THE UNIVERSITY
of EDINBURGH

Understanding and Generating Language with Discourse Representation Structures

Jiangming Liu

Doctor of Philosophy

Institute for Language, Cognition and Computation
School of Informatics
University of Edinburgh

2021

Abstract

Natural Language is a way for humans to understand what is happening in the world. However, machines with intelligence prefer symbolic representations that explicitly represent linguistic information of utterances. That is why fundamental natural language processing tasks are necessary. Several symbolic formalisms have been proposed to represent the meaning of natural language. Unlike other symbolic formalisms, Discourse Representation Theory (DRT) is model-theoretic, interpretable and was proposed with the intention to capture more linguistic phenomena such as scope, quantification, and presupposition within and across sentences. Also, the recent development of resources for DRT allows developing tools based on this formalism on a larger scale compared to previous attempts, which were mostly relied on hand engineering. Thesis explores two natural language processing problems relating to Discourse Representation Structure (DRS), namely parsing and generation. We address several questions (1) how to obtain the meaning representation of natural language with arbitrary lengths based on discourse representation theory; (2) how to transform discourse representations in their box-oriented form to computational formats that are easy to model; (3) how to design neural models to automatically generate discourse representation structures from natural language and vice versa; (4) how to adopt annotations of varying quality to improve our models and move to low-resource languages analysis.

We discuss discourse representation theory in the context of related meaning representations and show how DRT deals with various linguistics phenomena, such as predicate-argument structure, word senses, scope and quantification, presupposition, temporal expressions, anaphoric coreference and rhetorical relations. By comparing DRT to related meaning representations, we show why it is important to develop tools based on DRT and why it is better than other formalism.

A computational format is necessary to model discourse representation structures. We provide the definition of Discourse Representation Tree Structures (DRTS) that are derived from discourse representation boxes. We propose a neural DRTS parser with a hierarchical encoder and a 3-step decoder. Furthermore, we improve upon DRTS by introducing a lossless transformation algorithm that allows us to deal with presuppositions and senses. We adopt the Transformer as our DRS parser and compare DRS parsing in tree vs clause format.

In order to explore discourse representation analysis in multiple languages, we propose $\mathbb{U}$niversal Discourse Representation Structure ($\mathbb{U}$DRS) that allows to bridge semantic symbols with the pre-trained language models and to be portable to global

knowledge bases (e.g., GermaNet for German and HowNet for Chinese) instead of only English knowledge bases. It raises the problem of low-resource language analysis. In the monolingual analysis scenario, we propose an iterative learning algorithm that can adopt varying quality annotations. In the cross-lingual analysis scenario, we propose *one-to-many* approach which translates gold standard English to non-English text and trains multiple models (one per language) on the translations, and *many-to-one* approach that translates non-English text to English, and then runs a relatively accurate English model on the translated text methods. These two methods significantly improve DRS parsing in low-resource languages.

We also introduce a general neural framework for DRS-to-text generation that maps DRSs to natural language. Our generator is based on an encoder-decoder architecture equipped with a novel TreeLSTM model. Based on our success with DRS parsing and generation, we empirically study neural interlingual machine translation that first parses the source language to DRSs, and then generates the target language from the DRSs. Although it cannot reach the commercial machine translation systems (e.g., Google Translate) which are trained on billions of data, our neural interlingual machine translation system outperforms the competitive baselines.

Taken together, this thesis explores understanding and generating natural language with discourse representation structures by investigating semantic formalisms (i.e., discourse representation structures and universal discourse representation structures), designing neural models for the monolingual and the cross-lingual semantic parsing and natural language generation, and studying DRS-interlingual machine translation. Our experiments on Groningen Meaning Bank and Parallel Meaning Bank show the successes of neural discourse representation structure parsing and generation and shed light on natural language understanding and generation for natural language processing tasks (e.g., DRS-interlingual machine translation).

Lay summary

In the era of digital information, texts are produced and are available from various sources, such as newspapers, social media, and science. Processing and understanding this information is key to knowing what is happening globally, which also provides decision making support for companies and industries. Computers that are capable of processing text in seconds equipped with high-performance processing units have been developed. However, unlike humans, machines hardly understand natural language. Instead, they prefer computational representations. By addressing this problem, it is necessary to make the cycle of converting natural language to machine-interpretable representation and then back to natural language in a human-friendly format bridges the interaction between human and computers.

One main challenge is that understanding and generating natural language depends on the context across sentences instead of single sentences. For example, when explaining the single sentence *he did it*, we need to know who *he* is and what *it* is that should be interpreted outside of the sentence. To represent the semantic across sentences, we explore Discourse Representation Theory (DRT) as the machine-interpretable representation designed to represent meanings of text within and across sentences.

To understand and generate natural languages, we formalize and study the problem of automatically parsing natural languages to Discourse Representation Structures (DRS) and automatically generating natural language from DRSs within and across sentences for English and low-resource languages. Based on our success with DRS parsing and generation, we study DRS-interlingual machine translation that first understand the source language to DRSs, and then generates the target language from the DRSs.

Acknowledgements

Edinburgh is a beautiful city, and it is my pleasure to study here. I am especially thankful to Shay Cohen and Mirella Lapata. They provided me with constructive suggestions and patient guidance on writing and giving presentations. I would also like to thank Alex Lascarides and Johan Bos for their advice on understanding Discourse Representation Theory.

I would like to thank my thesis examiners Andreas Vlachos and Mark Steedman. They agree to denote time in assessing every piece of my research and give me many comments to improve the thesis.

I want to express my great appreciation to Matt Gardner for mentoring me during the summer internship at AI2. His insightful views open my mind to more related research areas, doing fundamental researches to real-world applications.

I would also offer my gratitude to the Edinburgh NLP group for hosting great talks and inviting amazing speakers. I have had the fortune to attend weekly NLP group meeting to present work and share ideas. Many thanks to the NLP group members: Stefanos Angelidis, Esma Balkir, Rui Cai, Ronald Cardenas, Jianpeng Cheng, Maximin Coavoux, Marco Damonte, Li Dong, Javad Hosseini, Matthieu Labeau, Yang Liu, Chunchuan Lyu, Jonathan Mallinson, Nickil Maveli, Shashi Narayan, Nikos Papasarantopoulos, Ratish Puduppully, Tom Sherborne, John Torr, Yumo Xu, Zheng Zhao, Hao Zheng. I would also like to extend my thanks to my office mates Carol, Javad, Li, Rui, and Yanpeng for creating a pleasure and comfortable environment, where we worked together and chatted for relaxing.

最后，我非常感谢我的父母对我博士的学习给予无条件的支持。另外，我“被提醒”并且真心感谢我女朋友张雅林在博士学习期间的陪伴和帮助。

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.

(*Jiangming Liu*)

Table of Contents

Chapter 1

Introduction

In the era of digital information, texts are produced and are available from various sources, such as newspapers, social media, and science. Processing and understanding this information is key to knowing what is happening in the world, e.g., keeping individuals informed about the COVID-19 pandemic. It also provides decision making support for companies and industries. Our ability to digest all this information falls short – we cannot manually process this large amount of text.[1] This often leads to suboptimal decision making. Recently, computers capable of processing text in seconds equipped with high-performance processing units have been developed. However, unlike humans, machines hardly understand natural language. Instead, they prefer computational representations. In order to abstract natural languages to machine-interpretable symbolic representations, several semantic formalisms have been proposed such as Minimal Recursion Semantics (MRS; Copestake et al. 2005), Abstract Meaning Representation (AMR; Banarescu et al. 2013), and Discourse Representation Theory (DRT; Kamp and Reyle 1993). This thesis explores the potential of DRT for analyzing and generating text; in other words, the cycle of converting natural language to machine-interpretable representations and then back to natural language in a user-friendly format. Several challenges have to be addressed:

- Computational Format. Discourse Representation Theory (DRT; Kamp and Reyle 1993) adopts Discourse Representation Structures (DRS) to represent the meanings of texts. A DRS is displayed as a box that is intuitive and easy to read but not particularly amenable to modelling. It is necessary to convert the box format to a format that allows us to use it as both input and output for statistical natural

[1] 26 billion texts were produced daily online in the United States in 2017, which was reported in https://blog.microfocus.com/how-much-data-is-created-on-the-internet-each-day/

language models.

- Document-level Analysis across Languages. Natural text is a sequence of characters. There is a set of finite predefined characters in each human languages (e.g., English, Arabic, Chinese). Continuous characters make up a word, continuous words make up a sentence, and continuous sentences make up a paragraph/document. Also, natural language understanding takes place in contexts within and across sentences. For example, when explaining the single sentence *He did it*, we do not know who *he* is and what *it* is. The interpretations of *he* and *it* are placed outside of the sentence. The ability to interpret language in context matters for several document-level applications, including discourse machine translation, summarization and dialogue systems. So DRT analysis should handle single sentences and documents within a single language and across languages.

- Consistency and Fluency. DRT abstracts away from unnecessary lexical information (e.g., function words and lexical morphology) that contributes to the fluency of the natural text. For example, the articles (e.g., "a" in English and "le" in French) are excluded from DRT. Also, sentences with the same meaning may have different syntax or surface form or be paraphrases. For example, *According to the theory, we propose a method to solve this problem* and *Based on the theory, a method is proposed by us to solve this problem* have the same meaning but different syntax and words. In order to generate the text according to their meaning representations, we have to ensure that the generated text preserves the original meaning while recovering the lexical information under syntactic structures.

In order to address the challenges in the interaction between natural languages and DRSs, this thesis focuses on semantic parsing that converts natural language to DRSs, and adopts text generation as a way of translating DRSs back to natural language.

1.1 Semantic Parsing

Semantic parsing is the task of mapping natural language to machine-interpretable meaning representations. Semantic parsing can be categorized into shallow and deep semantic parsing. Shallow semantic parsing aims to capture concepts and roles in predefined lexical semantic frames (Gildea and Jurafsky, 2002), such as FrameNet (Baker

et al., 1998), VerbNet (Schuler, 2005), and PropBank (Palmer et al., 2005). Deep semantic parsing aims to produce semantic roles and logical forms of utterances, ensuring all content-bearing units in sentences have corresponding semantics. Grouped by application, semantic parsing can be categorized into purpose-specific and general-purpose semantic parsing. A purpose-specific semantic parser functions as a natural language interface to executable programs. Much previous research focuses on mapping natural queries to executable commands in a database environment (Yu et al., 2018b) such as Prolog (Zelle and Mooney, 1996; Tang and Mooney, 2000), lambda-calculus (Zettlemoyer and Collins, 2005; Kwiatkowksi et al., 2010), SQL (Giordani and Moschitti, 2009; Zhong et al., 2017; Yu et al., 2018b) and SPARQL (Reddy et al., 2014; Yih et al., 2015; Su et al., 2016), or code written in various programming languages (Ling et al., 2016; Yin and Neubig, 2017; Rabinovich et al., 2017b).

General-purpose semantic parsing outputs symbolic meaning representations of texts without grounding them to a specific database where outputs are executable. The symbolic meaning representation by a general-purpose semantic parser is linguistically-motivated and unveils semantic information such as predicates and their arguments, sense differentiation, scope, presupposition, and anaphoric coreference. By considering this semantic information, various symbolic meaning representations are defined together with available annotations, such as DELPH-IN Minimal Recursion Semantics (DM; Ivanova et al. 2012a), Prague Semantic Dependencies (PSD; Hajič et al. 2012; Miyao et al. 2014a), Elementary Dependency Structure (EDS; Oepen and Lønning 2006a), Universal Conceptual Cognitive Annotation (UCCA; Abend and Rappoport 2013a), Universal Decompositional Semantics (UDS; White et al. 2016), Abstract Meaning Representation (AMR; Banarescu et al. 2013), and Discourse Representation Theory (DRT; Kamp and Reyle 1993; Bos et al. 2017; Abzianidze et al. 2017). This thesis focuses on general-purpose deep semantic parsing, taking discourse representation theory as our symbolic semantic formalism.

1.2 Text Generation

Text generation is the task of generating sequences of words that can be read and understood by humans. Most text generation tasks produce texts by *conditioning on* some input. According to the input signals, text generation can be categorized as control-free, controllable, and input-conditional. Control-free text generation aims to produce sequences of words around several keywords (or topics) without any constraints on

the content, such as poem generation (Colton et al., 2012; Zhang and Lapata, 2014). Controllable text generation produces texts given attribute labels (Hu et al., 2017), e.g., sentiment, tense, and style. Input-conditional text generation transforms structured or non-structured data to natural language, including image captioning (Karpathy and Fei-Fei, 2015), data-to-text generation such as SQL-to-text generation (Iyer et al., 2016), AMR-to-text generation (Konstas et al., 2017), table-to-text generation (Wang et al., 2018), text summarization and machine translation. This thesis works on input-conditional data-to-text generation, specifically generating natural language according to the symbolic semantic representation given as input.

1.3 Deep Learning

Deep learning is a class of machine learning algorithms that use multiple-layer artificial neural networks to extract high-level features (Deng and Yu, 2014). The success of applying neural networks in computer vision (Krizhevsky et al., 2012) renewed interest in applying them to natural language processing (Collobert et al., 2011). Recently, various neural models achieved significant improvements on most natural language processing task, such as named entity recognition (Huang et al., 2015; Li et al., 2020), syntactic parsing (Chen and Manning, 2014; Dyer et al., 2016; Dozat and Manning, 2017; Kitaev and Klein, 2018), semantic parsing (Dong and Lapata, 2016; Damonte et al., 2017; Cai and Lam, 2020a), summarization (See et al., 2017) and machine translation (Bahdanau et al., 2015; Vaswani et al., 2017). Different from conventional methods that depend on manually-designed features, neural methods automatically extract features that are represented as tensors. From how to design features, research focus has moved to how to design neural network architectures that can capture relevant and useful information for corresponding tasks. This thesis aims to design novel neural models for DRT parsing and generation.

1.4 Research Questions

There is a collection of research questions related to DRT parsing and generation that this thesis aims to answer.

Why Discourse Representation Theory The meaning of the sentences involves various linguistic phenomena such as predicate-argument structures, sense differentia-

tion, coreference. Many symbolic meaning representations that are designed to this end. For example, DELPH-IN MRS-Derived Bi-Lexical Dependency (DM; Ivanova et al. 2012b) and Prague Semantic Dependencies (PSD; Hajič et al. 2012; Miyao et al. 2014b) capture predicate-argument relations by constructing semantic dependencies between words, Elementary Dependency Structures (EDS; Oepen and Lønning 2006b) abstract away from word surface form to graphs where graph nodes correspond to logical predicates and edges to labelled argument positions. Abstract Meaning Representation (AMR; Banarescu et al. 2013) further abstracts graph nodes to make lexical decomposition and adopts reentrancies to capture anaphoric coreference. For these meaning representations, we find that 1) they only focus on one or several linguistic phenomena; 2) they are designed for single sentences without considering linguistic phenomena across sentences, which play a crucial role in document-level applications; 3) Most of them are biased forward English sentences, although there exist for some non-English annotation resources. We try to answer the question if DRT can deal with more linguistic phenomena within and across the sentences in multiple languages.

Discourse Representation Structures Modeling The basic meaning-carrying units are discourse representation structures (DRSs). Conventional DRSs are depicted as two-layer boxes where the top layer contains variables, and the bottom layer contains conditions to describe the variables. It raises the problem that boxes are easy to read but unsuitable for modelling. When designing a DRT parser or generator, we have to transform the boxes to a computational format, the specific characteristics of which will be explored in this thesis.

Resources for Discourse Representation Theory Existing semantic parsers are, for the most part, data-driven using annotated examples consisting of utterances and their meaning representations (Zelle and Mooney, 1996; Wong and Mooney, 2006; Zettlemoyer and Collins, 2005). Neural models, achieving state-of-the-art performance on various natural language processing tasks, are data-hungry. However, the high-quality annotations of meaning representations are costly to obtain (annotators ideally should have a corresponding linguistics background). A tradeoff lies in constructing various-quality annotations (e.g., silver, bronze data). Some data are directly annotated by experts, and the others are automatically annotated by fully-trained (e.g., statistical or rule-based) parsers or human-in-the-loop methods. We investigate how to adopt various-quality annotations in designing our models.

Neural Structural Models The successful application of encoder-decoder models (Sutskever et al., 2014; Bahdanau et al., 2015) to a variety of NLP tasks has provided strong impetus to treat semantic parsing as a sequence transduction problem where an utterance is mapped to a target meaning representation in string format (Dong and Lapata, 2016; Jia and Liang, 2016; Kočiský et al., 2016; Foland and Martin, 2017); likewise, data-to-text generation is viewed as a sequence-to-sequence problem where a structural data is linearized to a sequence of tokens and then a sequence of words are autoregressively generated (Konstas et al., 2017). The fact that meaning representations do not naturally conform to a linear ordering has also prompted efforts to develop recurrent neural networks and transformers tailored to tree or graph-structured *decoders* (Dong and Lapata, 2016; Cheng et al., 2017; Yin and Neubig, 2017; Alvarez-Melis and Jaakkola, 2017; Rabinovich et al., 2017a; Buys and Blunsom, 2017; Damonte et al., 2017; Ballesteros and Al-Onaizan, 2017; Guo and Lu, 2018; Liu et al., 2018b; Zhang et al., 2019; Cai and Lam, 2020a), and *encoders* (Tai et al., 2015; Defferrard et al., 2016; Zhang et al., 2016; Li et al., 2016b; Alvarez-Melis and Jaakkola, 2017; Seo et al., 2018; Veličković et al., 2018; Wang et al., 2019; Yun et al., 2019). However, all these structural networks focus on sentence-level tasks, in this thesis we focus on discourse representation structures for getting not only sentences but also documents, which are more complicated structures with longer sequences of tokens. We investigate the existing models and propose novel models to this end.

Interlingual Machine Translation Machine translation methods can be categorized into direct, transfer, and interlingual translation (Hutchins and Somers, 1992; Dorr et al., 2004).[2] Nearly all current neural machine translation systems (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015; Vaswani et al., 2017) belong to the family of direct translation, which generates sequences of words in the target language by directly conditioning on the sequence of tokens in the source language. Prior to the advent of neural methods, machine translation was based on bilingual rules (lexical or syntactic), which were manually designed or automatically extracted from a bilingual corpus (statistical MT; Koehn 2009). This is an example of transfer-based systems, where transfer rules are applied to one pair of languages. However, interlingual machine translation has two steps.The source language is parsed to its interlingual representations at the first step, and then the target language is generated

[2]Although initially, the terms direct, transfer, and interlingual translation were reserved for rule-based machine translation, we argue that these labels also apply to statistical and neural machine translation according to the types of rules that are obtained by humans or statistics.

from the interlingual representations at the second step (Richens, 1958; Hutchins and Somers, 1992). The interlingual translation enable linguistic and translation problems more clearly and usefully formulated in terms of a standard language (Richens, 1958); when a new language appears, being able to parse the language to the interlingua is sufficient; it only requires a monolingual analyzer for the specific language and interlingual formalism. With these benefits, the question is how to define a representation formalism that can be used in interlingual machine translation. Given that DRT can capture a wealth of linguistic phenomena and represent text meaning, we propose to use DRSs as a bridge between languages for interlingual machine translation.

1.5 Thesis Overview

This thesis investigates Discourse Representation Theory and aims to develop neural models that map natural language to their Discourse Representation Structures (parsing) and generate natural language from their Discourse Representation Structures (generation). We also investigate whether discourse representation theory can be used as an interlingual formalism for machine translation.

