules have been designed as follows. For person names, the surname is exact. The full name is also exact. The first name only is inexact. For dates, usually years, or full time expressions (day+month+year, or month+year) are exact, while a day or a month should be considered inexact, unless the question asks for a month, or a recurring event, such as summer solstice, in which case an extraction such as *21 June* is exact. *June* alone would be inexact because the solstice corresponds to a day. A descriptive definition for some time expressions are also acceptable as exact (e.g. the solstice is the *longest day*). Finally, for place names, particular attention should be paid to multiple word expressions. A name such as *New York* is exact, *New* or *York* in that context are not answers. On the other hand, an extraction such as *Lhotse* can be considered as exact as *Lhotse*. But, in general, if a multiple word expression has been improperly tokenized, it should be rated as inexact if not discarded. Especially, if a string span has been rated inexact, a substring of this span should be considered even more inexact, and it is probably best to discard it as an answer.

Again, the level of specificity of a given answer candidate should also be assessed as exactness. This may vary depending on the question. For instance, the candidates *world* or *Earth* may not be considered informative enough to answer the question *Where is the Taj Mahal?* in a satisfying (exact) manner, but *Mars* is an exact enough answer to the question *Where is the volcano Olympus Mons located?*.

Finally, when a candidate extraction contains a question word, its exactness depends on the phrasing. For instance, *acupuncture therapy* is acceptable as an exact answer to *What is acupuncture?*, although the question term is a bit redundant. Misspellings could also be acceptable as exact answers. This is left to each annotator's judgment.

B.4 Dataset

The following represents the list the 50 test questions with their exact and strongly supported answer strings (only distinct strings are shown, taking into account case differences), as referred to in Chapters 5 and 6. Inexact and weakly supported answers are not shown here.

What are capers? biennial
What are capers? biennial spiny shrub
What are capers? bud
What are capers? buds
What are capers? Capparis spinosa
What are capers? condiment
What are capers? flower bud
What are capers? green
What are capers? Ingredient
What are capers? ingredients
What are capers? Mediterranean
What are capers? plants
What are capers? shrub
What are capers? spiny
What are capers? unopened green flower buds
What are Quaaludes? depressants
What are Quaaludes? downer
What are Quaaludes? epidemic
What are Quaaludes? illegal
What are Quaaludes? ludes
What are Quaaludes? meth
What are Quaaludes? methaqualone
What are Quaaludes? Methaqualone
What are Quaaludes? Mexican
What are Quaaludes? Mexican quaaludes