Figure 1.1 provides an overview of the thesis. By comparison to several popular meaning representations, we discuss discourse representation theory (Kamp and Reyle, 1993), showing its ability to deal with various linguistic phenomena, including scope, quantification, anaphoric coreference, the interpretation of pronouns and temporal expressions within and across sentences. With Neo-Davidsonian event semantics (Davidson, 1967; Parsons, 1990), DRT takes unary predicates with event arguments only and introduces other arguments through thematic roles into event semantics. DRT is extended by introducing rhetorical relations between discourse segments (Asher and Lascarides, 2003), enriching the meaning representations. Based on these extensions, we introduce two datasets that can be used to investigate automatically discourse representation structure parsing as benchmarks. The first dataset is the Groningen Meaning Bank (GMB, Bos et al. 2017), which is designed for English text only, and the annotations are automatically obtained via BOXER (Bos, 2015) and then manually corrected. The other dataset is the Parallel Meaning Banking (PMB, Abzianidze et al. 2017), which focuses on building DRT annotations for multiple languages, including English, German, Italian, and Dutch.

Based on these available datasets, we develop a neural DRT parser to automatically obtain Discourse Representation Structures (DRSs), the basic meaning-carry units in

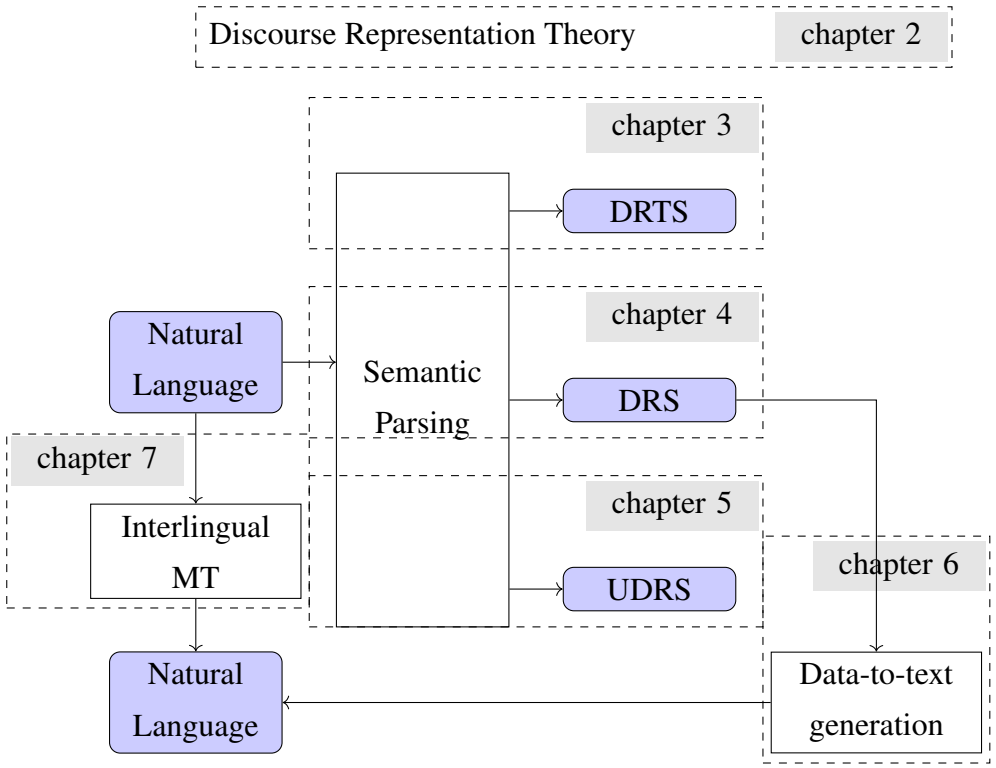


Figure 1.1: Overview of the thesis. We focus on three natural language processing tasks (white boxes), i.e., semantic parsing, data-to-text generation and interlingual machine translation, which are based on discourse representation theory (Chapter 2). In semantic parsing, we introduce the works of mapping natural languages to Discourse Representation Tree Structures (DRTS; Chapter 3), full Discourse Representation Structures (DRS; Chapter 4), and Universal Discourse Representation Structures (UDRS; Chapter 5). In data-to-text generation, we introduce the work of translating full Discourse Representation Structures to natural language (Chapter 6). Based on the semantic parser and DRS-to-text generator, we explore the interlingual machine translation taking DRS as interlingual formalism (Chapter 7).

DRT. The original DRSs are depicted as boxes that are easy to read but not friendly to model. We define Discourse Representation Tree Structures (DRTSs), which are simplified and derived from the DRS boxes, and design a neural parser that can produce the DRTS for texts in arbitrary length. The parser is based on the encoder-decoder framework and enhanced with several components: a hierarchical encoder represents the inter- and intra- sentential information, a 3-step decoder produces large DRTSs at different levels of granularity, and state trackers ensuring well-formed DRTS output. Also, in order to work on the full DRS parsing without any simplification, we compare

the two computational DRS formats (i.e., clauses and trees) and adopt the Transformer (Vaswani et al., 2017) as our parser. With iterative training on various-quality datasets, the final parser achieves state-of-the-art performance.

Recent DRS parsers are designed to obtain only English DRSs. One reason is that non-English languages do not have enough annotated training data required by the data-hungry neural parsers. We consider the task of cross-lingual semantic parsing in the style of Discourse Representation Theory (DRT), where knowledge from annotated corpora in a resource-rich language is transferred via bitext to guide learning in other languages. We introduce 𝕌niversal Discourse Representation Theory (𝕌DRT), a variant of DRT which explicitly anchors semantic representations to tokens in the linguistic input. We develop a semantic parsing framework based on the Transformer architecture and employ it to obtain semantic resources in multiple languages following two learning schemes. The **Many-to-One** approach translates non-English texts to English and then runs a relatively accurate English parser on the translated text, while the **One-to-Many** approach translates gold standard English to non-English text and trains multiple parsers (one per language) on the translations. Experimental results on the Parallel Meaning Bank (PMB, Abzianidze et al. 2017) show that our proposal outperforms strong baselines by a wide margin and can be used to construct (silver-standard) meaning banks for 99 languages.

Text generation from meaning representations is not restricted to single sentences. DRSs are *document-level* representations which encode rich semantic detail pertaining to rhetorical relations, presupposition, and co-reference *within* and *across* sentences. We formalize the task of neural DRS-to-text generation and provide modelling solutions for condition ordering and variable naming that render generation from DRSs non-trivial. Our generator relies on a novel sibling treeLSTM model which can accurately represent DRS structures and is more generally suited to trees with wide branches.

We take machine translation to bring together the parsing and generation models developed in this thesis. A cross-lingual DRS parser yields DRSs for the source language, and given a DRS parse, and we adopt the DRS-to-text generator to produce output in the target language, aiming to shed the lights on the interlingual machine translation involving linguistics and semantics.

The main contributions of this thesis are:

- We formally define Discourse Representation Tree Structures for sentences and documents and present a general framework for parsing discourse structures of

arbitrary length and granularity.

- We propose a box-to-tree conversion algorithm that is lossless and reversible, and then we compare two computational DRS formats in DRS parsing
- We propose 𝕌DRS to explicitly anchor DRSs to lexical tokens, which we argue is advantageous for monolingual and cross-lingual parsing.
- We propose a general cross-lingual semantic parsing framework based on the Transformer architecture following the one-to-many and many-to-one learning paradigms; we show the approach's scalability by creating a large corpus with (silver-standard) discourse representation annotations in 99 languages.
- We propose a general DRS-to-text generation framework based on neural models, where a novel sibling tree LSTM model is proposed to be suitable to model wide tree structures.
- We carry an empirical study of interlingual machine translation based on cross-lingual DRS parsing and DRS-to-text generation.

1.6 Outline

The rest of the thesis is organized as follows:

- Chapter 2 introduces Discourse Representation Theory (DRT) together with related meaning representations. We show how DRT deals with various linguistic phenomena and introduce two DRS datasets, namely the Groningen Meaning Bank (GMB; Bos et al. 2017) and the Parallel Meaning Bank (PMB; Abzianidze et al. 2017).
- Chapter 3 provides the definition of Discourse Representation Tree Structures that are simplified and derived from DRS boxes. We propose a neural DRTS parser to parse multiple sentences (documents) into their DRTSs, using the encoder-decoder framework with a hierarchical encoder and 3-step decoding. We also propose state trackers to ensure well-formed outputs. Experimental results demonstrate that our proposed DRTS parser outperforms competitive baselines.
- Chapter 4 improves upon Discourse Representation Tree Structure by introducing a lossless conversion algorithm from DRS boxes to DRS trees. The new

conversion allows representing presupposition and word senses which are necessary for the interpretation of DRSs. Based on this conversion, we adopt the Transformer as our DRS parser and compare DRS parsing in tree vs clause format. Also, we propose an iterative training method to enhance the training procedure using data varying in annotation quality. Experimental results show that our proposed model outperforms several competitive baselines.

- Chapter 5 presents 𝕌niversal Discourse Representation Structure (𝕌DRS) that anchors predicates and constants in the meaning representation language to input tokens, thereby allowing to bridge semantic symbols with the pre-trained models. We also propose two cross-lingual DRS parsing methods, one-to-many and many-to-one, which significantly improve DRS parsing in low-resource languages.

- Chapter 6 introduces a general neural framework for DRS-to-text generation, which takes DRSs as input and outputs a sequence of words considering semantic representations within and across sentences. We also propose sibling treeLSTM, a new structural encoder advantageous to model wide but flat trees. Experimental results show that the model with the proposed sibling tree LSTM outperforms competitive baselines.

- In Chapter 7, we show the potential of our cross-lingual parser (Chapter 5) and generator (Chapter 6) for interlingual machine translation. Experimental results show that the proposed DRS-interlingual MT system outperforms a direct neural machine translation system in the same settings but still underperforms commercial MT systems (i.e., Google Translate).

- Chapter 8 concludes the thesis and discusses future work.

Chapter 2

Discourse Representation Theory

In this chapter, we review various popular meaning formalism that has been proposed in the literature for general-purpose semantic parsing. By considering to deal with the various linguistic phenomena, we first show the extension of Discourse Representation Theory (DRT; Kamp and Reyle 1993) with the formal construction of Discourse Representation Structures that are the basic meaning-carry unit in DRT, and then we show the properties of the extended DRT and how it deals with several linguistic phenomena. Compared to the other meaning formalisms in mind such as DELPH-IN Minimal Recursion Semantics (DM; Ivanova et al. 2012b), Prague Semantic Dependencies (PSD; Hajič et al. 2012; Miyao et al. 2014b), Elementary Dependency Structure (EDS; Oepen and Lønning 2006a), Universal Conceptual Cognitive Annotation (UCCA; Abend and Rappoport 2013a) and Abstract Meaning Representation (AMR; Banarescu et al. 2013), DRT with more linguistic concerns has not been explored further in modelling purpose due to the lack of available annotated corpus. Recently, some DRS resources are released to the public, which motivates DRS modelling in the era of deep learning. The descriptions of two available annotated corpus are given at the end of the chapter.

2.1 Discourse Representation Structure

DRT uses discourse representation structures (DRSs) to represent a hearer's mental representation of discourse as it unfolds over time. A DRS has two components: a set of discourse references representing entities and events under discourse and a set of conditions representing information related to the referents. By enriching the DRT with more linguistic concerns, the conventional DRT has been extended by using a neo-

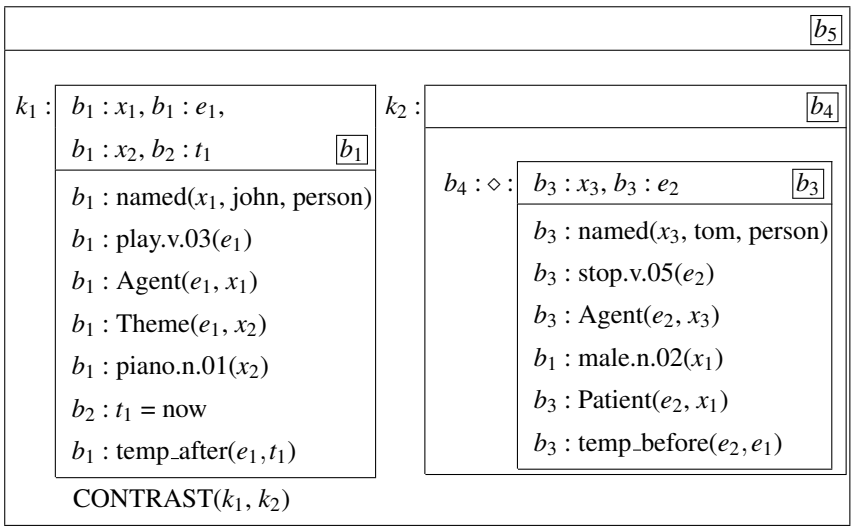


Figure 2.1: The DRS in corpora for text *John is going to play the piano. Tom might stop him.*

Davidsonian analysis of events (Kipper et al., 2008), i.e., events are first-order entities characterized by one-place predicate symbols (e.g., john(e_1) in Figure 2.1), and using the thematic roles of VerbNet (Schuler, 2005) as relations between individual entities, and the synsets of WordNet (Fellbaum, 1998) to denote individual concepts. In addition, it addresses the presupposition following Projective Discourse Representation Theory (PDRT; Venhuizen et al. 2013), an extension of DRT specifically developed to account for the interpretation of presuppositions. In PDRT, each basic DRS introduces a label, which can be bound by a pointer indicating the interpretation site of semantic content. To account for the rhetorical structure of texts, DRS in corpora adopts Segmented Discourse Representation Theory (SDRT; Asher and Lascarides 2003). In SDRT, discourse segments are linked with rhetorical relations reflecting different characteristics of textual coherence, such as temporal order and communicative intentions (see CONTRAST(k_1, k_2) in Figure 2.1).

More formally, Bos et al. (2017) gives the constructions of DRS in Groningen Meaning Bank (GMB) and Parallel Meaning Bank (PMB), which are expressions of type $\langle exp_e \rangle$ (denoting individuals or discourse referents) and $\langle exp_t \rangle$ (i.e., truth values):

$$\langle exp_e \rangle ::= \langle ref \rangle, \quad \langle exp_t \rangle ::= \langle drs \rangle | \langle sdrs \rangle, \tag{2.1}$$

discourse referents $\langle ref \rangle$ are in turn classified into six categories, namely, common referents (x_n), event referents (e_n), state referents (s_n), segment referents (k_n), proposition referents (p_n), and time referents (t_n). $\langle drs \rangle$ and $\langle sdrs \rangle$ denote basic and segmented

DRSs, respectively:

$$\langle drs \rangle ::= \langle pvar \rangle : \boxed{\frac{(\langle pvar \rangle, \langle ref \rangle)^*}{(\langle pvar \rangle, \langle condition \rangle)^*}}, \tag{2.2}$$

$$\langle sdrs \rangle ::= \boxed{\frac{k_1 : \langle exp_t \rangle, k_2 : \langle exp_t \rangle}{coo(k_1, k_2)}} \mid \boxed{\frac{\substack{k_1:\langle exp_t \rangle \\ k_2:\langle exp_t \rangle}}{sub(k_1, k_2)}}, \tag{2.3}$$

Basic DRSs consist of a set of referents ($\langle ref \rangle$) and conditions ($\langle condition \rangle$), whereas segmented DRSs are *recursive* structures that combine two $\langle exp_t \rangle$ by means of coordinating (*coo*) or subordinating (*sub*) relations. DRS conditions can be basic or complex:

$$\langle condition \rangle ::= \langle basic \rangle | \langle complex \rangle, \tag{2.4}$$

Basic conditions express properties of discourse referents or relations between them:

$$\begin{aligned} \langle basic \rangle ::= & \langle sym_1 \rangle (\langle exp_e \rangle) \mid \langle sym_2 \rangle (\langle exp_e \rangle, \langle exp_e \rangle) \\ & \mid \langle exp_e \rangle = \langle exp_e \rangle \mid \langle exp_e \rangle = \langle num \rangle \\ & \mid timex(\langle exp_e \rangle, \langle sym_0 \rangle) \\ & \mid named(\langle exp_e \rangle, \langle sym_0 \rangle, class). \end{aligned} \tag{2.5}$$

where $\langle sym_n \rangle$ denotes n-place predicates, $\langle num \rangle$ denotes cardinal numbers, *timex* expresses temporal information (e.g., timex$(x_7, 2005)$ denotes the year 2005), and *class* refers to named entity classes (e.g., location).

Complex conditions are unary or binary. Unary conditions have one DRS as argument and represent negation ($\neg$), modal operators expressing necessity ($\Box$), and possibility ($\Diamond$). Condition $\langle ref \rangle : \langle exp_t \rangle$ represents verbs with propositional content (e.g., factive verbs). Binary conditions represent conditional statements ($\rightarrow$) and disjunctions ($\vee$).

$$\begin{aligned} \langle complex \rangle ::= & \langle unary \rangle \mid \langle binary \rangle, \\ \langle unary \rangle ::= & \neg \langle exp_t \rangle \mid \Box \langle exp_t \rangle | \Diamond \langle exp_t \rangle | \langle ref \rangle : \langle exp_t \rangle \\ \langle binary \rangle ::= & \langle exp_t \rangle \rightarrow \langle exp_t \rangle | \langle exp_t \rangle \vee \langle exp_t \rangle | \langle exp_t \rangle ? \langle exp_t \rangle \end{aligned} \tag{2.6}$$

2.2 Linguistic Phenomena

How to deal with various linguistic phenomena is the goal of designing semantic formalism. We discuss how the extended DRT uses DRSs to deal with linguistic properties and phenomena, such as predicate-argument, sense differentiation, scope, presupposition, temporal expression, coreference and rhetorical relation.

Predicate-Argument An argument is an expression that helps complete the meaning of a predicate, and the predicate is for the most part evoked by a verb as an action (or event) or a noun as an entity. DRT adopts Neo-Davidsonian event semantics (Davidson, 1967; Parsons, 1990), i.e., unary predicates with event arguments e, and introduces other arguments through thematic roles. Different from the syntax, thematic roles of the semantic arguments of the given predicate remain consistent as the form of that predicate changes. For example, in Figure 2.1, $\texttt{play.v.03}(e_1)$ is the unary predicate of the event evoked by the word *play* together with their `Agent` and `Theme`, where $\texttt{Agent}(e_1,x_1)$ means the entity *john* (x_1) performs the action, and $\texttt{Theme}(e_1,x_2)$ means the entity $piano(x_2)$ undergoes the action.

Sense Differentiation Sense differentiation is key to the meaning representation by identifying the meaning of the words. The sentence phrased with the same words could have different interpretations. For example, *we lost our capital.* could be interpreted as like we lost our money, or could be interpreted as like we lost the capital city of our country. Word senses in DRT are expressed as WordNet synset identifiers (Fellbaum, 1998) to differentiate each predicate's meaning. For example, the predicate `play.v.03` is interpreted as a verb with the 3rd sense showing *play on an instrument*.