What are Quaaludes? prescription

What are Quaaludes? Quaalude-addicted

What color is indigo? blue

What color is indigo? Blue

What color is indigo? purple

What do bats eat? 3000 mosquitoes each night

What do bats eat? bugs

What do bats eat? fruit

What do bats eat? Fruit

What do bats eat? Insect-eating

What do bats eat? insects

What do bats eat? Insects

What do bats eat? mosquitoes

What do bats eat? small flying night bugs

What do bats eat? spiders

What does the word fortnight mean? fourteen days

What does the word fortnight mean? ten days

What does the word fortnight mean? two weeks

What gasses are in the troposphere? aerosol

What gasses are in the troposphere? aerosol precursor gasses

What gasses are in the troposphere? aerosols

What gasses are in the troposphere? carbon dioxide

What gasses are in the troposphere? CH4

What gasses are in the troposphere? CO2

What gasses are in the troposphere? H20

What gasses are in the troposphere? Heat-trapping

What gasses are in the troposphere? Heat-Trapping

What gasses are in the troposphere? Heat-trapping gasses

What gasses are in the troposphere? Heat-trapping Gasses

What gasses are in the troposphere? Heat-Trapping Gasses

What gasses are in the troposphere? irritation gases

What gasses are in the troposphere? methane

What gasses are in the troposphere? N20

What gasses are in the troposphere? N2O

What gasses are in the troposphere? nitrous oxide

What gasses are in the troposphere? ozone

What gasses are in the troposphere? Smog

What gasses are in the troposphere? water vapor

What is acupuncture? acupuncture therapy

What is acupuncture? ancient

What is acupuncture? ancient system

What is acupuncture? Chinese

What is acupuncture? Chinese therapy

What is acupuncture? healing

What is acupuncture? medicine

What is acupuncture? Oriental medicine

What is acupuncture? practice

What is acupuncture? therapeutic

What is acupuncture? therapeutic purposes

What is acupuncture? therapy

What is acupuncture? treatment

What is Africa's largest country? Sudan

What is amoxicillin? antibiotic

What is amoxicillin? antibiotics

What is amoxicillin? drug

What is amoxicillin? Drug

What is amoxicillin? drugs

What is amoxicillin? Drugs

What is amoxicillin? medications

What is amoxicillin? penicillin-like

What is amoxicillin? penicillin-like antibiotics

What is amoxicillin? penicillins

What is a shaman? Ayahuasca

What is a shaman? Ayahuasca Visions

What is a shaman? ecstatic

What is a shaman? ecstatic experiences

What is a shaman? Hawaiian

What is a shaman? Hawaiian shamanism

What is a shaman? heal

What is a shaman? healer

What is a shaman? intermediary

What is a shaman? Peruvian

What is a shaman? Religious

What is a shaman? visions

What is a shaman? Visions

What is a thyroid? gland

What is a thyroid? Gland

What is a thyroid? physiological functions

What is a thyroid? thyroid gland

What is a thyroid? Thyroid Gland

What is autism? aspergers

What is autism? autism spectrum

What is autism? autism spectrum disorders

What is autism? Autism Spectrum Disorders

What is autism? complex

What is autism? complex brain disorder

What is autism? condition

What is autism? disability

What is autism? disorder

What is autism? disorders

What is autism? Disorders

What is autism? lifelong

What is autism? lifelong developmental disability

What is autism? perplexing

What is autism? perplexing condition

What is autism? Pervasive

What is autism? Pervasive Developmental Disorders

What is autism? spectrum

What is autism? Spectrum

What is autism? spectrum disorder

What is autism? unknown

What is bipolar disorder? Bipolar (Manic-Depressive) disorder

What is bipolar disorder? depression

What is bipolar disorder? Depression

What is bipolar disorder? depressive

What is bipolar disorder? diagnosis

What is bipolar disorder? dramatic mood swings

What is bipolar disorder? illness

What is bipolar disorder? irritable

What is bipolar disorder? manic

What is bipolar disorder? Manic

What is bipolar disorder? manic depression

What is bipolar disorder? manic-depression

What is bipolar disorder? Manic Depression

What is bipolar disorder? Manic-Depression

What is bipolar disorder? manic-depressive

What is bipolar disorder? Manic-Depressive

What is bipolar disorder? manic depressive illness

What is bipolar disorder? manic-depressive illness

What is bipolar disorder? mental

What is bipolar disorder? Mental

What is bipolar disorder? Mental Health

What is bipolar disorder? mood

What is cryogenics? cold

What is cryogenics? cold temperatures

What is cryogenics? cryopreservation

What is cryogenics? freezing

What is cryogenics? icy

What is cryogenics? icy cold

What is cryogenics? low-temperature

What is cryogenics? low temperatures

What is cryogenics? low-temperature state

What is cryogenics? methods

What is cryogenics? science

What is cryogenics? study

What is myopia? common

What is myopia? common vision defect

What is myopia? Genetic

What is myopia? Genetic factors

What is myopia? nearsightedness

What is myopia? Nearsightedness

What is myopia? Near-sightedness

What is myopia? short-sightedness

What is myopia? Short-sightedness

What is myopia? vision defect

What is naproxen? Aleve

What is naproxen? Anaprox

What is naproxen? anti-inflammatory

What is naproxen? drug

What is naproxen? Drug

What is naproxen? drugs

What is naproxen? Drugs

What is naproxen? Gastrointestinal Side Effects

What is naproxen? medication

What is naproxen? medicines

What is naproxen? Naprelan
What is naproxen? Naprosyn
What is naproxen? naproxen sodium
What is naproxen? nonsteroidal
What is naproxen? nonsteroidal anti-inflammatory drugs
What is naproxen? NSAIDs
What is peyote? active ingredient
What is peyote? bead
What is peyote? Beading
What is peyote? Beading Term
What is peyote? beads
What is peyote? cacti
What is peyote? cactus
What is peyote? Cactus
What is peyote? divine cactus
What is peyote? drugs
What is peyote? Hallucinogens
What is peyote? mescaline
What is peyote? particular bead
What is peyote? peyote cactus
What is peyote? Peyote Ceremony
What is peyote? Peyote Stitch
What is peyote? plant
What is peyote? slowest growing cacti
What is peyote? soft-drugs
What is peyote? stitch
What is peyote? Stitch
What is sodium chloride? chemical
What is sodium chloride? Chemical
What is sodium chloride? chemicals
What is sodium chloride? colorless

What is sodium chloride? common salt

What is sodium chloride? Common Salts

What is sodium chloride? common table salt

What is sodium chloride? compound

What is sodium chloride? crystal

What is sodium chloride? crystalline

What is sodium chloride? crystal structure

What is sodium chloride? cubic

What is sodium chloride? cubic lattice

What is sodium chloride? food preservative

What is sodium chloride? gaseous NaCl

What is sodium chloride? halite

What is sodium chloride? Halite

What is sodium chloride? HALITE

What is sodium chloride? lattice

What is sodium chloride? mineral

What is sodium chloride? MINERAL

What is sodium chloride? mineral halite

What is sodium chloride? mineral specimens

What is sodium chloride? NaCl

What is sodium chloride? normal solid state

What is sodium chloride? preservative

What is sodium chloride? regular table salt

What is sodium chloride? salt

What is sodium chloride? Salt

What is sodium chloride? Salts

What is sodium chloride? seasoning

What is sodium chloride? solid

What is sodium chloride? white

What is sodium chloride? white crystalline compound

What is Teflon? binding resins

What is Teflon? coating
What is Teflon? coatings
What is Teflon? Coatings
What is Teflon? DuPont product
What is Teflon? DuPont technology
What is Teflon? DuPont Teflon non-stick coatings
What is Teflon? ethylene
What is Teflon? fluorinated ethylene
What is Teflon? fluorocarbon
What is Teflon? fluorocarbon coatings
What is Teflon? non-stick
What is Teflon? non-stick coatings
What is Teflon? polytetrafluoroethylene
What is Teflon? Polytetrafluoroethylene
What is Teflon? Protector
What is Teflon? registered trademark
What is Teflon? repellency
What is Teflon? resins
What is Teflon? substance
What is Teflon? Teflon non-stick coatings
What is the capital of Ethiopia? Addis Ababa
What is the capital of Mongolia? Ulaanbaatar
What is the capital of Mongolia? Ulaan Baatar
What is the capital of Mongolia? Ulan Bator
What is the capital of Persia? persepolis
What is the capital of Persia? Persepolis
What is the capital of Persia? Tehran
What is the capital of Syria? Damascus
What is the capital of Zimbabwe? Harare
What is the coldest place on earth? Anarctica
What is the coldest place on earth? Antarctic