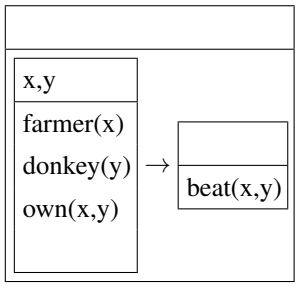
Scope and Quantification Scope such as negation and modality is the range where the semantic should be identified to show *negation*, *possibility*, and *necessity*. DRT expresses the scope in nested boxes where the referents and conditions are scoped in the boxes with or without scope-bearing operators, e.g., $\diamond$ means possibility in Figure 2.1. Similarly, the scopes of quantifiers in logic is the range in the formula where the quantifiers engages in. For example, in sentence *if a farmer owns a donkey, he beats it*, the *he* does not refer to a singular farmer, but to every farmer that owns a donkey. The first-order logic is

$$\forall x \forall y((farmer(x) \wedge donkey(y) \wedge owns(x,y)) \rightarrow beat(x,y)).$$

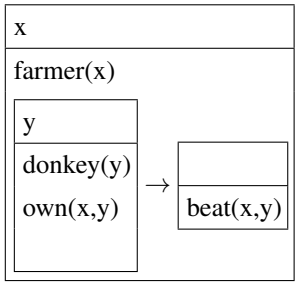
However, in sentence *if the farmer owns a donkey, he beats it*, the *he* refers to a specific farmer. The first-order logic is

$$\exists x((farmer(x) \wedge \forall y((donkey(y) \wedge owns(x,y)) \rightarrow beat(x,y))).$$

DRT depicts the scopes of quantification for *if a farmer owns a donkey, he beats it* as:



and *if the farmer owns a donkey, he beats it* as:



Presupposition A presupposition is an implicit assumption about the world or background belief relating to an utterance whose truth is taken for granted in discourse. [1] For example, the two sentences *a man bought the plant* and *the man bought the plant* have different meanings. The word *man* in the first sentence represents a general male person without semantic scopes, while the word *man* with the article *the* in the second sentence is specific to the man who is described somewhere (e.g., discourses and dialogues). DRT is extended by importing the projection pointers, taking into account the interpretation of presupposition. Each referent and condition is assigned with a pointer to a box. The pointer of asserted content will be bound by its local context, which is necessary to interpret the local context. For example, *the man* is depicted as $\begin{array}{|l|}\hline b_2 : x_1 \quad \boxed{b_1} \\ \hline b_2 : \texttt{male.n.02}(x_1) \\ \hline \end{array}$, where `male.n.02` is bounded to outside box b_2 not the current box b_1, while *a man* is depicted as $\begin{array}{|l|}\hline b_1 : x_1 \quad \boxed{b_1} \\ \hline b_1 : \texttt{male.n.02}(x_1) \\ \hline \end{array}$, where `male.n.02` is bounded to current box b_1.

Temporal Expressions Tense is a category that expresses time reference to the moment of speaking (Brown, 2005), and tenses are usually manifested by the use of specific forms of verbs, showing that an action happened in past, is happening or will

[1] The definition is from Wikipedia

happen in future. Tenses show the temporal information in semantics. DRT explicitly describes tenses by comparing the event unary predicate to the absolute time point `now`, and also it describes event order with a set of predefined relations. For example, in Figure 2.1, the condition `temp_after`(e_1, t_1) describes the action `play.v.03`(e_1) occurred in the future (after now (t_1)), and the condition `temp_before`(e_2, e_1) means that the action `stop.v.05`(e_2) which occurred before the action *play*.

Coreference Coreference happen when two or more expressions in a text refer to the same entity. For example, in text *John is going to play the piano. Tom might stop him*, the proper noun *John* (antecedent) and the pronoun *him* (anaphor) refer to the same person, namely to *John*. DRT is a variable-rich meanining representation, the coreference is resolved by using the same variable to describe the same entities or using a *equation* condition to bridge coreferential variables. In the above example, the predicate `male.n.02`(x_1) (*him*) refers to the person `named`$(x_1$, john, person) (*John*), which is shown in Figure 2.1, or DRT describes *John* and *him* with `named`$(x_1$, john, person) and `male.n.02`(x_4), respectively, with an extra condition $x_1 = x_4$.[2]

Rhetorical Relation A rhetorical relation (discourse relation) describes how discourse segments are logically connected (Asher and Lascarides, 2003). Jasinskaja and Karagjosova (2015) summarizes the rhetorical relations as *Elaboration* that holds between two discourse units where the second describes the same state of affairs as the first one (in different words), *Explanation* that gives the cause or reason why the state of affairs presented, *Parallel* that holds between two or more discourse units in virtue of the similarity or uniformity of their content along some relevant dimension, *Contrast* that connects discourse units whose content is opposite or contradictory, *Narration* holds events taking place one after the other and the order of events matching the textual order of utterances, and *Result* holds two events where the second event does not only follow the first in time but is also caused by it. DRT is extended by linking discourse segments to each other via binary relations, reflecting textual coherence such as temporal order and communicative intentions. For example, the two segments *John is going to play the piano Tom might stop him* are opposite in terms of playing the piano. DRT marks the two segments with names k_1 and k_2, respectively, and then

[2]However, in my mind, they did not resolve the complicate coreferences, such as split antecedents. For example, in text *John would leave and Tom would stay. They are sad*, the anaphor *they* has a split antecedent, referring to both John and Tom. One possible solution is defining an operator ($\subset$) to make *John* $\subset$ *They* and *Tom* $\subset$ *They*.

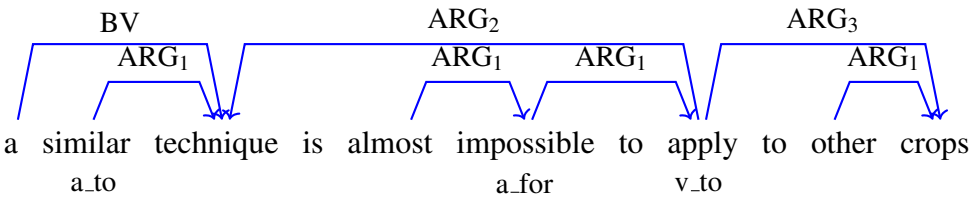


Figure 2.2: The DM representation for the sentence *A similar technique is almost impossible to apply to other crops*, where ARG shorts for argument, and a_to, a_for and v_to are the identified senses for *similar*, *impossible* and *apply*, respectively.

use a rhetorical relation CONTRAST(k_1, k_2) between them, which is shown in Figure 2.1.

2.3 Semantic Formalism

There are some other semantic formalisms that are used for general-purpose semantic parsing. By comparing these semantic representations to discourse representation theory, we show that DRT captures more semantics of texts, which motivate the design of accurate DRT parsers automatically outputting the DRSs of the given text. Next, we introduce these semantic representations in terms of their abilities to deal with linguistic phenomena discussed above.[3]

DELPH-IN MRS-Derived Bi-Lexical Dependency (DM; Ivanova et al. 2012b) DM is simplified from underspecified logical forms, English Resource Semantics (ERS, Flickinger 2000) that is rooted in general theories of grammar, Head-Driven Phrase Structure Grammar (HPSG, Pollard and Sag 1994). The simplifications of DM recast predicate-argument structure in the form of bi-lexical dependency graphs, where edge labels show types of semantic arguments. An example is shown in the top of Figure 2.2. DM is capable to deal with coarse-grained sense differentiation, e.g., the verb *apply* has the sense *apply_to* instead of the sense *apply_for* in Figure 2.2.

Prague Semantic Dependencies (PSD; Hajič et al. 2012; Miyao et al. 2014b) PSD is simplified from the multi-layer syntactico-semantic annotations rooted in the general linguistic framework of Functional Generative Description (FGD, Sgall et al. 1986). Similar to DM, PSD essentially recast core predicate–argument structure in the form

[3] https://github.com/cfmrp/tutorial

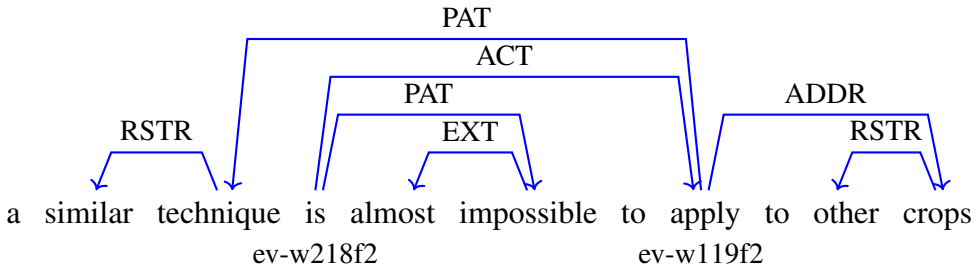


Figure 2.3: The PSD representation for the sentence *A similar technique is almost impossible to apply to other crops*, where RSTR shorts for restriction, PAT for patient, ACT for actor, EXT for extend, and ADDR for addressee, and ev-w218f2 and ev-w119f2 are the identified senses for verbs *is* and *apply*, respectively.

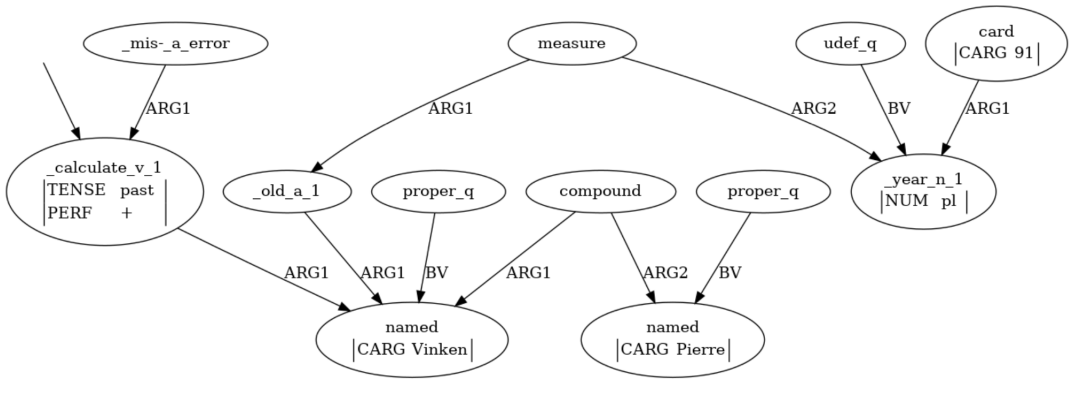


Figure 2.4: The EDS representation for the sentence *Pierre Vinken, 91 years old, had mis-calculated.*, where each node is a predicate and each direct edges (A $\xrightarrow{label}$ B) means B is a *label* of A. ARG stands for argument and BV for.

of pure bi-lexical dependency graphs. Different from DM, PSD treats determiners as non-content-bearing units and adjuncts as dependencies, e.g., restriction (RSTR) and extend (EXT) in Figure 2.3.

Elementary Dependency Structures (EDS; Oepen and Lønning 2006a) Unlike DM and PSD organized with bi-lexical semantic dependencies, EDS faces the challenge of lexical decomposition. It encodes English Resource Structure (ERS; Flickinger 2000) in a semantic dependency graph, where each node corresponds to predicates and edges to arguments but discards the partial information on the semantic scope from the full ERS. For example, in Figure 2.4, the predicate *_calculate_v_1* has a argument *Vinken* and it is also a argument of the predicate *_mis-_a_error*. The meaning of word *mis-calculated* is decomposed to two predicate nodes *_calculate_v_1* and *_mis-_a_error*

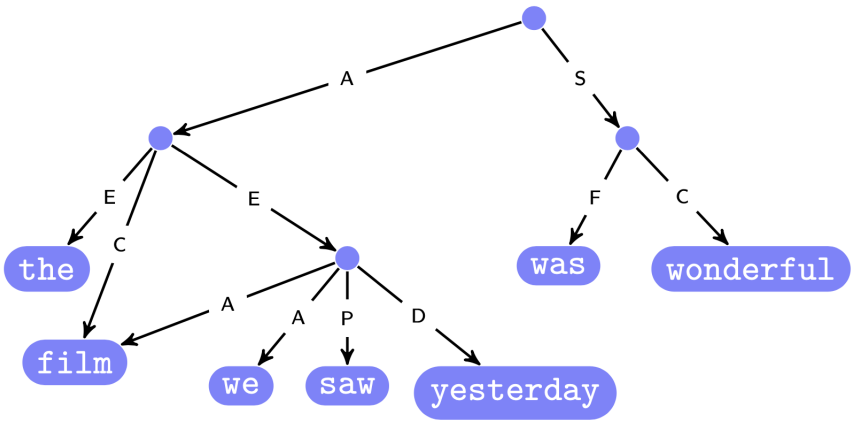


Figure 2.5: UCCA of the sentence *The film we saw yesterday was wonderful.*. P stands for a process that is the main relation of a scene (usually an action or a movement); S stands for a state that is also a main relation but severs for different purposes; A stands for a participant in a broad sense including locations, abstract entities and scenes serving as arguments; D stands for a adverbial that is a secondary including temporal relations; C stands for a center that point out the conceptualization of the parent unit; E stands for a elaborator that is a non-scene relation which applies to a single center (C); F stands for a function that does not introduce a relation or participant.

which are connected with a directed arc. Although EDS abstracts from word surface, there are explicit anchors to sub-string of the sentence. An example is shown in Figure 2.4.

Universal Conceptual Cognitive Annotation (UCCA; Abend and Rappoport 2013a) UCCA captures predicate-argument structure in a sentence by abstracting away from syntactic details. The annotation is intuitive in a typologically-motivated fashion, without relying on language-specific underlying theory. The backbone of UCCA is a directed acyclic graph with lexical tokens as terminals, which can be related to internal nodes with one or more semantic categories (i.e., edges). The scene is the basic meaning-carrying unit, which involves a predicate, participants and modifiers. An example is shown in Figure 2.5. A single scene is evoked by the word *saw* with the relation *Process (P)*, other participants and modifiers are connected with a small set of predefined semantic relations or can be further broken down.

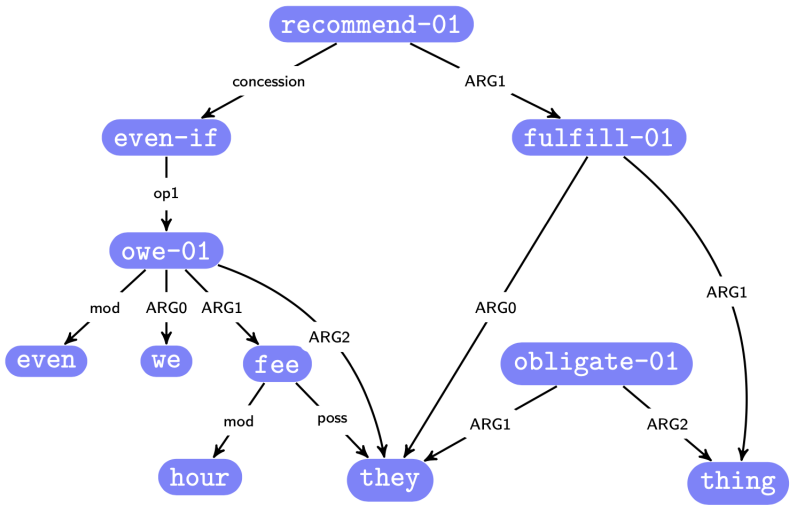


Figure 2.6: AMR of the sentence *Even if we owed their hourly fees, they still should fulfill their obligations.* Each node is a concept, and each edge is labeled as a semantic role from a predefined relation set including general semantic relations (e.g., :direction), and the relations for arguments (e.g., :ARG0), quantities (e.g., :unit), date-entities (e.g., :month), and lists (e.g., :op1)

Abstract Meaning Representation (AMR; Banarescu et al. 2013) AMR represents the meaning of sentences as a rooted, directed and acyclic graph, which is comparable to EDS in dealing with predicate-argument and lexical decomposition. AMR makes extensive use of PropBank framesets for sense differentiation. The underlying property of AMR is abstracting away from syntactic representations to ensure that sentences with the same basic meaning and different syntax, share the same AMR. Without grammar, AMR is agnostic about how we might want to derive meaning from strings or vice-versa. An example is shown in Figure 2.6. ARM graphs adopt the reentrancies to resolve coreference (Szubert et al., 2020).

Rhetorical Structure Theory (RST; Mann and Thompson 1988) RST is proposed to use rhetorical relations (discourse relations) to analyze the text. Different from most linguistic theories, RST focuses on building relations across two or more text spans. RST analyzes texts to several units in two types. Nuclei are considered as the basic information of text, while satellites are secondary that contribute to the nuclei. Based on the consideration of rhetorical relations over texts, Penn Discourse Treebank (PDTB; Prasad et al. 2008) is proposed to produce a large-scale corpus in which discourse connectives are annotated, along with their arguments, thus exposing a clearly defined

	DM	PSD	EDS	UCCA	AMR	RST	DRT
Predicate-Argument	✓	✓	✓	✓	✓		✓
Sense Differentiation	✓	✓	✓		✓		✓
Temporal Expressions						✓	✓
Scope and Quantification							✓
Presupposition							✓
Coreference					✓		✓
Rhetorical Relation						✓	✓
Inter-sentence						✓	✓
Universal				✓			

Table 2.1: The summary of various semantic representations in the corresponding available corpus.

level of discourse structure.

Compared to the semantic representation above, DRT can not only represent predicate-arguments, sense distinctions, and coreference but also cover additional linguistic phenomena such as scope, presupposition, temporal expressions, and rhetorical relations. Also, DRSs can be incrementally constructed on the whole document, and there exist some available corpora that are introduced below. Table 2.1 summarizes these semantic representations' ability to deal with various linguistic phenomena. Representing predicate-arguments is the basic ability of the semantic representations, and sense differentiation is necessary for most of the semantic representations. Only UCCA has no sense differentiation because the purpose of the UCCA development is for the universal concepts instead of English only. As shown in this table, DRT covers all the linguistic information within and across sentences, which is originally designed for English only.

2.4 DRS Corpora

We introduce the corpora that used in our following work. Groningen Meaning Bank (GMB) collects English text with their DRS annotations.[4] Parallel Meaning Bank (PMB) collects English, German, Italian and Dutch text with their DRS annotations.[5]

[4] http://www.let.rug.nl/bjerva/gmb/
[5] https://pmb.let.rug.nl/

sections	# doc	# sent	# token	avg
00-99	10,000	62,010	1,354,149	21.84
20-99	7,970	49,411	1,078,953	21.83
10-19	992	6,116	132,852	21.72
00-09	1,038	6,483	142,344	21.95

Table 2.2: Statistics on the GMB (avg denotes the average number of tokens per sentence).

language	bronze	silver	gold	avg
English	120,622	67,965	5,929	7.92
German	102,998	4,312	1,474	7.77
Italian	61,504	2,550	774	6.89
Dutch	20,554	1,073	711	7.27

Table 2.3: Statistics on the PMB 2.2.0. (avg denotes the average number of tokens per sentence)

GMB DRSs in GMB were obtained from Boxer (Bos, 2008, 2015), and then refined using expert linguists and crowdsourcing methods. Boxer constructs DRSs based on a pipeline of preprocessing tools involving POS-tagging, named entity recognition, and parsing. Specifically, it relies on the syntactic analysis of the C&C parser (Clark and Curran, 2007), a general-purpose parser using the framework of Combinatory Categorial Grammar (CCG; Steedman 2001). DRSs are obtained from CCG parses, with semantic composition being guided by the CCG syntactic derivation.

Documents in GMB were collected from a variety of sources, including *Voice of America* (a newspaper published by the US Federal Government), the Open American National Corpus, Aesop's fables, humorous stories and jokes, and country descriptions from the *CIA World Factbook*. The dataset consists of 10,000 documents annotated with DRSs. Various statistics on the GMB are shown in Table 2.2. Bos et al. (2017) recommend sections 20–99 for training, 10–19 for tuning, and 00–09 for testing.

PMB DRSs in PMB were obtained in a similar way to the GMB, and PMB provides DRS annotations in four languages: English, German, Dutch and Italian. The PMB annotations was grouped into gold data that is fully manually checked, silver data that

are partially manually checked, and bronze data that does not have any manual annotations. Documents in PMB were collected from a variety of sources: Tatoeba1[6], News Commentary (via OPUS, Tiedemann 2012), Recognizing Textual Entailment (Giampiccolo et al., 2007), Sherlock Holmes stories[7], and the Bible (Christodouloupoulos and Steedman, 2015). PMB only contains short texts (i.e., individual sentences) and various statistics on the PMB 2.2.0 are shown in Table 2.3.

2.5 Summary

Discourse Representation Theory is an old but popular theory of meaning representation, which is designed to account for various linguistic phenomena. Compared to other existing semantic formalism, DRT adopts discourse representation structure (DRS) to depict the meaning of text within and across sentence covering more linguistic information. In the past, little work has been done on purely data-driven DRT parsing due to the lack of large-scale available corpora with DRS annotations. As more corpora are released, it is worth to revisit the data-driven DRT models. The success of the neural models in natural language processing further motivates the researches on DRT with neural models. In the following chapters, we show our works on monolingual and cross-lingual DRT parsing and generation with neural networks, and then based on the success of parsing and generation, we explore the opportunity of machine translation via DRS as interlingua.

[6] https://tatoeba.org
[7] http://gutenberg.org, http://etc.usf.edu/lit2go, http://gutenberg.spiegel.de

Chapter 3

Discourse Representation Tree Structure

Semantic parsing is the task of mapping natural language to machine interpretable meaning representations. Various models have been proposed over the years to learn semantic parsers from linguistic expressions paired with logical forms, SQL queries, or source code (Wong and Mooney, 2007; Liang et al., 2011; Zettlemoyer and Collins, 2005; Banarescu et al., 2013; Kwiatkowski et al., 2011; Zhao and Huang, 2015).

Existing semantic parsers are, for the most part, data-driven using annotated examples consisting of utterances and their meaning representations (Zelle and Mooney, 1996; Wong and Mooney, 2006; Zettlemoyer and Collins, 2005). The successful application of encoder-decoder models (Sutskever et al., 2014; Bahdanau et al., 2015) to a variety of NLP tasks have provided a strong impetus to treat semantic parsing as a sequence transduction problem where an utterance is mapped to a target meaning representation in string format (Dong and Lapata, 2016; Jia and Liang, 2016; Kočiský et al., 2016; Foland and Martin, 2017). The fact that meaning representations do not naturally conform to a linear ordering has also prompted efforts to develop recurrent neural network architectures tailored to a tree or graph-structured decoding (Dong and Lapata, 2016; Cheng et al., 2017; Yin and Neubig, 2017; Alvarez-Melis and Jaakkola, 2017; Rabinovich et al., 2017a; Buys and Blunsom, 2017; Damonte et al., 2017; Ballesteros and Al-Onaizan, 2017; Guo and Lu, 2018; Liu et al., 2018b; Zhang et al., 2019; Cai and Lam, 2020a)

Most previous work focuses on building semantic parsers for question answering tasks, such as querying a database to retrieve an answer (Dahl et al., 1994; Zelle and Mooney, 1996; Cheng et al., 2017). Instead, we focus on open-domain semantic pars-

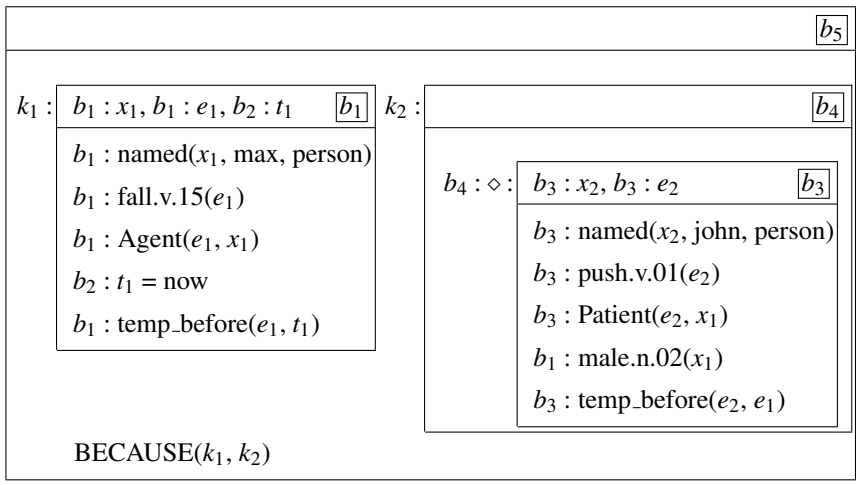
ing for text with arbitrary length and develop a general-purpose system that generates formal meaning representations in the style of Discourse Representation Theory (DRT; Kamp and Reyle 1993). As explained in Chapter 2, DRT is a popular theory of meaning representation (Kamp, 1981; Kamp and Reyle, 1993; Asher, 1993; Asher and Lascarides, 2003) designed to account for a variety of linguistic phenomena, including the interpretation of pronouns and temporal expressions within and across sentences. The basic meaning-carrying units in DRT are Discourse Representation Structures (DRSs) which consist of discourse referents (e.g., x_1, x_2) representing entities in the discourse and discourse conditions (e.g., $\max(x_1)$, $\text{male}(x_1)$) representing information about discourse referents. An example of a two-sentence discourse in box-like format is shown in Figure 3.1(a). DRT parsing resembles the task of mapping sentences to Abstract Meaning Representations (AMRs; Banarescu et al. (2013)) in that logical forms are broad-coverage, and they represent compositional utterances with varied vocabulary and syntax and are *ungrounded*, i.e., they are not tied to a specific database from which answers to queries might be retrieved (Zelle and Mooney, 1996; Cheng et al., 2017; Dahl et al., 1994).

Our work departs from previous general-purpose semantic parsers (Flanigan et al., 2016a; Foland and Martin, 2017; Lyu and Titov, 2018; van Noord et al., 2018b) in that we focus on building representations for *entire* documents rather than *isolated* utterances, and introduce a novel semantic parsing task based on DRT. Specifically, our model operates over Discourse Representation Tree Structures (DRTSs) which are DRSs rendered in a tree-style format (see Figure 3.1(b)). Discourse representation parsing has been gaining more attention lately.[1] The semantic analysis of text beyond isolated sentences can enhance various NLP applications such as information retrieval (Zou et al., 2014), summarization (Goyal and Eisenstein, 2016), conversational agents (Vinyals and Le, 2015), machine translation (Sim Smith, 2017; Bawden et al., 2018), and question anwsering (Rajpurkar et al., 2018).

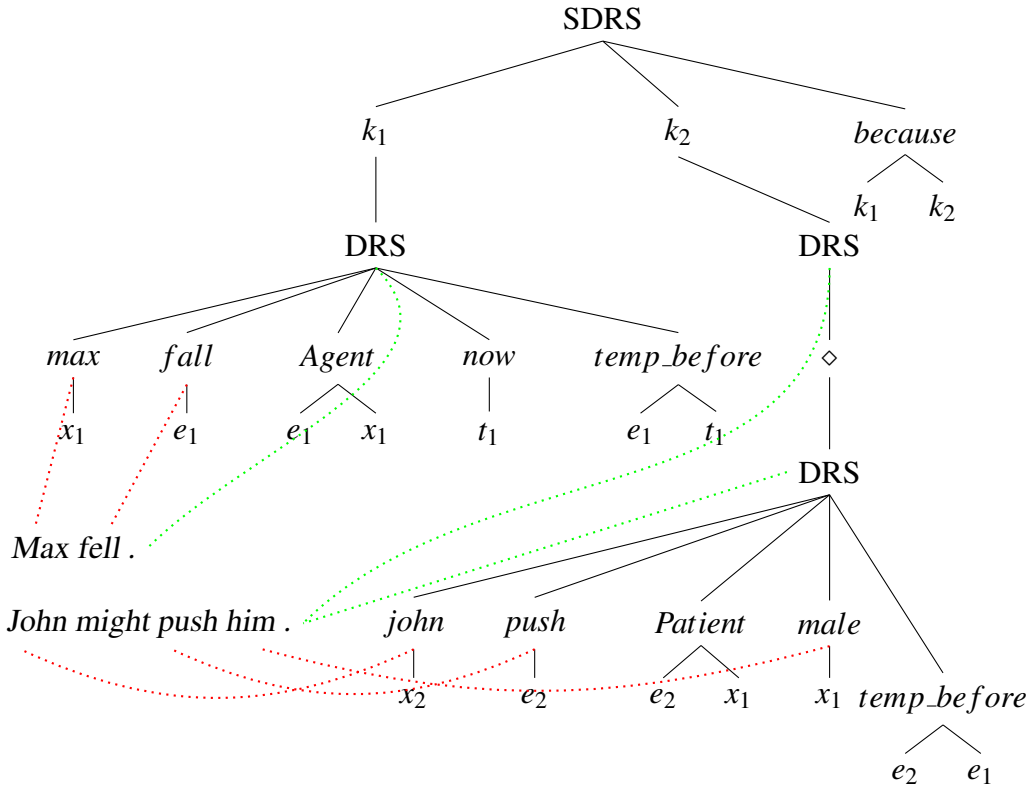
3.1 Discourse Representation Tree Structure

DRSs are displayed in a box-like format that is intuitive and easy to read but not particularly amenable to structure modelling. In this section, we introduce the discourse representation tree structure (DRTS) induced from the DRS boxes, and then we give

[1] The shared task on Discourse Representation Structure parsing in IWCS 2019. https://sites.google.com/view/iwcs2019/home



(a)



(b)

Figure 3.1: Meaning representation for the discourse "*Max fell. John might have pushed him.*" in box-like format (top) and as a tree (bottom). Red lines indicate conditions corresponding to words and green lines indicate DRS nodes corresponding to sentences. $\diamond$ is modality operators for possibility.

the formal definition of Discourse Representation Tree Structure.

3.1.1 DRS-to-DRTS

DRTS is induced from the simplified DRS boxes. we omit referents in the top part of the DRS (e.g., x_1, and e_1 in Figure 3.1(a)) but preserve them in conditions with the assumption that all referents are interpreted in the boxes where they are newly introduced (e.g. $b_1 : x_1$ is simpified to x_1, saying x_1 is presupposed to be interpreted in box b_1), and all conditions are interpreted in the current boxes (e.g., $b_1 : \texttt{male.n.02}(x_1)$ is simplify to $\texttt{male}(x_1)$, saying $\texttt{male}(t_1)$ is presupposed to be interpreted in the current box b_3).[2] Next, we transform the DRS definitions in Chapter 2 to induce the definition of DRTS. Noted that DRTS are represented in bracketed format, i.e. *parent*(*childen*). Definition (2.1) is converted to:

$$\begin{aligned} \langle drs \rangle ::= \text{DRS}(& \\ & [\langle condition \rangle]^* \\ &), \end{aligned} \tag{3.1}$$

where DRS denotes a basic DRS node, e.g. the non-terminal nodes with DRS label in Figure 3.1(b), and $*$ means the scoped representations can appear multiple times. We also modify discourse referents to SDRSs (e.g., k_1, k_2 in Figure 3.1(a)) which we regard as elements bearing scope over expressions $\langle exp_t \rangle$ and add a 2-place predicate $\langle sym_2 \rangle$ to describe the discourse relation between them. So, definition (2.3) becomes:

$$\begin{aligned} \langle sdrs \rangle ::= \text{SDRS}(& \\ & [\langle ref \rangle (\langle exp_t \rangle)]^* \\ & [\langle sym_2 \rangle (\langle ref \rangle, \langle ref \rangle)]^* \\ ), & \end{aligned} \tag{3.2}$$

where SDRS denotes a segmented DRS node, e.g. the non-terminal nodes with SDRS label in Figure 3.1(b), $\langle ref \rangle (\langle exp_t \rangle)$ is a segment, e.g. the subtree k_1(DRS()), and $\langle sym_2 \rangle (\langle ref \rangle, \langle ref \rangle)$ is a segment relation, e.g. the subtree $because(k_1, k_2)$ in Figure 3.1(b).

We treat cardinal numbers $\langle num \rangle$ and $\langle sym_0 \rangle$ in relation *timex* as constants. We introduce the binary predicate "card" to represent cardinality (e.g., $|x_8| = 2$ is $\texttt{card}(x_8, 2)$). We also simplify $\langle exp_e \rangle = \langle exp_e \rangle$ to $\text{eq}(\langle exp_e \rangle, \langle exp_e \rangle)$ using the binary relation "eq" (e.g., $x_1 = x_2$ becomes $\text{eq}(x_1, x_2)$). Moreover, we ignore *class* in *named* and transform $named(\langle exp_e \rangle, \langle sym_0 \rangle, class)$ into $\langle sym_1 \rangle (\langle exp_e \rangle)$ (e.g., $\text{named}(x_1, \text{max}, person)$

[2] We do not consider the senses as well.

becomes $\max(x_1)$). Consequently, basic conditions (see definition (2.5)) are simplified to:

$$\langle basic \rangle ::= \langle sym_1 \rangle (\langle exp_e \rangle) | \langle sym_2 \rangle (\langle exp_e \rangle, \langle exp_e \rangle) \tag{3.3}$$

For example, the condition b_3 : named(x_2, john, person) in Figure 3.1(a) is converted to the subtree $john(x_2)$ in Figure 3.1(b). Analogously, we treat *unary* and *binary* conditions as scoped functions, and definition (2.6) becomes:

$$\begin{aligned} \langle unary \rangle &::= \quad \neg \mid \Box \mid \Diamond \mid \langle ref \rangle (\langle exp_t \rangle) \\ \langle binary \rangle &::= \quad \rightarrow \mid \vee \mid ?(\langle exp_t \rangle, \langle exp_t \rangle), \end{aligned} \tag{3.4}$$

For example, the box b_3, which is scoped by $\diamond$ in Figure 3.1(a), can be converted to the subtree located under the non-termonal node $\diamond$ in Figure 3.1(b).

The tree can be linearized in top-down and left-to-right manner, which is convenient for modeling purposes. The DRTS in Figure 3.1(b) is linearized to the sequence of symbols: "*SDRS(k_1 (DRS(max(x_1) fall(e_1) Agent(e_1 x_1) now(t_1) temp_before(e_1 t_1))) k_2 (DRS($\diamond$(DRS(john(x_3) push(e_2) Patient(e_2 x_1) male(x_1) temp_before(e_2 e_1)))))) because(k_1 k_2))*".

3.1.2 DRTS Definition

A Discourse Representation Tree Structure is represented by a labeled tree over a domain $\mathbb{D} = [R, V, C, N]$ where R denotes relation symbols, V denotes variable symbols, C denotes constants and N denotes scoping symbols. Variables V are indexed and can refer to entities x, events e, states s, time t, propositions p, and segments k. R is the disjoint union of a set of elementary relations R_e and segment relations R_s. The set N is defined as the union of binary scoping symbols N_b and unary scoping symbols N_u, where $N_b = \{\rightarrow, \vee, ?\}$, denoting conditions involving implication, disjunction, and duplex,[3] and $N_u = \{\Diamond, \Box, \neg\}$ denoting modality operators expressing possibility, necessity, and negation.

There are seven types of nodes in a DRTS: *simple* scoped nodes, *proposition* scoped nodes, *segment* scoped nodes, elementary DRS nodes, segmented DRS nodes, atomic nodes and variable nodes. Variables nodes are leaf nodes such that their label is an instantiated variable or a constant $v \in V \cup C$. Atomic nodes can take one of the labels in R, and their children are variable nodes.[4] The subtrees rooted in atomic nodes can

[3] duplex relations are employed for representing quantifiers like "most", because these conditions do not all permit a translation to first-order logic.

[4] In our formulation, the constants are only used for denoting numbers. Proper names are denoted as relations, such as John.

either be unary or binary. For example, in Figure 3.1(b), $\texttt{male}(x_1)$ denotes an atomic node with a unary relation, while $\text{Patient}(e_2, x_1)$ denotes a binary relation node.

A simple scoped node can take one of the labels in N. A node that takes a label from N_u has only one child: either an elementary or a segmented DRS node. A binary scope label node can take two children nodes which are an elementary or a segmented DRS. A proposition scoped node can take as label one of the proposition variables p. Its children are elementary or segmented DRS nodes. A segment scoped node can take as label one of the segment variables k, and its children are elementary or segmented DRS nodes.

An elementary DRS node is labelled with "DRS" and has children (one or more) that are atomic nodes (taking relations from R_e), simple scoped nodes, or proposition scoped nodes. Atomic nodes may use any of the variables except for segment variables k. Finally, a segmented DRS node (labelled with "SDRS") takes at least two children nodes which are segment scoped nodes and at least one atomic node (where the children allowed are the segment variables that were chosen for the other children nodes and the relations are taken from R_s). For example, the root node in Figure 3.1(b) is an SDRS node with two segment variables k_1 and k_2 and the instantiated relation is $\texttt{because}(k_1, k_2)$. The children of the nodes labelled with the segment variables are elementary or segmented DRS nodes. A full DRTS is a tree with an elementary or segmented DRS node as root.

3.2 Problem Formulation

The task of discourse representation tree structure parsing can be formulated as a structure prediction problem. Given a text $S = [s_{11}, ..., s_{ij}, ...]$ where s_{ij} is the jth word in the ith sentence, the parser outputs a discourse representation tree structure $T = [t_1, ..., t_j, ...]$ where t_j is the jth token of DRTS in bracketed string format:

$$T^* = \arg\max_{T \in \mathbb{T}} P(T|S), \tag{3.5}$$

where T^* is the optimal DRTS from the set of all possoble DRTSs $\mathbb{T}$, $P(T|S)$ is the probability of outputting T given S. We choose to model $P(T|S)$ with neural networks which generates a DRTS symbol t_j depending on the historical prediction $t_{<j}$ given the text S,

$$P(T|S) = \prod_j P(t_j|t_{<j}, S, \theta). \tag{3.6}$$

In this formulation, the model parameters θ are optimized with supervised learning methods given a set of training examples $\{(S_i, T_i)\}_{i=1}^{n}$. We investigate a unified framework for sentence- *and* document-level DRTS parsing based on the encoder-decoder architecture (Bahdanau et al., 2015), where the encoder is used to obtain the text representations while the decoder generates the corresponding DRTS.

3.3 Models

3.3.1 Sequential Encoder

Sequential encoder treats the text as a sequence of words, which is so simple that cannot capture the hierarchical structures of the text, which is composed with multiple sentences. Documents (or sentences) are pre-proccessed and represented as a sequence of words $\langle sep_1\rangle$,s_{11},...,$\langle sep_i\rangle$,...,s_{ij},...,$\langle sep_{n+1}\rangle$, where n is the number of sentences, and $\langle sep_i\rangle$ denotes the left boundary of the ith sentence.[5] The jth token in the ith sentence of a document is represented by vector $x_{ij} = f([e_{s_{ij}}; \bar{e}_{s_{ij}}; e_{\ell_{ij}}])$ which is the concatenation (;) of randomly initialized word embeddings $e_{w_{ij}}$, pre-trained word embeddings $\bar{e}_{w_{ij}}$, and lemma embeddings $e_{\ell_{ij}}$ (where $f(\cdot)$ is a non-linear function). Embeddings $e_{w_{ij}}$ and $e_{\ell_{ij}}$ are randomly initialized and tuned during training, while $\bar{e}_{w_{ij}}$ are fixed.