What is the coldest place on earth? Antarctica
What is the coldest place on earth? Earth Dome A
What is the coldest place on earth? laboratory
What is the coldest place on earth? Leiden
What is the coldest place on earth? physics laboratory
What is the coldest place on earth? Russian Vostok Station
What is the coldest place on earth? Vostok
What is the coldest place on earth? Vostok Antarctica
What is the currency of Bolivia called? BOB
What is the currency of Bolivia called? BOB.
What is the currency of Bolivia called? Bolivian Boliviano
What is the currency of Bolivia called? boliviano
What is the currency of Bolivia called? Boliviano
What is the currency of Bolivia called? Bolivianos
What is the currency of Bolivia called? centavos
What is the currency of Bolivia called? cents
What is the currency of Bolivia called? dollar
What is the currency of Bolivia called? Standard dollar sign
What is the currency used in China?
What is the currency used in China? chiao/jiao
What is the currency used in China? Chinese Yuan
What is the currency used in China? CNY
What is the currency used in China? fen
What is the currency used in China? Renminbi
What is the currency used in China? Renminbi Yuan
What is the currency used in China? RMB
What is the currency used in China? RMBY
What is the currency used in China? Yiao
What is the currency used in China? Yuan
What is the fourth highest mountain in the world? Lhotse
What is the fourth highest mountain in the world? Mountain Climb Lhotse

What is the fourth highest mountain in the world? Mount Lhotse

What is the name of the country in the Pyrenees mountains between France and Spain? Andorra

When is the summer solstice? June 21

When is the summer solstice? longest day

When was Hiroshima bombed? 1945

When was Hiroshima bombed? August 6 , 1945

When was the first liver transplant? 1963

When was the first liver transplant? 1964

When was the telephone invented? 1876

When was the telephone invented? 1912

When was the telephone invented? 1941

When was the telephone invented? February 1912

When were the first postage stamps issued in the United States? 1847

Where is the Euphrates River? Ancient Babylonia

Where is the Euphrates River? Armenian plateau

Where is the Euphrates River? Asia

Where is the Euphrates River? Babylonia

Where is the Euphrates River? Iraq

Where is the Euphrates River? Middle East

Where is the Euphrates River? Persian Gulf

Where is the Euphrates River? southwest Asia

Where is the Euphrates River? Syria

Where is the Euphrates River? Turkey

Where is the Euphrates River? western Asia

Where is the Lourve? France

Where is the Lourve? Paris

Where is the volcano Olympus Mons located? Mars

Where is the volcano Olympus Mons located? Martian

Where is the volcano Olympus Mons located? planet Mars

Where is the volcano Olympus Mons located? Tharsis

Where is the volcano Olympus Mons located? Tharsis Buldge
Where is the volcano Olympus Mons located? Tharsis Plateau
Which African country's major export is coffee? Ethiopia
Which African country's major export is coffee? Kenya
Which African country's major export is coffee? Rwanda
Who developed the vaccination against polio? Dr. Albert Sabin
Who developed the vaccination against polio? Jonas Salk
Who developed the vaccination against polio? Sabin
Who developed the vaccination against polio? Salk
Who discovered America? Ancient Hebrew Explorers
Who discovered America? ancient Hebrews
Who discovered America? China
Who discovered America? Christopher Columbus
Who discovered America? Columbus
Who discovered America? Ericson
Who discovered America? Hebrew
Who discovered America? Hebrews
Who discovered America? Leif Ericson
Who discovered radium? Curie
Who discovered radium? French
Who discovered radium? French chemist
Who discovered radium? French physicist
Who discovered radium? French physicist Marie Curie
Who discovered radium? Madame Curie
Who discovered radium? Marie Curie
Who discovered radium? Marie Sklodowska Curie
Who discovered radium? Nobel laureate Marie Curie
Who discovered radium? Pierre Curie
Who discovered radium? Polish
Who discovered radium? Polish chemist
Who discovered radium? Polish Scientist

Who discovered radium? Sklodowska

Who invented the instant Polaroid camera? Edwin Herbert Land

Who invented the instant Polaroid camera? Edwin Land

Who invented the instant Polaroid camera? Land

Who invented the telephone? Alexander Graham Bell

Who invented the telephone? ANTONIO MEUCCI

Who invented the telephone? Bell

Who invented the telephone? German

Who invented the telephone? German scientist

Who invented the telephone? Meucci

Who invented the telephone? MEUCCI

Who is a German philosopher? Ernst Tugendhat

Who is a German philosopher? Ficthe

Who is a German philosopher? Friedrich Nietzsche

Who is a German philosopher? Heidegger

Who is a German philosopher? Immanuel Kant

Who is a German philosopher? Kant

Who is a German philosopher? Martin Heidegger

Who is a German philosopher? Marx

Who is a German philosopher? Nietzsche

Who is a German philosopher? Schelling

Who is a German philosopher? Tugendhat

Who is Duke Ellington? Artist

Who is Duke Ellington? composer

Who is Duke Ellington? prolific composer

Who is the governor of Colorado? Bill Owens

Who is the governor of Colorado? Bill Ritter

Who is the governor of Colorado? Democrat

Who is the governor of Colorado? Governor Bill Ritter

Who is the governor of Colorado? Owens

Who is the governor of Colorado? Ritter

Who is the governor of Colorado? Romer

Who is Tom Cruise married to? Holmes

Who is Tom Cruise married to? Katie Holmes

Who is Tom Cruise married to? Kidman

Who is Tom Cruise married to? Nicole Kidman

Who was Abraham Lincoln? 16th president

Who was Abraham Lincoln? 1861-1865

Who was Abraham Lincoln? president

Who was Abraham Lincoln? President

Who was Abraham Lincoln? sixteenth President

Who was elected president of South Africa in 1994? Mandela

Who was elected president of South Africa in 1994? Nelson Mandela

Who was elected president of South Africa in 1994? Nelson R. Mandela

Who was Galileo? astronomer

Who was Galileo? Astronomer

Who was Galileo? Italian

Who was Galileo? Italian natural philosopher

Who was Galileo? Italian physicist

Who was Galileo? philosopher

Who was Galileo? physicist

Who was Galileo? Physicist

Appendix C

List of Publications

C.1 TREC QA Track and CLEF Task

1. G.de Chalendar, T.Dalmas, F. Elkateb-Gara, O. Ferret, B. Grau, M. Hurault-Plantet, G. Illouz, L. Monceaux, I. Robba, and A. Vilnat. *The Question-Answering System Qalc at LIMSI: Experiments Using the Web and WordNet*, In proceedings of TREC 11, 2002.