The encoder represents words and sentences in a unified framework compatible with sentence- and document-level DRTS parsing. In experiments, we employed recurrent neural networks with long-short term memory units (LSTMs; Hochreiter and Schmidhuber 1997).

Our sequential encoder represents input text with a Bidirectional LSTM (BiLSTM), which is shown in Figure 3.2:

$$(\overleftrightarrow{h}_{11}, ..., \overleftrightarrow{h}_{ij}, ..., \overleftrightarrow{h}_{nm}) = \text{BiLSTM}(x_{11}, ..., x_{ij}, ..., x_{nm}),$$

where $\overleftrightarrow{h}_{ij}$ $(1 \leq i \leq n, 1 \leq j \leq m)$ denotes the hidden representation of the encoder for x_{ij}, which is the input representation of the jth token in the ith sentence.

[5] The left boundary of sentence i is the right boundary of sentence $i-1$, all $\langle sep_i\rangle$ are the same symbol.

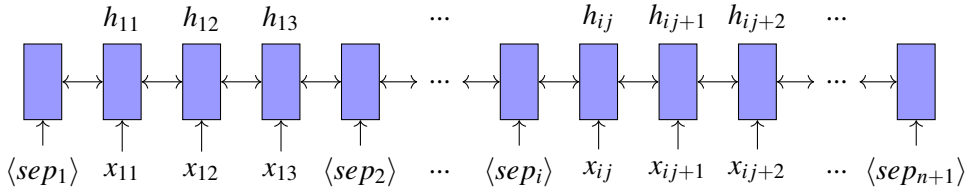


Figure 3.2: The sequential encoder, where x_{ij} and h_{ij} are the input representation and the hidden representation of the jth word in the ith sentence, respectively. $\langle sep_i \rangle$ are the same to be a reserved symbol.

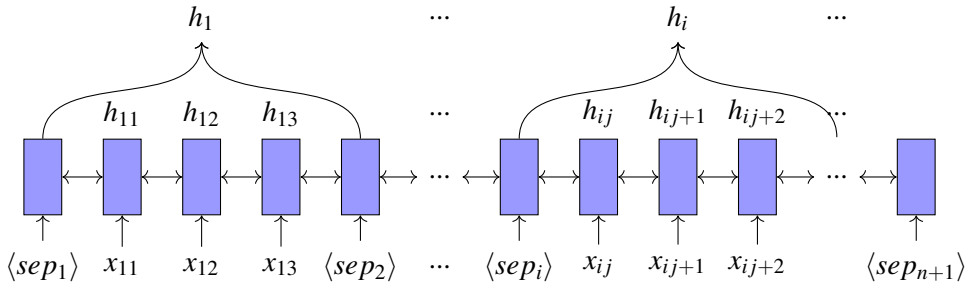


Figure 3.3: The shallow hierarchical encoder, where h_i is the representation of the ith sentence.

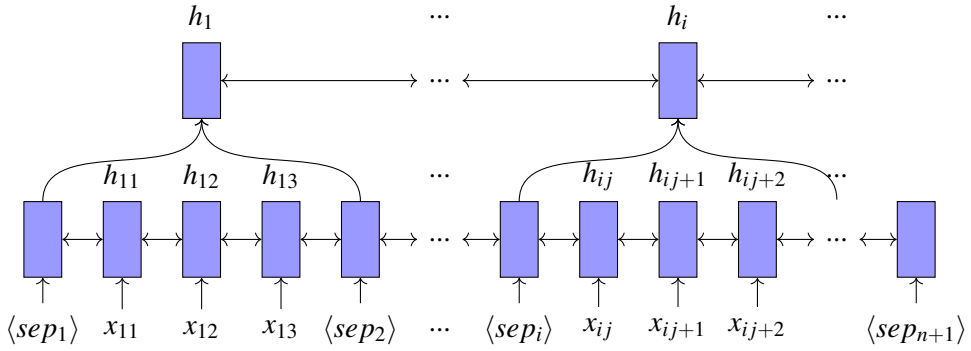


Figure 3.4: The deep hierarchical encoder, the sentence representation in shallow hierarchical encoder is fed to another BiLSTM to get the hidden representation of the sentences h_i.

3.3.2 Hierarchical Encoder

The sequential encoder considers the documents as a sequence of words without taking into account that documents consist of sentences, each of which represents its semantic content. We consider two methods to adopt the hierarchical information of documents.

Shallow Sentence Representation Each sentence can be represented via the concatenation of the forward hidden state of its right boundary and the backward hidden state of its left boundary, i.e., $h_i = [\overrightarrow{h}_{\langle sep_i \rangle}; \overleftarrow{h}_{\langle sep_{i+1} \rangle}]$, which is shown in Figure 3.3.

Deep Sentence Representation An alternative to the shallow sentence representation just described, is a sentence Bidirectional LSTM encoder:

$$(\overleftrightarrow{h_1}, ..., \overleftrightarrow{h_i}, ..., \overleftrightarrow{h_n}] = \text{BiLSTM}(h_1, ..., h_i, ..., h_n), \tag{3.7}$$

which takes h_i, the shallow sentence representation, as input, which is shown in Figure 3.4. Different the shallow encoder which assumes that the boundary symbols can capture the sentence representations, the deep encoder adopt another BiLSTM to capture the contextual features in sentence level, enriching the sentence representation by the interaction among sentences.

3.3.3 Sequential Decoder

The sequential decoder (Bahdanau et al., 2015), which is shown in Figure 3.5, is a (forward) LSTM:

$$h_j^d = \text{LSTM}(e_{j-1}), \tag{3.8}$$

where h_j^d denotes the hidden representation of t_{j-1}, $j-1$th symbol in linearized DRTS, e_{j-1} are randomly initialized embeddings tuned during training, and t_0 denotes the start of sequence.

The decoder uses the contextual representation of the encoder, h_j^{ct}, together with the embedding of the previously predicted token to output the next token from the vocabulary V:

$$a_j = f([h_j^{ct}; e_{t_{j-1}}]), \tag{3.9}$$

where h_{ct_j} is obtained using an attention mechanism:

$$h_j^{ct} = \text{MATTN}(h_j^d, H^e), \tag{3.10}$$

where MATTN is the multiple attention function outputting the context representations given the current hidden representation h_j^d and the set of encoder hidden representations H^e that includes the word representations and the sentence representations (see Section 3.3.5). We obtain the probability distribution over the output tokens as:

$$P_j = P(t_j | t_{<j}, S) = \text{SOFTMAX}(a_j) \tag{3.11}$$

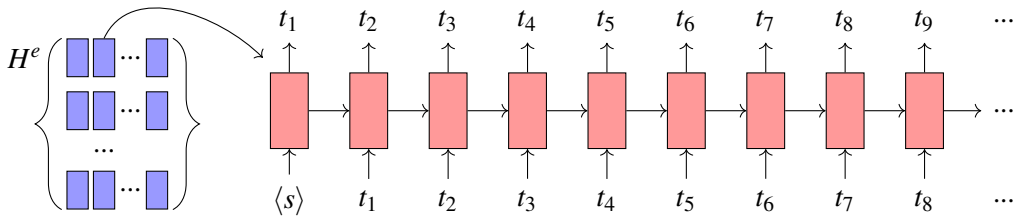


Figure 3.5: The sequential decoder. Red boxes are decoder hidden units while blue boxes are encoder hidden units, which are used for multiple attentions (see Section 3.3.5) on each decoder hidden unit.

3.3.4 Hierarchical Decoder

The structure prediction problem is rather challenging: the length of a bracketed DRTS is nearly 20 times longer (about 5000 tokens) than its corresponding sentence, and the structural patterns are hardly captured sequentially. The simple sequential decoders do not have long memory (Zhao et al., 2020) when producing a really long string. We propose 3-step decoding, which is similar to a coarse-to-fine decoder (Dong and Lapata, 2018), producing several short strings in different steps instead of the whole string in one time, leveraging the short memory effectively.

As shown in Figure 3.1(b), a bracketed DRTS, $t_1, t_2, ..., t_n$ consists of three parts: internal structure $T^{\text{str}} = t_1^{\text{str}}, t_2^{\text{str}}, ...t_u^{\text{str}}$ (e.g., "*SDRS(k_1(DRS( )) k_2(DRS($\diamond$(DRS( )))))*"), conditions $T^{\text{con}} = t_1^{\text{con}}, t_2^{\text{con}}, ..., t_r^{\text{con}}$ (e.g., max, fall, Agent), and referents $T^{\text{ref}} = t_1^{\text{ref}}, t_2^{\text{ref}}, ..., t_v^{\text{ref}}$ (e.g., x_1, e_1), where u is the length of internal structure, r is the number of conditions, v is the number of referents, and $u + r * 2 + v = n$.[6]

The hierarchical decoder take 3-step decoding (see Figure 3.6), which first predicts the structural make-up of the DRTS, then the conditions, and finally their referents in an end-to-end framework. The probability distribution of structured output T given natural language input S is rewritten as:

$$\begin{aligned} P(T|S) = &\ P(T^{\text{str}}, T^{\text{con}}, T^{\text{ref}}|S) \\ = &\ \prod_j P(t_j^{\text{str}}|t_{<j}^{\text{str}}, S) \times \prod_j P(t_j^{\text{con}}|t_{<j}^{\text{con}}, T^{\text{str}}, S) \times \prod_j P(t_j^{\text{ref}}|t_{<j}^{\text{ref}}, T^{\text{con}}, T^{\text{str}}, S), \end{aligned} \tag{3.12}$$

where $T_{<j}^{\text{str}}$, $T_{<j}^{\text{con}}$, and $T_{<j}^{\text{ref}}$ denote the tree structure, conditions, and referents predicted so far. We next discuss how each decoder is modeled.

[6] Each condition has one and only one right bracket.

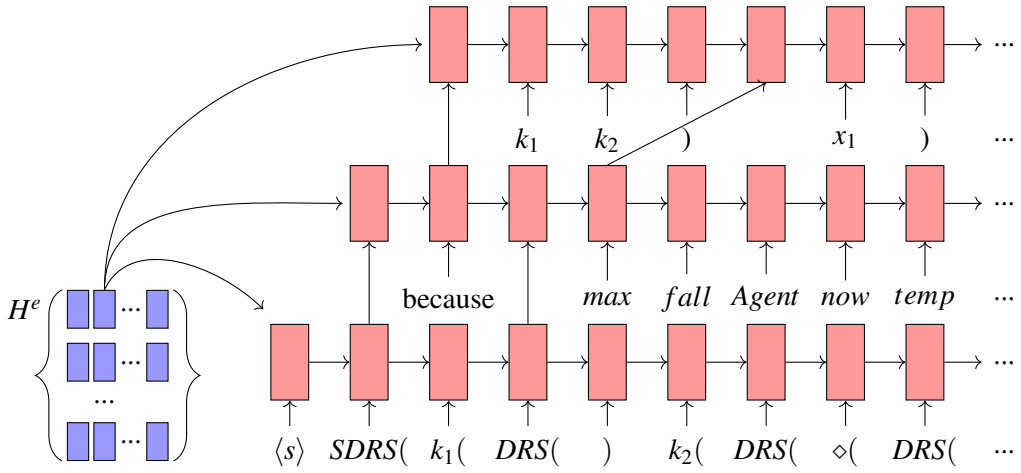


Figure 3.6: The hierarchical decoder. Red boxes are decoder hidden units while blue boxes are encoder hidden units, which are used to attend to each encoder units multiple times.

Stage 1 Our decoder first generates internal structure $t_1^{\text{str}},...,t_u^{\text{str}}$. The probabilistic distribution of the kth prediction is:

$$P(t_k^{\text{str}}|t_{<k}^{\text{str}},S) = \text{SOFTMAX}(a_k^{\text{str}}),$$

where score a_k^{str} is computed as:

$$a_k^{\text{str}} = f([h_k^{\text{str}};\text{MATTN}(h_k^{\text{str}},,H^e)] \tag{3.13}$$

where h_k^{str} is the hidden representation of the decoder in structure prediction, i.e., $h_k^{\text{str}} = \text{LSTM}(e_{k-1}^{\text{str}})$.[7], and H^e is the set of encoder hidden representation.

Stage 2 Given elementary or segmented DRS nodes generated in Stage 1, atomic nodes $t_1^{\text{con}},...,t_r^{\text{con}}$ are predicted, with the aid of copy strategies which we discuss shortly. The probabilistic distribution of the kth prediction is:

$$P(t_k^{\text{con}}|t_{<k}^{\text{con}},T^{\text{str}},S) = \text{SOFTMAX}([a_k^{\text{con}};a_k^{\text{copy}}]),$$

where a_k^{con} and a_k^{copy} are generation and copy scores, respectively, over the kth prediction.

$$a_k^{\text{con}} = f([h_k^{\text{con}};\text{MATTN}(h_k^{\text{con}},,H^e)] \tag{3.14}$$

$$a_k^{\text{copy}} = \text{SCORE}^{\text{copy}}(h_k^{\text{con}},M) \tag{3.15}$$

[7] e_{-1}^{str} is embeddings of the special token $\langle s \rangle$ denoting the start of sequence.

where $\text{SCORE}^{\text{copy}}$ is the function outputting the weights of attentions (see Section 3.3.5), and M are copy representations which is discussed below; and h_k^{con} is the hidden representation of the decoder in Stage 2, which is obtained based on how the previous token was constructed:

$$h_k^{\text{con}} = \begin{cases} \text{LSTM}(g^{\text{copy}}(\bar{h}_{\ell_{k-1}^{\text{con}}})) & t_{k-1}^{\text{con}} \text{ is copied} \\ \text{LSTM}(e_{k-1}^{\text{con}}) & t_{k-1}^{\text{con}} \text{ is generated} \\ \text{LSTM}(g^{\text{str2con}}(h^{\text{str}})) & k=0 \end{cases}, \tag{3.16}$$

where, e_{k-1}^{con} is the input embedding of the t_{k-1}^{con}, and g^* are linear functions. The generation of atomic nodes in the second stage is conditioned on h^{str}, the decoder hidden representation of elementary or segmented DRS nodes from Stage 1 by the linear function g^{str2con}, ℓ_{k-1}^{con} is the lemma of the token t_{k-1}^{con}, and $\bar{h} \in M$ is the copy representations.

For the generation of atomic nodes, we copy lemmas from the input text. However, copying is limited to unary nodes that mostly represent entities and predicates (e.g., john(x_1), eat(e_1)), and correspond almost verbatim to input tokens. Binary atomic nodes denote semantic relations between two variables and do not directly correspond to the surface text. For example, given the DRTS for the utterance "*the oil company is deprived of ...*", nodes oil(x_1) and company(x_2) will be copied from *oil* and *company*, while node of(x_2, x_1) will not be copied from *deprived of*.

Copy representations $M_d = \{\bar{h}_\ell\}^{|L|}$, where $\ell \in L$ and L is the set of distinct lemmas in document d, are constructed from its encoder hidden representations $[h_{11} : h_{nm}]$, by averaging the encoder word representations which have the same lemma:

$$\bar{h}_\ell = \frac{1}{N} \sum_{(ij):\ell_{ij}=\ell} h_{ij},$$

and N is the number of tokens with lemma ℓ.

Stage 3 Finally, we generate referents, $t_1^{\text{ref}}, ..., t_v^{\text{ref}}$ given the internal structures T^{str} and the conditions T^{con} which are generated in Stage 1 and Stage 2, respectively. The probabilistic distribution of the kth prediction is:

$$P(t_k^{\text{ref}} | t_{<k}^{\text{ref}}, T^{\text{con}}, T^{\text{str}}, S) = \text{SOFTMAX}(a_k^{\text{ref}}),$$
$$a_k^{\text{ref}} = f([h_k^{\text{ref}}; \text{MATTN}(h_k^{\text{ref}},, H^e)] \tag{3.17}$$

where h_k^{ref} is the decoder hidden representation in the third stage:

$$h_k^{\text{ref}} = \begin{cases} \text{LSTM}(e_{k-1}^{\text{ref}}) & k \neq 0 \\ \text{LSTM}(g^{\text{con2ref}}(h^{\text{con}})) & k = 0 \end{cases}$$

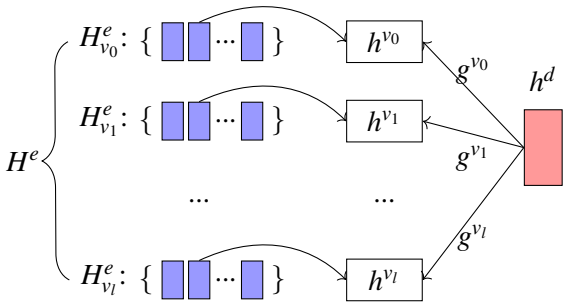


Figure 3.7: The multiple attention component. Linear function $g^v(\cdot)$ transforms decoder hidden representations (red box) into different vector spaces, where v shows which linear function is applied, e.g., $h^{\text{word}} = g^{\text{word}}(h^d)$. Blue boxes are various encoder hidden representations, e.g., H^e_{word} is a sequence of word hidden representations, and H^e_{sent} is a sequence of sentence hidden representations.

Here, the generation of referents is conditioned upon h^{con}, the decoder hidden representation of atomic nodes from the second stage by the linear function g^{con2ref}.

3.3.5 Multiple Attention

The attention mechanism is a standard strategy employed in encoder-decoder frameworks. However, the standard attention strategy (Bahdanau et al., 2015) aims only to extract the feature in a sequence of words. We propose to extend it to be multiple attentions, which can extract features on the given encoder representations in various levels H^e (e.g. words and sentences in our models). The multiple attention is general, allowing us to add more encoder representations to H^e, which is shown in Figure 3.7. Similarly, the multi-head attention strategy (Vaswani et al., 2017) splits the hidden representation to different parts and does attention for each part of hidden representation. Differently, we identify the attentions with names which are allowed to be supervised (see Section 3.3.7).

Given a set of various encoder hidden representations $H^e = \{H^e_v\}^{|V|}$ where $v \in V$ is a name identifying the representations and V is the set of names (e.g. "word" means word-level encoder representations, and "sent" means sent-level encoder representations) and the decoder hidden representation h^d, the multiple attention is computed as:

$$\text{MATTN}(h^d, H^e) = \{h^{\text{att}}_v\}^{|V|} = \{\text{ATTN}^v(h^d, H^e_v)\}^{|V|}, \tag{3.18}$$

where ATTN^v is function with the standard attention mechanism (Bahdanau et al.,

2015):

$$\text{ATTN}^v(h^d, H_v^e = [h_1, ..., h_i, ..., h_m]) = \sum_{i=1}^{m} \beta_i h_i, \tag{3.19}$$

where weight β_i is computed by:

$$\beta_i = \frac{\exp(h^v \cdot h_i)}{\sum_{o=1}^{m} \exp(h^v \cdot h_o)}, \tag{3.20}$$

where $h^v = g^v(h^d)$ and $g_v(\cdot)$ is a linear function identified with the name v (e.g., g^{word}) which transforms h^d to another vector space named v. Multi-attention scores can be also obtained from the attention weights:

$$\text{SCORE}^v(h^d, H_v^e = [h_1, ..., h_m]) = [\beta_1 : \beta_m] \tag{3.21}$$

The sequential encoder in Section 3.3.1 only has word-level encoder hidden representation H^e_{word}, so the set of encoder hidden representation is $H^e = \{H^e_{\text{word}}\}$. The hierarchical encoder in Section 3.3.2 has sentence-level hidden representation H^e_{sent} in addition, so the set of encoder hidden representations is $H^e = \{H^e_{\text{word}}, H^e_{\text{sent}}\}$. The multiple attention component can represent more features in various levels, e.g., syntactic structures.