2. J. L. Leidner, J. Bos, T. Dalmas, J. R. Curran, S. Clark, C. J. Bannard, M. Steedman, and B. Webber. *The QED Open-Domain Answer Retrieval System for TREC 2003*. In proceedings of TREC 12, 2003.

3. K. Ahn, B. Alex, J. Bos, T. Dalmas, J. L. Leidner and M. B. Smillie. *Cross-lingual Question Answering with QED*. Workshop of the Cross-Lingual Evaluation Forum (CLEF), 2004.

4. K. Ahn, J. Bos, S. Clark, J. R. Curran, T. Dalmas, J. L. Leidner, M. B. Smillie and B. Webber. *Question Answering with QED and WEE at TREC-2004*. In proceedings of TREC 13, 2004.

C.2 Reading Comprehension: CBC4Kids

1. T. Dalmas, J. L. Leidner, B. Webber, C. Grover and J. Bos. *Generating Annotated Corpora for Reading Comprehension and Question Answering Evaluation.* In proceedings of EACL, Question Answering Workshop, 2003.

2. J. L. Leidner, T. Dalmas, B. Webber, C. Grover and J. Bos. *Automatic Multi-layered Corpora for Evaluating Question Answering methods: CBC4Kids.* In proceedings of EACL, Linguistically Interpreted Corpora Workshop, 2003.

3. T. Dalmas, J. L. Leidner, B. Webber, C. Grover and J. Bos. *Annotating CBC4Kids: A Corpus for Reading Comprehension and Question Answering Evaluation.* Technical Report EDI-INF-RR-0204, School of Informatics, University of Edinburgh.

C.3 Information Fusion and Answer Comparison for QA

1. T. Dalmas and B. Webber. *Generating a Model, Selecting a view, Information Fusion in QA.* In Proceedings of JE ATALA on Question Answering, 2003.

2. T. Dalmas and B. Webber. *Information Fusion for Answering Factoid Questions.* In proceedings of the 2nd CoLogNET-ElsNET Symposium, 2003.

3. T. Dalmas and B. Webber. *Answer Comparison: Analysis of Relationships between Answers to 'Where'-Questions.* In proceedings of the 7th Annual CLUK Research Colloquium, 2004.

4. T. Dalmas and B. Webber. *Using Information Fusion for Open Domain Question Answering.* In proceedings of the KRAQ workshop, IJCAI, 2005.

5. T. Dalmas and B. Webber. *Answer Comparison in Automated Question Answering.* Questions and Answers: Theoretical and Applied Perspectives. A Special Issue of the Journal of Applied Logic. 2006. DOI: 10.1016/j.jal.2005.12.002.

Bibliography

[Abney et al., 2000] Abney, S., Collins, M., and Singhal, A. (2000). Answer Extraction. In *Proceedings of ANLP 2000.*

[Ahn, 2005] Ahn, K. (2005). Topic-Oriented Question Answering. PhD proposal. University of Edinburgh.

[Allison, 1992] Allison, L. (1992). Lazy Dynamic-Programming Can Be Eager. *Information Processing Letters, 43(4)*, pages 207–212.

[Anand et al., 2000] Anand, P., Breck, E., Brown, B., Light, M., Mann, G., Riloff, E., Rooth, M., and Thelen, M. (2000). Fun with Reading Comprehension. Technical report, Johns Hopkins University, Baltimore, MD.

[Baldridge et al., 2001] Baldridge, J., Morton, T., and Bierner, G. (2001). The OpenNLP Maximum Entropy Package.

[Banko and Brill, 2001] Banko, M. and Brill, E. (2001). Scaling to Very very Large Corpora for Natural Language Disambiguation. In *39th Annual Meeting of the Association for Computational Linguistics*, pages 26–33.

[Barzilay, 2003] Barzilay, R. (2003). *Information Fusion for Multidocument Summarization: Paraphrasing and Generation.* PhD thesis, Columbia University.

[Barzilay and McKeown, 2005] Barzilay, R. and McKeown, K. R. (2005). Sentence Fusion for Multidocument News Summarization. In *Computational Linguistics*, page To appear.

[Barzilay et al., 1999] Barzilay, R., McKeown, K. R., and Elhadad, M. (1999). Information Fusion in the Context of Multi-Document Summarization. In *37th Annual Meeting of the Association for Computational Linguistics*, pages 550–557.

[Belkin et al., 1982] Belkin, N. J., Oddy, R. N., and Brooks, H. M. (1982). ASK for Information Retrieval: Part 1. Background and Theory. *Journal of Documentation*, 38(2):61–71.

[Belnap and Steel, 1976] Belnap, N. D. and Steel, T. B. (1976). *The Logic of Questions and Answers.* New Haven and London, Yale University Press.