3.3.6 Model Training

The model is trained to minimize a cross-entropy loss objective over the whole training dataset:

$$L(\theta) = -\sum_{(S,T)\in\mathcal{D}} \log P(T|S,\theta) \tag{3.22}$$

where $\mathcal{D}$ is the training dataset which consists of pairs of DRTS T and text S, and θ are the parameters of the model. We use stochastic gradient descent and adjust the learning rate with Adam (Kingma and Ba, 2014).

3.3.7 Supervised Attention

The attention mechanism from Section 3.3.5 can automatically learn alignments between encoder and decoder hidden representations. However, as shown in Figure 3.1(b), DRTSs are constructed recursively, and alignment information between DRTS nodes and sentences is available, indicating which sentences the subtrees are attending to describe, and the alignments discover the semantic accommodation in the text. For

example, the subtree scoped by the ⋄ is aligned to the second sentence, and the model should put more attention on the second sentence when generating the subtree.

For this reason, we propose a method to explicitly learn this alignment by exploiting the feature representations afforded by multi-attention. Specifically, we obtain alignment weights via multi-attention:

$$\textsc{Score}^{\text{align}}(h_k, H^e_{\text{sent}}) = [\beta_{k1} : \beta_{km}]$$

where $\beta_{km} = P(\pi_k = m | h_k, H^e_{\text{sent}})$, i.e., the probabilistic distribution over alignments from sentences to the kth prediction in the decoder, where $\pi_k = m$ denotes the kth prediction aligned to the mth sentence. We add an alignment loss to the objective in Equation (3.22):

$$L(\theta) = -\sum_{(S,T)\in\mathcal{D}} \log P(T|S,\theta) - \sum_{(S,A,T)\in\mathcal{D}} \log P(A|T,S,\theta) \tag{3.23}$$

where A are the alignments showing the sentences that each node in DRTS is aligned to. We then use these alignments in two ways.

Alignments as Features Alignments are incorporated as additional features in the decoder by concatenating the aligned sentence representations with the scoring layers. Equations (3.13), (3.14), and (3.17) are thus rewritten as:

$$a^c_k = f([h^c_k; \textsc{MAttn}(h^c_k, H^e); h_{\pi_k}]), \tag{3.24}$$

where $c \in \{\text{str}, \text{con}, \text{ref}\}$, and h_{π_k} is the π_kth sentence representation when making kth prediction.

At test time, the scoring layer requires the alignment information, so we first select the sentence with the highest probability, i.e., $\pi^*_k = \arg\max_{\pi_k} P(\pi_k | h_k, H^e_{\text{sent}})$, and then add its representation $h_{\pi^*_k}$ to the scoring layer.

Copying from Alignments We use alignment as a means to modulate which information is copied. Specifically, we allow copying to take place only over sentences aligned to elementary DRS nodes. We construct copy representations for each sentence in a document, i.e., $M_1, ..., M_i, ..., M_n$ where $M_i = \{\bar{h}_{\ell_k}\}^{|L_i|}$, $\ell_i \in L_i$, and L_i is the set of distinct lemmas in the ith sentence:

$$\bar{h}_{\ell_i} = \frac{1}{N} \sum_{(ij):\ell_{ij}=\ell_i} h_{ij},$$

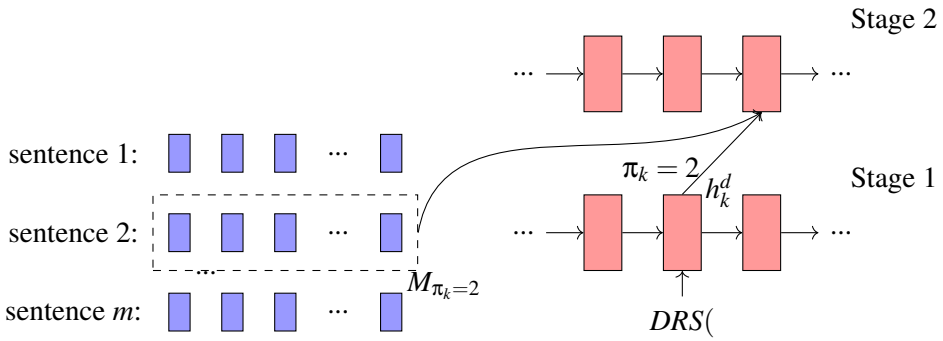


Figure 3.8: Example of the inference with supervised alignments, where in the first stage, the kth prediction shows the alignment to the sentence $\pi_k = 2$ which words can be copied during the predictions conditioning on h_k^d in the second stage.

Given the alignment between elementary DRS nodes and sentences, we calculate the copying score by rewriting Equation (3.15) as:

$$a_k^{\text{copy}} = \text{Score}^{\text{copy}}(h_k^{\text{con}}, M_{\pi_k})$$

where π_k is the index of the sentence that is aligned to the elementary DRS node, and h_k^{con} is the decoder hidden representation in Stage 2 which generate the condition t_k.

At test time, when an elementary DRS is generated in the first stage, we further predict which sentence the node should be aligned to. The information is then passed onto the second stage, and elements from the aligned sentence can be copied. An example is shown in Figure 3.8.

3.3.8 Constraint-based Inference

Recall that the hierarchical decoder consists of three stages, each of which is a sequence-to-sequence model. As a result, there is no guarantee that tree output will be well-formed. To ensure the generation of syntactically valid trees, at each step, we generate the set of valid candidates T_k^{valid} which do not violate the DRTS definitions in Section 3.1, and then select the highest scoring tree as our prediction:

$$t_k^* = \underset{t_k \in T_k^{\text{valid}}}{\arg\max}\, P(t_k | t_{<k}, S, \theta),$$

where θ are the parameters of the model, and T_k^{valid} the set of valid candidates at step k.

In order to guide sequential predictions, we define a *state tracker* equipped with four functions: INITIALIZATION initializes the stack tracker, UPDATE updates the

Algorithm 1 Inference with State Tracker

1: **procedure** INFERENCE(ST, θ)
2: INITIALIZATION(ST)
3: $k = 1$
4: **repeat**
5: T_k^{valid} = VALID(ST)
6: $t_k^* = \arg\max_{t_k \in T_k^{\text{valid}}} P(t_k | t_{<k}, \theta)$
7: UPDATE(ST, t_k^*); $k = k + 1$
8: **until** ISTERMINATED(ST)
9: return $[t_1^*, ..., t_k^*]$
10: **end procedure**

stack tracker according to token t, ISTERMINATED determines whether the state tracker should terminate, and VALID returns the set of valid candidates in the current state. The state tracker provides an efficient interface for applying constraints during decoding. Sequential inference with the state tracker is shown in Algorithm 1; θ are model parameters and T_k^{valid} all possible valid predictions at step k.

Stage 1 Algorithm 2 implements the state tracker functions for Stage 1; *DRS* denotes an elementary DRS node (i.e., "DRS("), *SDRS* denotes a segmented DRS node (i.e., "SDRS("), *prop*SN is short for *proposition* scoped node (e.g., "p_1("), *segm*SN is short for *segment* scoped node (e.g., "k_1("), *simp*SN is short for *simple* scoped node (e.g., "¬("), and *EndSym* denotes a right bracket (e.g., ")").

Function INITIALIZATION (lines 1–3) initializes the state tracker as an empty stack with a counter. Lines 4–11 implement the function UPDATE, where the predicted token t is placed on top of the stack if it is not an *EndSym* (line 6) and the counter is incremented if t is an elementary DRS node (line 7). The top of the stack is popped if t is an *EndSym*, i.e., the children of the node on top of the stack have been generated, and the stack is updated (line 9).

Lines 12–14 implement the function ISTERMINATED. If the stack is empty, decoding in Stage 1 is completed. Function ISTERMINATED is called after function UPDATE has been called at least once (see lines 7–8 in Algorithm 1).

Lines 15–29 implement the function VALID, which returns the set of valid candidates T^{valid} in the current state. If the stack is empty, which means that a root of a DRTS should be constructed, T^{valid} only includes elementary and segmented DRS

Algorithm 2 State Tracker for Stage 1

```
1:  procedure INITIALIZATION(ST)
2:      ST.stack = []; ST.count = 0
3:  end procedure
4:  procedure UPDATE(ST, t)
5:      if t is not EndSym then
6:          ST.stack.push(t)
7:          if t is DRS then ST.count += 1
8:      else
9:          ST.stack.pop(); ST.stack.top.childnum += 1
10:     end if
11: end procedure
12: procedure ISTERMINATED(ST)
13:     return ST.stack.isempty()
14: end procedure
15: procedure VAILD(ST)
16:     if ST.stack.empty() then return {DRS, SDRS}
17:     top = ST.stack.top
18:     if top is propSN or segmSN or unary simpSN then
19:         if top.childnum < 1 then return {DRS, SDRS}
20:     else if top is binary simpSN then
21:         if top.childnum < 2 then return {DRS, SDRS}
22:     else if top is DRS then
23:         if ST.count < MAX_DRS then return {EndSym, propSN, simpSN}
24:     else if top is SDRS then
25:         if top.childnum < 2 then return {segmSN}
26:         if ST.count < MAX_DRS then return {EndSym, segmSN}
27:     end if
28:     return {EndSym}
29: end procedure
```

nodes (line 16). We use *top* to denote the top node of the stack (line 17). If *top* is a *proposition* scoped node or *segment* scoped node or unary *simple* scoped node, T^{valid} includes an elementary and segmented DRS node only if *top* has no children (lines 18–19). Similarly, binary *simple* scoped nodes should only have two elementary

Algorithm 3 State Tracker for Stage 2

```
1: procedure INITIALIZATION(ST, type)
2:     ST.count = 0; ST.completed = False; ST.type = type
3: end procedure
4: procedure UPDATE(ST, t)
5:     ST.count += 1
6:     if y is EndSym then ST.completed = True
7: end procedure
8: procedure ISTERMINATED(ST)
9:     return ST.completed
10: end procedure
11: procedure VALID(ST)
12:     if ST.count = 0 then return R_{ST.type}
13:     if ST.count < MAX_REL_{ST.type} then return R_{ST.type} ∪ {EndSym}
14:     return {EndSym}
15: end procedure
```

or segmented DRS nodes as children (lines 20–21). If *top* is an elementary DRS node and the number of elementary DRS nodes is within the threshold MAX_DRS according to the training data, *top* can have more children, i.e., T^{valid} includes *EndSym* and scoped nodes, except *segmented* scoped, nodes (lines 22–23). If *top* is a segmented DRS node and has less than two children, T^{valid} only includes segment scoped nodes (line 25). Furthermore, if the number of elementary DRS nodes is within the threshold MAX_DRS, *top* can have more children, i.e., T^{valid} includes *EndSym* and *segmented* scoped nodes (line 26).

Stage 2 The state tracker functions for Stage 2 are shown in Algorithm 3. Lines 1–3 implement the function INITIALIZATION, which initializes state tracker as a relation counter, a type flag, and a completed flag. The relation counter records the number of relations that have been already constructed. The type flag shows the type of nodes, i.e., *e* for elementary DRS nodes or *s* for segmented DRS nodes, based on which the relations are constructed. The completed flag checks if the construction is completed. Line 4–7 implement the function UPDATE. If *EndSym* is predicted, the completed flag is set to true, and the completed flag is checked (lines 8–10, function ISTERMINATED).

Lines 11–15 implement the function VALID. If the number of constructed relations

is zero, T^{valid} only includes R (line 12). If the number of constructed relations is within the threshold $\text{MAX\_REL}_{\text{ST.type}}$, it is possible to construct more relations (line 13). If the number of children exceeds the threshold, T^{valid} only includes *EndSym* to complete the construction of relations (lines 14).

Stage 3 Algorithm 4 implements the state tracker functions for Stage 3, where V_e includes entity variables, event variables, state variables, time variables, proposition variables, and constants, and V_s includes segment variables. Lines 1–3 (INITIALIZATION) initialize the state tracker with a variable counter, a type flag, and a completed flag. The variable counter records the number of variables that have already been constructed. The type flag shows the type of nodes (e for elementary DRS nodes or s for segmented DRS nodes), based on which the variables are constructed. The completed flag checks if the construction is completed. Lines 4–7 implement the function UPDATE. If *EndSym* is predicted, the completed flag is set to true and checked (lines 8–10, function ISTERMINATED).

Lines 11–18 implement the function VALID. If no variables are constructed, T^{valid} only includes $V_{\text{ST.type}}$ (lines 16–17). If only one variable is constructed and ST.type is a segmented DRS, T^{valid} only includes V_s to construct one more variable because relations in segmented DRS nodes are binary (line 13–15). If two variables are constructed, T^{valid} only includes *EndSym* (line 17).

3.4 Evaluation

Due to the complex nature of our structured prediction task, we cannot expect model output to exactly match the gold standard. For instance, the numbering of the referents may be different, but nevertheless valid, or the order of the children of a tree node (e.g., "DRS(india(x_1) say(e_1))" and "DRS(say(e_1) india(x_1))" are the same). We thus use F_1 instead of exact match accuracy.

We adopt Counter (van Noord et al., 2018a), a metric designed to evaluate scoped meaning representations. Counter is based on Smatch[8], a metric used to evaluate AMR graphs (Cai and Knight, 2013). The metric calculates F_1 over two sets of clauses (see Chapter 2), applying multiple restarts to obtain a best variable mapping having the most matched clauses. For example, given predicted and gold DRS shown in Figure 3.9: where the best variable mappings from predicted DRS to gold DRS are {b_1:b_1,

[8] https://github.com/snowblink14/smatch

Algorithm 4 State Tracker for Stage 3

```
1: procedure INITIALIZATION(ST, type)
2:     ST.count = 0; ST.completed = False; ST.type = type
3: end procedure
4: procedure UPDATE(ST, t)
5:     ST.count += 1
6:     if y is EndSym then ST.completed = True
7: end procedure
8: procedure ISTERMINATED(ST)
9:     return ST.completed
10: end procedure
11: procedure VALID(ST)
12:     if ST.count = 0 then return V_ST.type
13:     if ST.count = 1 then
14:         if ST.type is elementary DRS then return V_ST.type ∪ {EndSym}
15:         if ST.type is segmented DRS then return V_ST.type
16:     end if
17:     return {EndSym}
18: end procedure
```

Predicted	Gold
b_1 max e_1	b_1 max x_1
b_1 stop x_3	b_1 stop e_1
b_1 Agent x_3 e_1	b_1 Agent e_1 x_1
b_3 john x_2	b_2 john x_2
b_3 Patient x_2 x_2	b_2 Patient e_1 x_2
	b_2 now t_1
	b_2 temp_after e_1 t_1

Figure 3.9: The predicted and gold DRSs in clause format.

b_3:b_2, e_1:x_1, x_3:e_1, x_2:x_2}. So the precision, recall and F-score are computed as:

$$\begin{aligned} Precision(P) &= \frac{\text{the number of correctly predicted clauses}}{\text{the number of predicted clauses}} \\ Recall(R) &= \frac{\text{the number of correctly predicted clauses}}{\text{the number of gold clauses}} \\ \mathrm{F}_1 &= \frac{2*P*R}{P+R} \end{aligned} \tag{3.25}$$

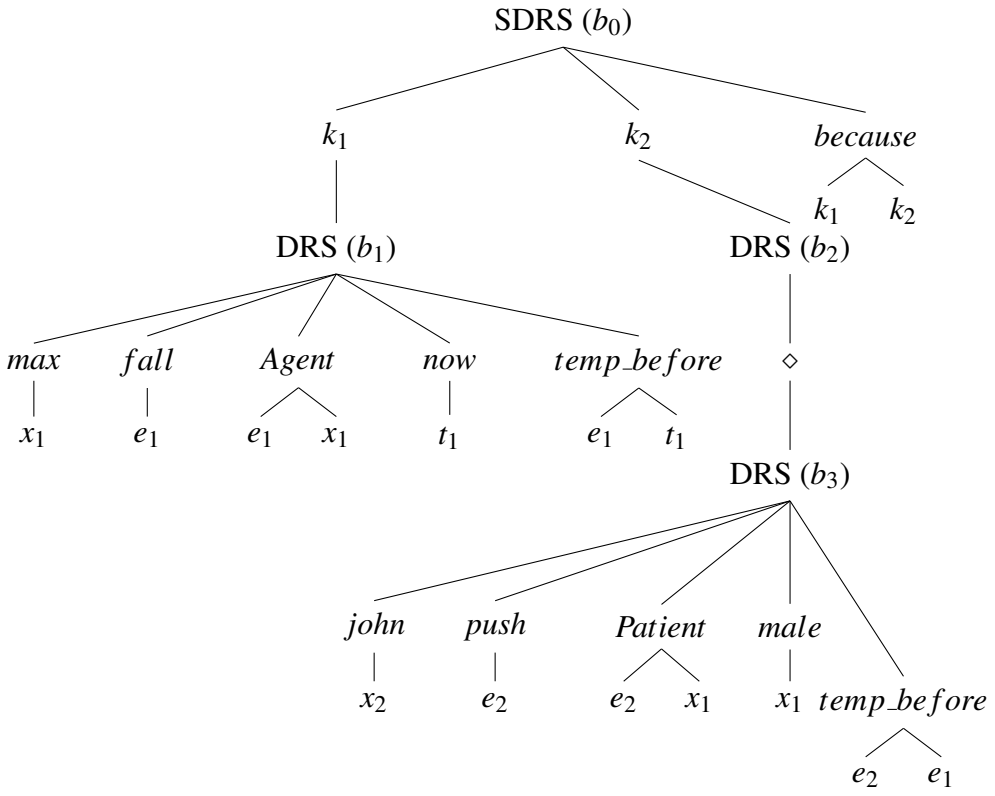


Figure 3.10: The DRTS with box labels, where each SDRS or DRS node is assigned with a unique label.

We converted DRTSs into a set of clauses in the top-down procedure by two steps. First, the elementary DRS nodes and segmented DRS nodes are assigned with unique labels identify the nodes. For example, the DRTS in Figure 3.1(b) is labelled to be DRTS in figure 3.10. Then, for each atomic node or scoped node, clauses are obtained with the box label of its parent node and the variables or constants of its children. In Figure 3.10, take the atomic node *max* as an example, two clause "b_1 REF x_1" and "b_1 max x_1" are obtained where b_1 is the box label of its parent node, and x_1 is the variable of its child. Similarly, the clause "$b_2 \diamond b_3$" is obtained for the scoped node $\diamond$. [9]

Algorithm 5 show the detailed conversion. IsAtomic returns *true* if the child is a atomic node (e.g., india(x_1)) or a simple/proposition scoped node (e.g., p_1, $\neg$ and $\Box$), where one clause is created with a box label, a relation and arguments (line 6). The algorithm will recursively travel all DRS or SDRS nodes (line 11). Furthermore, the clauses are introduced to connect DRS or SDRS nodes to the referents that first appear in a condition (lines 5), where "REF" is to introduce general variables, such as x, e, s,

[9] Due to having the unique labels for each DRS and SDRS node, we squash out the segmented scope by replacing k with the DRS/SDRS unique labels b.

t and p, and "DRS" is to introduce box labels, b. The set of clauses for the DRTS in Figure 3.10 is:

b_1 REF x_1	b_1 temp_before e_1 t_1	b_3 Patient e_2, x_1
b_1 max x_1	$b_2 \diamond b_3$	b_3 male x_1
b_1 REF e_1	b_3 REF x_2	b_3 temp_before e_2 e_1
b_1 fall e_1	b_3 john x_2	b_0 DRS b_1
b_1 Agent e_1 x_1	b_3 REF e_2	b_0 DRS b_2
b_1 REF t_1	b_3 push e_2	b_0 because b_1 b_2
b_1 now t_1		

Algorithm 5 DRTS-to-Clause Conversion

Input: T, DRTS

Output: G, a set of clauses

```
1:  G ← Ø; R ← Ø
2:  procedure TRAVERSALDRS(parent)
3:      for child in parent do
4:          if ISATOMIC(child) or ISSIMPROP(child) then
5:              ADDREFERENT(parent.b, child)
6:              G ← G ∪ {(top  child.rel  child.args)}
7:          end if
8:          TRAVERSALDRS(child.nextDRS)
9:      end for
10: end procedure
11: TRAVERSALDRS(T)
12: return G
```

3.5 Experiments

Our experiments were carried out on the Groningen Meaning Bank (GMB; Bos et al. 2017) which provides a large collection of English texts annotated with Discourse Representation Structures. We preprocessed the GMB into the tree-based format defined in Section 3.1 and created two benchmarks, one which preserves document-level boundaries and a second one which treats sentences as isolated instances. Various statistics on these are shown in Table 3.1.