[Berners-Lee, 1998] Berners-Lee, T. (1998). Semantic Web Roadmap. an Attempt to Give a High-Level Plan of the Architecture of the Semantic WWW. Editing status: Draft. http://www.w3.org/DesignIssues/Semantic.html.

[Brill et al., 2002] Brill, E., Dumais, S., and Banko, M. (2002). Analysis of the AskMSR Question-Answering System. In *Empirical Methods in Natural Language Processing Conference.*

[Brill et al., 2001] Brill, E., Lin, J., Banko, M., Dumais, S., and Ng, A. (2001). Data Intensive Question Answering. In *10th Text Retrieval Conference*, pages 393–400.

[Buchholz and Daelemans, 2001] Buchholz, S. and Daelemans, W. (2001). Complex answers: A case study using a WWW question answering system. *Natural Language Engineering 1 (1).*

[Burger et al., 2002] Burger, J., Cardie, C., Chaudhri, V., Gaizauskas, R., Harabagiu, S., Israel, D., Jacquemin, C., Lin, C., Maiorano, S., Miller, G., Moldovan, D., Ogden, B., Prager, J., Riloff, E., Singhal, A., Shrihari, R., Strzalkowski, T., Voorhees, E., and Weishedel, R. (2002). Issues, Tasks and Program Structures to Roadmap Research in Question and Answering. *NIST.*

[Burke et al., 1997] Burke, R., Hammond, K., and Kulyukin, V. (1997). Question answering from frequently-asked question files: Experiences with the faq finder system. Technical report, Intelligent Information Laboratory, University of Chicago.

[Cardie et al., 2004] Cardie, C., Wiebe, J., Wilson, T., and Litman, D. (2004). Combining Low-Level and Summary Representations of Opinions for Multi-Perspective Question Answering. In Maybury, M., editor, *New Directions in Question Answering*. MIT Press.

[Chaffin, 1992] Chaffin, R. (1992). The Concept of a Semantic Relation. In Lehrer, A. and Kittay, E., editors, *Frames, Fields and Contrasts*, pages 253–288. Lawrence Erlbaum.

[Chaffin et al., 1988] Chaffin, R., Herrmann, D. J., and Winston, M. (1988). An Empirical Taxonomy of Part-whole Relations: Effects of Part-whole Relation Type on Relation Identification. *Language and Cognitive Processes*, 3(1):17–48.

[Chen et al., 1994] Chen, C. H., Basu, K., and Ng, T. (1994). An Algorithmic Approach to Concept Exploration in a Large Knowledge Network. Technical report, MIS Department, University of Arizona, Tucson, AZ.

[Clarke et al., 2001] Clarke, C. L. A., Cormack, G. V., and Lynam, T. R. (2001). Exploiting Redundancy in Question Answering. In *24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 358–365.

[Collins and Loftus, 1975] Collins, A. and Loftus, E. (1975). A Spreading Activation Theory of Semantic Processing. *Psychological Review*, 82:407–428.

[Cugini et al., 2000] Cugini, J., Laskowski, S., and Sebrechts, M. (2000). Design of 3D Visualization of Search Results: Evolution and Evaluation. In *IST/SPIE's 12th Annual International Symposium: Electronic Imaging 2000: Visual Data Exploration and Analysis*.

[Cugini et al., 1996] Cugini, J., Piatko, C., and Laskowski, S. (1996). Interactive 3D Visualization for Document Retrieval. In *Workshop on New Paradigms in Information Visualization and Manipulation , ACM Conference on Information and Knowl edge Management*.

[Cutting et al., 1992] Cutting, D. R., Karger, D. R., Pederson, J. O., and Tukey, J. W. (1992). Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections. In *15th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 318–329.

[Dagan et al., 2005] Dagan, I., Glickman, O., and Magnini, B. (2005). The PASCAL Recognising Textual Entailment Challenge. In *PASCAL Challenges Workshop on Recognising Textual Entailment*.

[Dalmas et al., 2003] Dalmas, T., Leidner, J. L., Webber, B., Grover, C., and Bos, J. (2003). Generating Annotated Corpora for Reading Comprehension and Question Answering Evaluation. In *European Chapter of the Association for Computational Linguistics - Question Answering Workshop*, pages 13–20.

[de Chalendar et al., 2002] de Chalendar, G., Dalmas, T., Elkateb-Gara, F., Ferret, O., Grau, B., Hurault-Plantet, M., Illouz, G., Monceaux, L., Robba, I., and Vilnat, A. (2002). The Question-Answering System QALC at LIMSI: Experiments in Using Web and WordNet. In *11th Text Retrieval Conference*, pages 407–416.

[E. H. Hovy, 2002] E. H. Hovy, U. Hermjakob, D. R. (2002). A question/answer typology with surface text patterns. *DARPA Human Language Technology*.

[Elhadad, 1993] Elhadad, M. (1993). *Using Argumentation to Control Lexical Choice: A Unification-based Implementation*. PhD thesis, Computer Science Department, Columbia University.

[Ely et al., 2002] Ely, J. W., Osheroff, J. A., Ebell, M. H., Chambliss, L., Vinson, D. C., Stevermer, J. J., and Pifer, E. A. (2002). Obstacles to Answering Doctors' Questions about Patient Care with Evidence: a Qualitative Study. *British Medical Journal*, 324.

[Ferro, 2000] Ferro, L. (2000). Reading Comprehension Tests: Guidelines for Question and Answer Writing. Technical report, The MITRE Corporation (unpublished).

[Fleischman et al., 2003] Fleischman, M., Hovy, E., and Echihabi, A. (2003). Offline Strategies for Online Question Answering: Answering Questions Before They Are

Asked. In *41th Annual Meeting of the Association for Computational Linguistics*, pages 1–7.