	Sentences		Documents			
	#sent	avg_w	#doc	#sent	avg_s	avg_w
train	41,563	21.1	7,843	48,599	6.2	135.3
dev	5,173	21.0	991	6,111	6.2	134.0
test	5,451	21.2	1,035	6,469	6.3	137.2

Table 3.1: Statistics on the GMB sentence- and document-level benchmarks (avg_w denotes the average number of words per sentence (or document), avg_s denotes the average number of sentences per document).

3.5.1 Settings

We carried out experiments on the sentence- and document-level GMB benchmarks in order to evaluate our framework. We used the same empirical hyper-parameters for sentence- and document-level parsing. The dimensions of word and lemma embeddings were 300 and 100, respectively. The encoder and decoder had two layers with 300 and 600 hidden dimensions, respectively. The dropout rate was 0.1. Pre-trained word embeddings (100 dimensions) were generated with Word2Vec trained on the AFP portion of the English Gigaword corpus.

We present two sets of experiments focusing on sentence- and document-level DRTS parsing, respectively. In order to investigate the benefit of the proposed hierarchical decoder, we compared three models focusing on sentence-level predictions: Seq2Seq is equipped with a standard sequential decoder, SentShallow is equipped with a sequential decoder and a copy mechanism, and SentDeep is equipped with a hierarchical decoder and a copy mechanism. All models are equipped with a sequential encoder (the input is a single sentence, and there is no need for a hierarchical encoder).

In order to investigate the hierarchical encoder, we take the document-level experiments where we built two baseline models. The first one treats documents as one long string (by concatenating all document sentences) and performs sentence-level parsing (DocSent). The second one parses each sentence in a document with a parser trained on the sentence-level version of the GMB and constructs a (flat) document tree by gathering all sentential DRTSs as children of a segmented DRS node (DocTree). We used the sentence-level DRTS parser for both baselines. We also compared four variants of our document-level model: one with multi-attention and shallow sentence repre-

Models	dev	test
Seq2Seq	56.92	57.69
SentShallow	65.63	65.24
SentDeep	**77.29**	**77.54**
w/o *pre*	75.87	-
w/o *pre* & *lem*	75.42	-

Table 3.2: Results on sentence-level GMB benchmark, where *pre* means the pretrained word embeddings and *lem* means the lemma embeddings. The highest scores are bold.

sentations (DocShallow); one with multi-attention and deep sentence representations (DocDeep); a DocDeep model with supervised attention and alignments as features (DocDeepFeat); and finally, a DocDeep model with copying modulated by supervised attention (DocDeepCopy). All variants of our DRTS parser and comparison models adopt constraint-based inference.

3.5.2 Results

Table 3.2 summarizes results on the sentence-level semantic parsing task. As can be seen, SentShallow performs better than Seq2Seq, and SentDeep with a hierarchical decoder outperforms both of them. Ablation experiments show that without pre-trained word embeddings or word lemma embeddings, the model generally performs worse. Compared to lemma embeddings, pre-trained word embeddings contribute more.

Table 3.3(a) presents various ablation studies for the document-level model on the development set. Deep sentence representations, when combined with multi-attention, bring improvements over shallow representations (+3.68 F_1). Using alignments as features and as a way of highlighting where to copy from yields further performance gains both in terms of F_1. The best performing variant is DocDeepCopy which combines supervised attention with copying. Table 3.3(b) shows our results on the test set; we compare the best performing DRTS parser (DocDeepCopy) against two baselines that rely on our sentence-level parser (DocSent and DocTree). DocDeepCopy, which has a global view of the document, outperforms variants that construct document representations by aggregating individually-parsed sentences.

Models	F_1
DocShallow	61.74
DocDeep	65.42
DocDeepFeat	66.43
DocDeepCopy	**69.45**

(a)

Models	F_1
DocSent	53.27
DocTree	58.22
DocDeepCopy	**66.56**

(b)

Table 3.3: (a) Results (dev set) on document-level GMB benchmark; (b) Results (test set) on document-level GMB. The highest scores are bold.

DocDeepCopy	atomic	scoped	DRS	All
sentences	0.22	0.26	1.78	2.09
documents	3.57	4.54	25.02	30.75

Table 3.4: Percentage of ill-formed outputs without constraints during inference (test set); atomic refers to atomic nodes, scoped refers to scoped nodes and DRS referes to DRS nodes (from Section 3.1) violated.

3.5.3 Analysis

In Table 3.4, we examine whether the constraint-based inference is helpful. In particular, we show the percentage of ill-formed DRTSs when constraints are not enforced. We present results for sentence- and document-level parsers overall and broken down according to the type of DRTS nodes being violated. 30.75% of document level DRTSs are ill-formed when constraints are not imposed during inference. This is in stark contrast with sentence-level outputs that are mostly well-formed (only 2.09% display violations of any kind). We observe that most violations concern elementary and segmented DRS nodes. The time complexity of our state tracker algorithms is $O(1)$, and the space complexity is $O(n)$ that is near $O(1)$, because n is the depth of the trees and trees are flat. In the implementations, we enforce the invalid prediction with masks without any losses.

Figure 3.11 shows how our parser (DocDeepCopy variant) and comparison systems perform on documents of varying length. Unsurprisingly, we observe that F_1 decreases with document length and that all systems have trouble modelling documents with 10 sentences and beyond. In general, DocDeepCopy has an advantage over

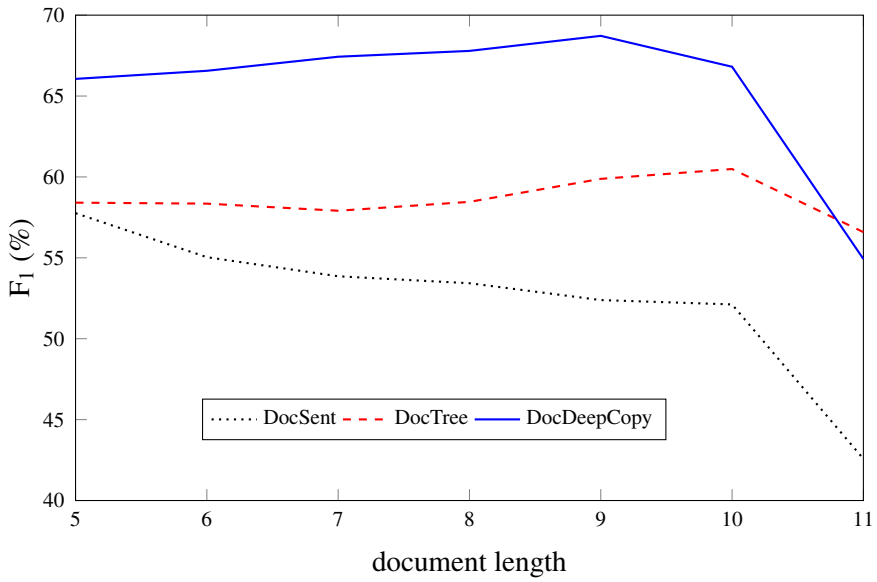


Figure 3.11: Model performance (F1%) as a function of document length (i.e., number of sentences).

comparison systems due to the more sophisticated alignment information and the fact that it aims to generate global document-level structures. Our results also indicate that modelling longer documents that are relatively few in the training set is challenging mainly because the parser cannot learn reliable representations for them. Moreover, as the size of documents increases, ambiguity for the resolution of coreferring expressions increases, suggesting that explicit modelling of anaphoric links might be necessary.

We provide example output of our model (DRTS parser, DocDeepCopy variant) for the GMB document below in Figure 3.12.

European Union energy officials will hold an emergency meeting next week amid concerns that the Russian-Ukrainian dispute over natural gas prices could affect EU gas supplies. An EU statement released Friday says the meeting is aimed at finding a common approach. It also expresses the European Commission's concern about the situation, but says the EU top executive body remains confident an agreement will be reached. A Russian cut-off of supplies to Ukraine will reduce the amount of natural gas flowing through the main pipeline toward Europe. But the commission says there is no risk of a gas shortage in the short term. German officials say they are hoping for a quick resolution to the dispute. Government spokesman, Ulrich Wilhelm says officials have been in contact with both sides at a working level, but will not mediate.

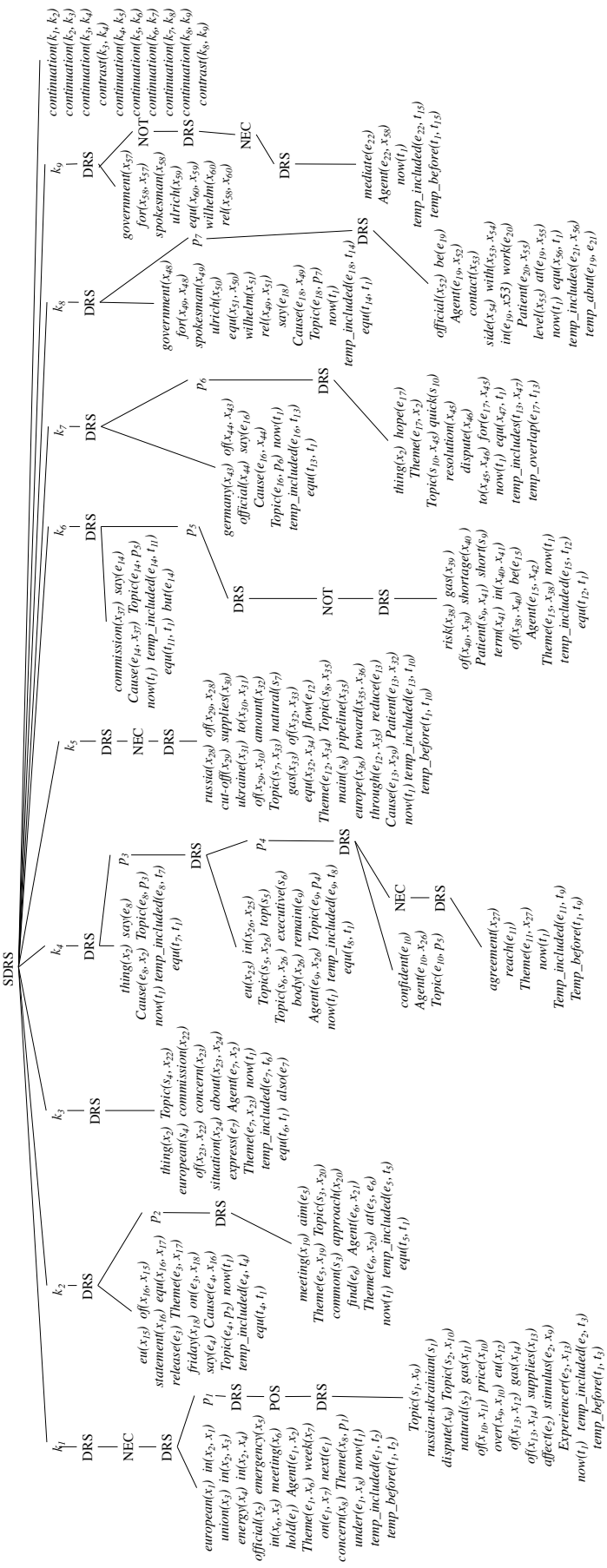


Figure 3.12: Output of DRTS parser (DocDeepCopy variant) for the document in Section 3.5.3

3.6 Related Work

A few recent approaches develop structured decoders which make use of the syntax of meaning representations. Dong and Lapata (2016) and Alvarez-Melis and Jaakkola (2017) generate trees in a top-down fashion, and some works incrementally generate trees and graphs in transition-based methods (Damonte et al., 2017; Ballesteros and Al-Onaizan, 2017; Guo and Lu, 2018; Liu et al., 2018b), while in other work (Xiao et al., 2016; Krishnamurthy et al., 2017) the decoder generates from a grammar that guarantees that predicted logical forms are well-typed. In a similar vein, Yin and Neubig (2017) generate abstract syntax trees (ASTs) based on the application of production rules defined by the grammar. Rabinovich et al. (2017b) introduce a modular decoder whose various components are dynamically composed according to the generated tree structure. In comparison, our model does not use grammar information *explicitly*. We first decode the structure of the DRTS, and then fill in details pertaining to its semantic content. Our model is not strictly speaking top-down, because we generate partial trees sequentially and then expand non-terminal nodes, ensuring that when we generate the children of a node, we have already obtained the coarse structure of the entire tree. The 3-step decoding is more related to the recent work on coarse-to-fine decoding (Dong and Lapata, 2018; Cai and Lam, 2019), generating the core structures and then moving to the details.

Le and Zuidema (2012) were the first to train a data-driven DRT parser using a graph-based representation. Our model is trained on the GMB (Bos et al., 2017), a richly annotated resource in the style of DRT, which provides a unique opportunity for bootstrapping wide-coverage semantic parsers. Boxer (Bos, 2008) was a precursor to the GMB, the first semantic parser of this kind, which deterministically maps CCG derivations onto formal meaning representations. Le and Zuidema (2012) were the first to train a semantic parser on an early release of the GMB (2,000 documents, Basile et al. 2012), however, they abandon lambda calculus in favour of a graph-based representation. The latter is closely related to AMR, a general-purpose meaning representation language for broad-coverage text. In AMR the meaning of a sentence is represented as a rooted, directed, edge-labelled and leaf-labelled graph. AMRs do not resemble classical meaning representations and do not have a model-theoretic interpretation. However, see Bos (2016) and Artzi et al. (2015) for translations to first-order logic.

Various mechanisms have been proposed to improve sequence-to-sequence models

including copying (Gu et al., 2016) and attention (Vaswani et al., 2017). Our copying mechanism is more specialized and linguistically-motivated: it considers the semantics of the input text for deciding which tokens to copy. While our multi-attention mechanism is fairly general, it extracts features from different encoder representations (word- or sentence-level) and flexibly integrates supervised and unsupervised attention in a unified framework.

A few recent approaches focus on the alignment between semantic representations and input text, either as a preprocessing step (Foland and Martin, 2017; Damonte et al., 2017) or as a latent variable (Lyu and Titov, 2018). Instead, our parser implicitly models word-level alignments with multi-attention and explicitly obtains sentence-level alignments with supervised attention, aiming to jointly train a semantic parser.

Following our work, GAT (Fu et al., 2020) is the most-recent work on DRTS parsing, which adopts the same hierarchical decoder, but it uses the input syntactic features with graph attention network, achieving the 71.65 F_1 in document-level GMB benchmark.

3.7 Summary

In this chapter, we defined the discourse representation tree structure (DRTS) and presented a DRS-to-DRTS conversion algorithm to obtain DRTSs from discourse representation structures. Furthermore, we proposed a novel semantic parsing task to obtain Discourse Representation Tree Structures and introduced a general framework for parsing texts of arbitrary length and granularity. Experimental results on two benchmarks show that our parser is able to obtain reasonably accurate sentence- and document-level discourse representation structures (77.54 and 66.56 F_1, respectively).

The proposed DRTS parsing framework is equipped with a hierarchical encoder and a hierarchical decoder, where multiple attentions are used to extract hierarchical features (e.g. word- and sentence-level). Furthermore, supervised attention enriched the features (i.e. alignments between sentences and nodes in DRTSs) that can be used to guide our copy strategy. We also propose a constraint-based inference with a State Tracker to guarantee the well-formedness of document-level DRTSs. The inference procedure is portable to other structure prediction tasks by instantiating task-specific functions (i.e. INITALIZATION, UPDATE, ISTERMINATED and VALID), ensuring well-formed output.

Although the proposed DRTSs are taken to represent the text with arbitrary lengths,

they are simplified from DRS by removing presuppositions and scoped refers in the top layer of the original DRS boxes. Also, the proposed models are data-driven, which depends on the large-scaled annotated corpus. However, GMB data is constructed with crowdsourcing methods and lacks the gold-standard test dataset. The next chapter will introduce the full DRS parsing, which is evaluated on the Parallel Meaning Bank (PMB), having a gold-standard test dataset.

Chapter 4

Discourse Representation Structure Parsing

The last chapter introduces Discourse Representation Tree Structures (DRTS), which are simplified and induced from DRS boxes. However, DRTS only keeps the part of linguistic information of DRS without presuppositions (Karttunen, 1974) and word senses (Stevenson and Wilks, 2003). We regard DRTS parsing as an independent task, and in this chapter, we move to the full DRS parsing as a new task that keeps all linguistic information, and the parsers are evaluated in the gold-standard dataset.

A presupposition is an implicit assumption about the world or background belief relating to an utterance whose truth is taken for granted in discourse.[1] The presuppositions in DRSs are interpreted as anaphora resolution (Van der Sandt, 1992) or as projected pointers (Venhuizen et al., 2013). For example, the two sentences *a man buys the plant* and *the man buys the plant* have different meanings. The word *man* in the first sentence represents a general male person without semantic scopes, while the word *man* with the article *the* in the second sentence is specific to the man who is described somewhere (e.g., discourses and dialogues). In the DRS of the first sentence, the male person $(b_1 : male.n.01(x_1))$ is interpreted in the current box b_1, which is shown in Figure 4.1(a), while the male person $(b_3 : male.n.01(x_1))$ in DRS of the second sentence is interpreted in the other box b_3, which is shown in Figure 4.1(b), where the truth value for the description of *the man* has to be considered if we are trying to interpret the truth value of the logic form for the sentence *the man buys the apple*.