[Fuhr et al., 2003] Fuhr, N., Goevert, N., Kazai, G., and Lalmas, M. (2003). Overview of the INitiative for the Evaluation of XML. In *First Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*.

[Gansner and North, 1999] Gansner, E. R. and North, S. C. (1999). An Open Graph Visualization System and Its Applications to Software Engineering. *Software – Practice and Experience, 00 (S1)*, pages 1–5.

[Girju, 2001] Girju, R. (2001). Answer Fusion with On-Line Ontology Development. In *North American Chapter of the Association for Computational Linguistics - Student Research Workshop*.

[Graesser et al., 1994] Graesser, A. C., McMahen, C. L., and Johnson, B. K. (1994). Question Asking and Answering. In Gernsbacher, M. A., editor, *Handbook of Psycholinguistics*, pages 517–538. San Diego: Academic Press.

[Green et al., 1961] Green, B., Wolf, A., Chomsky, C., and Laughery, K. (1961). BASEBALL: An Automatic Question Answerer. In *Western Joint Computer Conference*, pages 219–224. Reprinted in Grosz et al. (eds), Readings in Natural Language Processing, pp. 545-550.

[Hachey, 2002] Hachey, B. (2002). Recognising Clauses Using Symbolic and Machine Learning Approaches. Master's thesis. University of Edinburgh.

[Harabagiu and Maiorano, 1999] Harabagiu, S. and Maiorano, S. (1999). Finding Answers in Large Collections of Texts: Paragraph Indexing and Abductive Inference. In *AAAI Fall Symposium on Question Answering Systems*.

[Harabagiu et al., 2001] Harabagiu, S. M., Moldovan, D., and et al., M. P. (2001). Answering Complex, List and Context Questions with LCC's Question-Answering Server. In *10th Text Retrieval Conference*.

[Harman, 1992] Harman, D. (1992). Overview of the First Text REtrieval Conference (TREC 1). In *NIST Special Publication 500 207: The First Text REtrieval Confer ence (TREC)*, pages 1–20.

[Hearst and Pederson, 1996] Hearst, M. A. and Pederson, J. O. (1996). Reexamining the Cluster Hypothesis: Scatter/Gather on Retrieval Results. In *19th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 76–84.

[Hirschman and Gaizauskas, 2001] Hirschman, L. and Gaizauskas, R. (2001). Natural Language Question Answering: The View from Here. *Natural Language Engineering*, 7(4).

[Hirschman et al., 1999] Hirschman, L., Light, M., Breck, E., and Burger, J. (1999). Deep Read: A Reading Comprehension System. In *37th Annual Meeting of the Association for Computational Linguistics*, pages 325–332. College Park MD.

[Honkela, 1997] Honkela, T. (1997). *Self-Organizing Maps in Natural Language Processing*. PhD thesis, Helsinki University of Technology.

[Hovy et al., 2001] Hovy, E., Gerber, L., Hermjakob, U., Junk, M., and Lin, C.-Y. (2001). Question Answering in Webclopedia. In *9th Text Retrieval Conference*.

[Isard et al., 2003] Isard, A., Brockmann, C., Oberlander, J., and White, M. (2003). The Critical Agent Dialogue (CrAg) project. In *Proceedings of the 7th Workshop on the Semantics and Pragmatics of Dialogue (DiaBruck-03)*, pages 185–186, Wallerfangen, Germany. Poster abstract.

[Ittycheriah and Roukos, 2002] Ittycheriah, A. and Roukos, S. (2002). IBM's Statistical Question Answering System - TREC-11. In *11th Text Retrieval Conference*.

[Jijkoun and de Rijke, 2004] Jijkoun, V. and de Rijke, M. (2004). Answer Selection in a Multi-Stream Open Domain Question Answering System. In *Proceedings 26th European Conference on Information Retrieval (ECIR'04)*, pages 99–111.

[Katz et al., 2002] Katz, B., Felshin, S., Yuret, D., Ibrahim, A., Lin, J., Marton, G., McFarland, A. J., and Temelkuran, B. (2002). Omnibase: Uniform Access to Heterogeneous Data for Question Answering. In *7th International Workshop on Applications of Natural Language to Information Systems.*

[Katz et al., 2001] Katz, B., Lin, J., and Felshin, S. (2001). Gathering Knowledge for a Question Answering System from Heterogeneous Information Sources. In *39th Annual Meeting of the Association for Computational Linguistics - Workshop on Human Language Technology and Knowledge Management.*

[Krasner and Pope, 1988] Krasner, G. and Pope, S. (1988). A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80. *Journal of Object-Oriented Programming (JOOP).*

[Lehnert, 1978] Lehnert, W. (1978). *The Process of Question Answering.* Lawrence Erlbaum Publishers, Hillsdale, N.J.

[Leidner and Callison-Burch, 2003] Leidner, J. and Callison-Burch, C. (2003). Evaluating question answering systems using faq answer injection. *Proceedings of the 6th Annual CLUK Research Colloquium.*

[Leidner et al., 2003] Leidner, J. L., Dalmas, T., Webber, B., Grover, C., and Bos, J. (2003). Automatic Multi-layered Corpora for Evaluating Question Answering methods: CBC4Kids. In *European Chapter of the Association for Computational Linguistics - Linguistically Interpreted Corpora Workshop*, pages 39–46.

[LIBGIS, 1981] LIBGIS (1981). Library General Information Survey. U.S. Department of Education, Office of Educational Research and Improvement.

[Light et al., 2001] Light, M., Mann, G., Hirschmann, L., Riloff, E., and Breck, E. (2001). Analyses for Elucidating Current Question Answering technology. *Natural Language Engineering*, 7(4):325–342.

[Lin, 2002] Lin, C.-Y. (2002). The effectiveness of dictionary and web-based answer reranking. *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002).*

[Lin and Pantel, 2001] Lin, D. and Pantel, P. (2001). Discovery of inference rules for question answering. *In Natural Language Engineering 7(4):343-360.*

[Lin and Demner-Fushman, 2005] Lin, J. and Demner-Fushman, D. (2005). Automatically Evaluating Answers to Definition Questions. In *Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005).*

[Lin and Katz, 2003] Lin, J. and Katz, B. (2003). Question Answering from the Web Using Knowledge Annotation and Knowledge Mining Techniques. In *Proceedings of the 12th International Conference on Information and Knowledge Management*, pages 116–123.

[Lin and Katz, 2005] Lin, J. and Katz, B. (2005). Building a Reusable Test Collection for Question Answering. *Journal of the American Society for Information Science and Technology,.*

[Magnini et al., 2002a] Magnini, B., Negri, M., Prevete, R., and Tanev, H. (2002a). Is it the Right Answer? Exploiting Web Redundancy for Answer Validation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 425–432.

[Magnini et al., 2002b] Magnini, B., Negri, M., Prevete, R., and Tanev, H. (2002b). Towards automatic evaluation of question/answering systems. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC-2002)*, Las Palmas, Spain.

[Magnini et al., 2003] Magnini, B., Romagnoli, S., Vallin, A., Herrera, J., Peas, A., Peinado, V., Verdejo, F., and de Rijke, . (2003). The Multiple Language Question Answering Track at CLEF 2003. In *Results of the CLEF 2003 Cross-Language System Evaluation Campaign. Working Notes for the CLEF 2003 Workshop.*

[Mani and Bloedorn, 1997] Mani, I. and Bloedorn, E. (1997). Multi-document Summarization by Graph Search and Matching. In *15th National Conference on Artifical Intelligence (AAAI)*, pages 622–628.

[Mani and Bloedorn, 1999] Mani, I. and Bloedorn, E. (1999). Summarizing Similarities and Differences Among Related Documents. *Information Retrieval*, 1:35–67.

[Marton, 2006] Marton, G. (2006). Nuggeteer: Automatic Nugget-Based Evaluation using Descriptions and Judgements. Technical report, MIT CSAIL Work Product 1721.1/30.

[McKeown and Radev, 1995] McKeown, K. R. and Radev, D. (1995). Generating Summaries of Multiple News Articles. In *18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 74–82.

[Miller et al., 1993] Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., and Miller, K. (1993). Introduction to WordNet: an Online Lexical Database. Technical report, Princeton University.

[Moldovan et al., 2000] Moldovan, D., Girju, R., and Rus, R. (2000). Domain-Specific Knowledge Acquisition from Text. In *Proceedings of the Applied Natural Language Processing Conference*.

[Moldovan et al., 2002] Moldovan, D., Harabagiu, S., Girju, R., Morarescu, P., Lacatasu, F., Novishi, A., Badulescu, A., and Bolohan, O. (2002). LCC Tools for Question Answering. In *11th Text Retrieval Conference*.

[Monz, 2001] Monz, C. (2001). Document Fusion for Comprehensive Event Descriptionl. In *39th Annual Meeting of the Association for Computational Linguistics, Workshop on Human Language Technology and Knowledge Management*.

[Monz and de Rijke, 2001] Monz, C. and de Rijke, M. (2001). Light-Weight Inference for Computational Semantics. In *Inference in Computational Semantics*, pages 59–72.

[MUC, 1992] MUC (1992). Proceedings of the Fourth Message Understanding Conference (MUC-4). DARPA Software and Intelligent Systems Technology Office.

[Nyberg and Frederking, 2003] Nyberg, E. and Frederking, R. (2003). JAVELIN: A Flexible, Planner-Based Architecture for Question Answering. In *Proceedings of HLT-NAACL 2003 Demonstrations*.

[Ogilvie, 2004] Ogilvie, P. (2004). Retrieval Using Structure for Question Answering. In *First Twente Data Management Workshop (TDM) on XML Databases and Information Retrieval*.

[Paice, 1990] Paice, C. (1990). Constructing Literature Abstracts by Computer: Techniques and Prospects. *Information Processing and Management*, 26(1):171–186.

[Pantel and Lin, 2002] Pantel, P. and Lin, D. (2002). Document Clustering with Committes. In *SIGIR 2002*.

[Pantel and Ravichandran, 2004] Pantel, P. and Ravichandran, D. (2004). Automatically Labeling Semantic Classes. In *NAACL 2004*, pages 321–328.

[Pomerantz, 2005] Pomerantz, J. (2005). A Linguistic Analysis of Question Taxonomies. *Journal of the American Society for Information Science and Technology*, 56(7):715–758.

[Porter, 1980] Porter, M. F. (1980). An Algorithm for Suffix Stripping. *Program*, 14(3):130–137.

[Prager et al., 2001] Prager, Chu-Caroll, and Czuba (2001). Use of WordNet Hypernyms for Answering What-Is Questions. In *10th Text Retrieval Conference*. NIST.

[Radev et al., 2000] Radev, D., Jing, H., and Budzikowska, M. (2000). Centroid-based Summarization of Multiple Documents: Clustering, Sentence Extraction, and Evaluation. In *ANLP/NAACL Workshop on Summarization*.

[Radev and McKeown, 1998] Radev, D. and McKeown, K. R. (1998). Generating Natural Language Summaries from Multiple On-line Sources. *Computational Linguistics*, 24(3):469–500.

[Ratnaparkhi, 1996] Ratnaparkhi, A. (1996). A Maximum Entropy Part-of-Speech Tagger. In *Empirical Methods in Natural Language Processing Conference*, pages 133–141.

[Ravichandran and Hovy, 2002] Ravichandran, D. and Hovy, E. (2002). Learning surface text patterns for a question answering system. In *ACL*.

[Richardson, 1995] Richardson, J. V. (1995). *Knowledge-Based Systems for General Reference Work: Applications, Problems, and Progress.* San Diego: Academic Press.

[Riloff and Thelen, 2000] Riloff, E. and Thelen, M. (2000). A Rule-based Question Answering System for Reading Comprehension Tests. In *ANLP/NAACL Workshop on Reading Comprehension Tests as Evaluation For Computer-Based Language Understanding Systems*. Pittsburgh PA.

[Robinson and Rackstraw, 1972] Robinson, W. and Rackstraw, S. J. (1972). *A Question of Answers (Vol. I).* Boston: Routledge and Kegan Paul.

[Rocchio, 1966] Rocchio, J. J. (1966). *Document Retrieval Systems - Optimization and Evaluation.* PhD thesis, Harvard University.

[Saggion et al., 2004] Saggion, H., Gaizauskas, R., Hepple, M., Roberts, I., and Greenwood, M. A. (2004). Exploring the Performance of Boolean Retrieval Strategies For Open Domain Question Answering. In *IR4QA: Information Retrieval for Question Answering*.

[Silverstein et al., 1999] Silverstein, C., Henzinger, M., Marais, H., and Moricz, M. (1999). Analysis of a Very Large Web Search Engine Query Log. *SIGIR Forum*, 33(1):6–12.

[Simmons, 1965] Simmons, R. F. (1965). Answering English Questions by Computer: A Survey. *Communications of the ACM*, 8(1):53–70.

[Snow et al., 2004] Snow, R., Jurafsky, D., and Ng, A. (2004). Learning Syntactic Patterns for Automatic Hypernym Discovery. In *Advances in Neural Information Processing Systems 17*.

[Soubbotin and Soubbotin, 2001] Soubbotin, S. M. and Soubbotin, M. M. (2001). Patterns of Potential Expressions as Clues to Right Answers. In *10th Text Retrieval Conference*.

[Sparck-Jones, 2003] Sparck-Jones, K. (2003). Is Question Answering a Rational Task? In *2nd CoLogNET-ElsNET Symposium. Questions and Answers: Theoretical and Applied Perspectives.*

[Stalker and Murfin, 1996] Stalker, J. C. and Murfin, M. E. (1996). Why Reference Librarians Won't Disappear: A Study of Success in Identifying Answering Sources for Reference Questions. *RQ*, 35(4):489–503.

[Tsur et al., 2004] Tsur, T. O., de Rijke, M., and Sima'an, K. (2004). BioGrapher: Biography Questions as a Restricted Domain Question Answering Task. In *42nd Annual Meeting of the Association for Computational Linguistics*, pages 23–30.

[Voorhees, 2003] Voorhees, E. (2003). Evaluating the Evaluation:A Case Study Using the TREC 2002 Question Answering Track. In *Proceedings of HLT-NAACL 2003*, pages 181–188.

[Voorhees, 2001] Voorhees, E. M. (2001). Overview of the TREC-10 Question Answering Track. In *10th Text Retrieval Conference*, pages 1–15. NIST.

[Voorhees, 2002] Voorhees, E. M. (2002). Overview of the TREC 2002 Question Answering Track. In *11th Text Retrieval Conference*, page 1. NIST.

[Voorhees and Tice, 1999] Voorhees, E. M. and Tice, D. (1999). The TREC-8 Question Answering Track Evaluation. In *8th Text Retrieval Conference*, pages 83–106. NIST.

[Webber et al., 2002] Webber, B., Gardent, C., and Bos, J. (2002). Position statement: Inference in Question Answering. In *LREC Workshop on Question Answering: Strategy and Resources*, pages 19–26.

[Witten and Frank, 2005] Witten, I. H. and Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques.* 2nd Edition, Morgan Kaufmann, San Francisco.

[Woods, 1968] Woods, W. (1968). Procedural Semantics for a Question-Answering Machine. In *AFIPS National Computer Conference*, pages 457–471. AFIPS Press.

[Woods et al., 1972] Woods, W., Kaplan, R., and Nash-Webber, B. (1972). The Lunar Sciences Natural Language Information System: Final Report. Technical Report 2378, Bolt Beranek and Newman, Cambridge MA.

[Yang and Chua, 2002] Yang, H. and Chua, T. (2002). The Integration of Lexical Knowledge and External Resources for Question Answering. In *11th Text Retrieval Conference*. NIST.

[Yang et al., 2003a] Yang, H., Chua, T., and Wang, S. (2003a). Modeling Web Knowledge for Answering Event-based Questions. In *12th International World Wide Web Conference, Poster*.

[Yang et al., 2003b] Yang, H., Cui, H., Kan, M.-Y., Maslennikov, M., Qiu, L., and Chua, T.-S. (2003b). QUALIFIER in TREC-12 QA Main Task. In *12th Text Retrieval Conference*.

[Yangarber, 2000] Yangarber, R. (2000). *Scenario Customization for Information Extraction*. PhD thesis, New York University.

[Zamir and Etzioni, 1999] Zamir, O. and Etzioni, O. (1999). Grouper: A Dynamic Clustering Interface to Web Search Results. In *8th International World Wide Web Conference*.

[Zamir et al., 1997] Zamir, O., Etzioni, O., Madani, O., and Karp., R. M. (1997). Fast and Intuitive Clustering of Web Documents. In *Knowledge Discovery and Data Mining*, pages 287–290.