Word senses are used to identify the different meanings given the same words,

[1] https://en.wikipedia.org/wiki/Presupposition

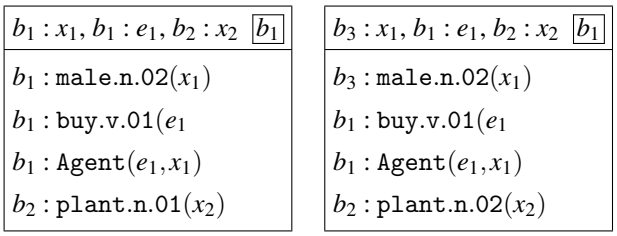


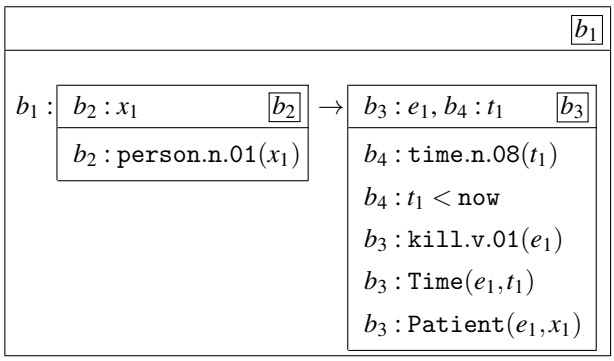
Figure 4.1: (a) DRS of the sentence *a man buys the plant*; (b) DRS of the sentence *the man buys the plant*

resulting in a problem of word sense disambiguation (WSD) (Stevenson and Wilks, 2003). The predicates in DRS corpora are assigned with word sense according to the WordNet (Fellbaum, 1998). For example, the word *plant* can be a living organism or a building for carrying on industrial labour. As shown in Figure 4.1, sentence *a man buys the plant* with the sense `plant.n.01` shows that a man buys the plant that is a kind of living organism, while the sentence *the man buys the plant* with the sense `plant.n.02` shows that the man buys a building.

In this chapter, we propose DRS parsing with neural models including the presupposition recovering and word sense disambiguation. However, GMB used in the last chapter has no gold-standard test dataset. We conduct experiments on Parallel Meaning Bank (PMB) that provides a large-scale text with DRS annotations, including the gold-standard test dataset with presuppositions and word senses. On the PMB benchmark, we investigate the DRS parsing in box, clause and tree format.

4.1 Computational Formats

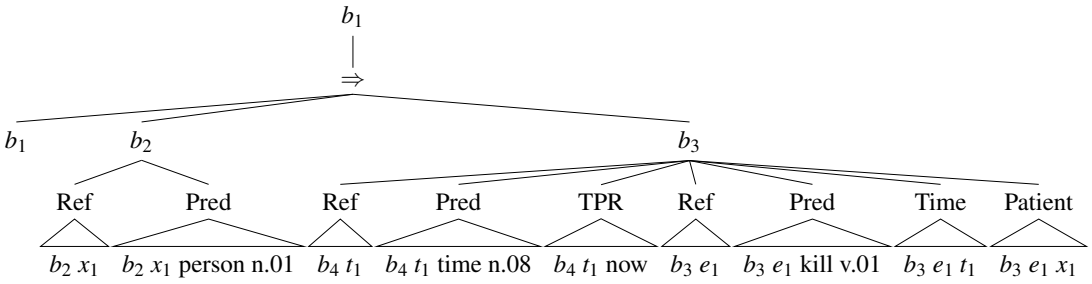
DRSs are displayed in a box-like format that is intuitive and easy to read but not convenient for modelling purposes. As a result, DRSs are often post-processed in a format that modern neural network models can straightforwardly handle (Liu et al., 2018a; van Noord et al., 2018b; Liu et al., 2019a). In this section, we provide an overview of the existing computational format, i.e. clause format, prior to describing our own proposed tree format without any of the simplifications made in DRTS.



(a)

b_1 IMP b_2 b_3
b_2 REF x_1
b_2 person "n.01" x_1
b_4 REF t_1
b_4 time "n.08" t_1
b_4 TPR t_1 "now"
b_3 REF e_1
b_3 kill "v.01" e_1
b_3 Time e_1 t_1
b_3 Patient e_1 x_1

(b)



(c)

Figure 4.2: Representation of DRS in (a) box format; (b) clause format (Abzianidze et al., 2017); (c) lossless tree format proposed in this chapter.

4.1.1 Clause Format

In the Parallel Meaning Bank (Abzianidze et al., 2017), DRS variables and conditions are converted to clauses. Specifically, variables in the top box layer are converted to clauses by introducing a special condition called "REF". Figure 4.2(b) presents the clause format of the DRS in Figure 4.2(a); here, "b_2 REF x_1" indicates that variable x_1

Algorithm 6 Box to Clause

Input: B, DRS in box format

Output: C, DRS in clause format

```
1: P = GETVARIABLEBOUND(B); V ← Ø; C ← Ø
2: procedure TRAVERSAL(b)
3:     if b is elementary DRS then TRAVERSALDRS(b) end if
4:     if b is segmented DRS then TRAVERSALSDRS(b) end if
5: end procedure
6: procedure TRAVERSALDRS(b)
7:     for cond in b.conds do                          ▷ each condition
8:         for v in cond.args do                       ▷ each argument
9:             if v not in V then C = C ∪ {(P[v] REF v)}; V = V ∪ {v} end if
10:        end for
11:        if cond is basic then
12:            C = C ∪ {(cond.bound cond.name cond.args)}
13:        else if cond is unary complex then
14:            C = C ∪ {(cond.bound cond.name cond.B)}
15:            TRAVERSAL(cond.B)
16:        else if cond is binary complex then
17:            C = C ∪ {(cond.bound cond.name cond.B1 cond.B2)}
18:            TRAVERSAL(cond.B1); TRAVERSAL(cond.B2)
19:        end if
20:    end for
21: end procedure
22: procedure TRAVERSALSDRS(b)
23:    for b′ in b.B do
24:        TRAVERSAL(b′)
25:    end for
26:    for rel in b.rels do
27:        C = C ∪ {(rel.bound rel.name rel.B1 rel.B2)}
28:    end for
29: end procedure
```

is bound in box b_2. Analogously, clause "b_3 `kill v.01` e_1" corresponds to condition `kill.v.01`(e_1) which is bound in box b_3 and b_4 TRP t_1 "now" is bound in box b_4 (TRP corresponds to temporal).[2] The mapping from boxes to clauses is not reversible; in other words, it is not straightforward to recover the original box from the clause format and restore the syntactic structure of the original sentence. For instance, the clause format discloses that temporal information is bound to box b_4, but not which

[2]For the full list of DRS clauses see https://pmb.let.rug.nl/drs.php.

box this information is located in (i.e., b_3 in Figure 4.2(a)). Although the clauses are annotated in PMB, Algorithm 6 converts the DRS boxes to the clauses which are equivalent to the clauses in PMB.

In the Algorithm 6, the function GETVARIABLEBOUND returns pairs of variables and box labels (indicating where these are bound) by enumerating all nested boxes.[3] The element $P[v]$ represents the label of the box bounding variable v. The function TRAVERSALDRS transforms elementary DRSs to clauses. Basic conditions are converted to a clause in lines 16–17, where *cond.args* is a list of the arguments of the condition (e.g., predicates and variables). Unary complex conditions (i.e., negation, possibility, and necessity) are converted to clauses in lines 18–20, while lines 21–23 show how to convert binary complex conditions (i.e., implication, disjunction, and duplication) to clauses.[4] The function TRAVERSALSDRS transforms segmented DRSs to clauses by transforming each nested box in lines 23–24 and producing clauses for discourse relations in lines 26 –27. An example is shown in Figure 4.2(b).

4.1.2 Tree Format

The DRS boxes are recursively constructed, and the boxes are nested in the another box, which is similar to the style of tree construction. We propose a box-to-tree conversion algorithm, which is reversible, i.e., it preserves all information present in the DRS box, including presupposition, word senses and the syntactic structure of the original text. Our conversion procedure is described in Algorithm 7. Similar to the box-to-clause algorithm, basic conditions are converted to a tree in lines 12–13, where *cond.args* is a list of the arguments of the condition (e.g., predicates and variables). Unary complex conditions (i.e., negation, possibility, and necessity) are converted to subtrees in lines 14–16, while lines 17–19 show how to convert binary complex conditions (i.e., implication, disjunction, and duplication) to subtrees. Each nested box in segmented DRS is converted to trees in lines 27–28 with their discourse relations in lines 30–31. An example is shown in Figure 4.2(c).

4.2 Models

Following Chapter 3 and previous work on semantic parsing (Dong and Lapata, 2016; Jia and Liang, 2016; Liu et al., 2018a; van Noord et al., 2018b), we adopt a neural

[3] Each variable has exactly one bounding box.

[4] Chapter 2 shows more detail on basic and complex conditions in DRS boxes.

Algorithm 7 Box to Lossless Tree

Input: B, DRS in box format

Output: T, DRS in tree format

$P = \text{GETVARIABLEBOUND}(B)$; $V \leftarrow \emptyset$; $T = \text{TRAVERSE}(B)$
procedure TRAVERSAL(b)
  if b is elementary DRS **then** return $\text{TRAVERSALDRS}(b)$ **end if**
  if b is segmented DRS **then** return $\text{TRAVERSALSDRS}(b)$ **end if**
end procedure
procedure TRAVERSEDRS(b)
  $t = \text{TREE}(b.name, [])$
  for *cond* **in** $b.conds$ **do** ▷ each condition
    for v **in** $cond.args$ **do** ▷ each argument
      if v **not in** V **then** $c = \text{TREE}(\text{REF}, [P[v], v])$; $\text{ADDCHILD}(t, c)$; $V = V \cup \{v\}$ **end if**
    end for
    if *cond* is basic **then**
      $c = \text{TREE}(cond.name, cond.bound, cond.args)$
    else if *cond* is unary complex **then**
      $c = \text{TREE}(cond.name, [])$
      $\text{ADDCHILD}(c, \text{TREE}(cond.bound, []))$; $\text{ADDCHILD}(c, \text{TRAVERSE}(c.B))$
    else if *cond* is binary complex **then**
      $c = \text{TREE}(cond.name, [])$; $\text{ADDCHILD}(c, \text{TREE}(cond.bound, []))$
      $\text{ADDCHILD}(c, \text{TRAVERSE}(c.B1))$; $\text{ADDCHILD}(c, \text{TRAVERSE}(c.B2))$
    end if
    $\text{ADDCHILD}(t, c)$
  end for
  return t
end procedure
procedure TRAVERSALSDRS(b)
  $t = \text{TREE}(b.name, [])$
  for b' **in** $b.B$ **do**
    $\text{ADDCHILD}(t, \text{TRAVERSE}(b'))$
  end for
  for *rel* **in** $b.rels$ **do**
    $c = \text{TREE}(rel.name, [rel.B1, rel.B2])$; $\text{ADDCHILD}(t, c)$
  end for
  return t
end procedure

sequence-to-sequence model which assumes that trees or clauses can be linearized into PTB-style bracketed sequences and sequences of symbols, respectively. Specifically, our encoder-decoder model builds on the Transformer architecture (Vaswani et al.,

2017), a highly efficient model which has achieved state-of-the-art performance in machine translation (Vaswani et al., 2017) and question answering (Yu et al., 2018a), and summarization (Liu et al., 2019b).

Our DRS parser takes a sequence of tokens, $s_1, s_2, ..., s_n$ as input and outputs their linearized DRS $t_1, t_2, ..., t_m$, where n is the number of input tokens, and m is the number of the symbols in the output DRS.

Encoder Each input token is represented by a vector x_k, which is the sum of word embeddings e_k and position embeddings p_k: $x_k = e_k + p_k$. The input representations, $x_1, x_2, ..., x_n$, are fed to the Transformer encoder to obtain their hidden representations, $h_1, h_2, ..., h_1$,:

$$[h_1 : h_n] = \text{LAYERNORM}(\text{ENCODER}([x_1 : x_n])), \tag{4.1}$$

where each layer of the ENCODER is:

$$\begin{aligned} [\bar{x}_1 : \bar{x}_n] &= \text{LAYERNORM}([x : x_n]), \\ [\bar{h}_1 : \bar{h}_n] &= \text{MULTIHEADSELFATTN}([\bar{x}_1 : \bar{x}_n]), \\ [h_1 : h_n] &= \text{FFN}([\bar{h}_1 : \bar{h}_n] + [x_1 : x_n]), \end{aligned} \tag{4.2}$$

where LAYERNORM is a layer normalization function (Ba et al., 2016); MULTIHEAD-SEFLATTN is the multi-head self-attention mechanism introduced in Vaswani et al. (2017) which allows the model to jointly attend to information from different representation subspaces (at different) positions; and FFN is a two-layer feed-forward network with the ReLU function.

Decoder The decoder uses the contextual representations of the encoder together with the embeddings ($t_{<k} = e_{<k} + p_{<k}$) of previously predicted tokens to output the next token t_k with the highest probability:

$$p(t_k|t_{<k}) = \text{SOFTMAX}(\text{LINEAR}(\bar{h}_k^d)), \tag{4.3}$$

where

$$\bar{h}_k^d = \text{LAYERNORM}(\text{DECODER}(t_{<k}, [h_1 : h_n])), \tag{4.4}$$

and each layer of the DECODER consists of five components:

$$\begin{aligned}
\bar{t}_{<k} &= \text{LAYERNORM}(t_{<k}), \\
q_{k-1} &= \text{MULTIHEADATTN}(\bar{t}_{k-1}, \bar{t}_{<k}), \\
\bar{q}_{k-1} &= \text{LAYERNORM}(q_{k-1} + t_{<k}), \\
c_{k-1} &= \text{MULTIHEADATTN}(\bar{q}_{k-1}, [x_0 : x_{n-1}]), \\
h_k^d &= \text{FFN}(c_{k-1} + q_{k-1} + t_{k-1}),
\end{aligned} \tag{4.5}$$

and $\text{MULTIHEADATTN}(t_{k-1}, t_{<k})$ returns the contextual representation for t_{k-1} according to its context information $t_{<k}$.

4.3 Training

Our models are trained with the standard back-propagation strategy, which requires a large-scale corpus with gold-standard annotations. However, annotating DRSs requires linguists with the knowledge of discourse representation theory, and it also has to deal with annotation agreement. Fortunately, non-gold-standard (a.k.a. auto-standard) DRSs can be easily and automatically obtained by the trained parser. In this section, we introduce the iterative training method that uses the auto-standard at various levels, and it is model-independent.

The auto-standard data, $\mathbb{D}_{auto} = D_1, D_2, ..., D_m$, is set of data with different qualities, where $D_i (1 \leq i \leq m)$ is the auto-standard data in the ith level, and D_1 has lowest quality and D_m has highest quality. The models are first trained on the collections, $\mathbb{D}_{auto}$ of all the data with different qualities, and then models are trained on the dataset, $\mathbb{D}_{auto}/D_1$ that excludes the data with the lowest quality D_1. The models trained on the D_m are our final model by iteratively removing the data with low quality. The details are shown in Algorithm 8.

4.4 Experiments

In this section, we describe the dataset used in our experiments, as well as details concerning the training and evaluation of our models.

Algorithm 8 The iterative training

Input: M, the model; $\mathbb{D}$, the auto-standard training data

Output: M_{opt}, the optimal model

1: $M_{opt} = M$
2: **for** i **in** $1...m$ **do**
3: $\quad M_{opt} = \text{TRAIN}(\mathbb{D})$
4: $\quad \mathbb{D} = \mathbb{D}/D_i$
5: **end for**

4.4.1 Settings

Data Our experiments were carried out on the Parallel Meaning Bank 2.2.0 (Abzianidze et al., 2017), which is annotated with DRSs in English. The dataset contains gold standard training/development/test data, and The PMB also provides silver and bronze standard training data for all languages. Silver data is only partially checked for correctness, while bronze data is not manually checked in any way. Both types of data were built using Boxer (Bos, 2008). We use bronze-, silver- and gold-standard training data as D_1, D_2, D_3, respectively, to the iterative training.

Hyper-parameters We used the pre-trained word embedding obtained on Wikipedia with GloVe (Pennington et al., 2014).[5] All models share the same hyperparameters. The dimension of the word embeddings is 300, the Transformer encoder and decoder have 6 layers with a hidden size of 300 and 6 heads; the dimension of position-wise feedforward networks is 4,096. The models were trained to minimize a cross-entropy loss objective with an l_2 regularization term. We used Adam (Kingma and Ba, 2014) as the learning rate optimizer; the initial learning rate was set to 0.001 with a 0.7 learning rate decay for every 4,000 updates starting after 30,000 updates. The batch size was 2,048 tokens.

Evaluation We evaluated the output of our semantic parser using COUNTER (van Noord et al., 2018a), a recently proposed metric suited for scoped meaning representations. COUNTER is taken on the DRSs in clause format and then computes precision and recall on matching clauses. Because the DRSs in tree format are reversible to the original boxes, they DRSs tree are easily converted into clauses via the boxes.

[5] https://nlp.stanford.edu/projects/glove/

4.4.2 Systems

We compared 11 models on the English portion of the PMB data:

- **Spar** is a baseline system that outputs the same DRS for each test instance.[6]
- **Sim-Spar** is a baseline system that outputs the DRS of the most similar sentence in the training set, based on a simple word embedding metric (van Noord et al., 2018b).
- **Boxer** is a system that outputs the DRSs of sentences according to their supertags and CCG derivations (Bos, 2015). Each word is assigned with lexical semantic representation according to its supertag category, and the semantic representation of a larger span is obtained by combining semantic representations of two continuous spans (or words). With the help of CCG derivation, the semantic representation of the whole sentence is achieved.
- **Graph** is a graph neural network to generate DRS by inducing directed acyclic graphs grammars (Fancellu et al., 2019). They extract a set of grammar rules from the training data, and the models learn how to apply the rules to obtain the DRSs.
- **Transition** is a neural transition-based model which incrementally generates the DRSs (Evang, 2019). They designed a set of transition actions that are applied in a stack-buffer framework. The stack contains the sequence of the generated partial DRS, and the buffer stores the income words. Transition actions are decided to consume a word in buffer or to merge the top two partial DRS to a new DRS. The system is terminated when all words are consumed, and only one DRS remains on the top of the stack.
- **Neural-Boxer** is a neural sequence-to-sequence model that outputs DRSs in clause format (van Noord et al., 2018b), which is the similar to our models but with LSTM.
- **MultiEnc** is a sequence-to-sequence model with multiple encoders that outputs DRSs in clause format (van Noord et al., 2019), which extend **Neural-Boxer** by adding various encoders to encode different lexical information, e.g., part-of-speech and syntax.
- **Cls/Tree-Transformer** is the Transformer model from Section 4.2; it outputs DRSs in clause format and tree format using the box-to-tree conversion algorithm introduced in Section 4.1.2. We also reimplement the LSTM models outputing DRSs in clause format and trees format, which are named as **Cls/Tree-LSTM**.

[6] In PMB (release 2.2.0) this is the DRS for the sentence "*Tom voted for himself.*"

DRS	Prec	Rec	F_1
SPAR	44.4	37.8	40.8
SIM-SPAR	57.0	58.4	57.7
BOXER	72.1	72.3	72.2
Transition	75.6	74.6	75.1
Graph	–	–	76.4
Neural-BOXER	85.0	81.4	83.2
MultiEnc	87.6	86.3	87.0
Cls-LSTM	82.5	83.3	83.9
Tree-LSTM	84.3	84.7	84.3
Cls-Transformer	88.1	87.7	87.9
Tree-Transformer	**88.6**	**88.9**	**88.7**
w/o bronze	86.1	86.0	86.0 (-2.6)
w/o bronze & silver	79.9	84.4	82.1 (-6.5)

Table 4.1: English DRS parsing results (PMB 2.2.0 test set), where the results for SPAR, SIM-SPAR, BOXER, Transition, Graph, Neural-BOXER and MultiEnc are taken from respective papers; best result per metric shown in bold.

4.4.3 Results

We present results on DRS parsing in order to assess the performance of our model on its own and whether differences in the format of the DRS representations make any difference. Having settled the question of which format to use.

Table 4.1 summarizes our results on the DRS parsing. As can be seen, neural models overwhelmingly outperform comparison baselines. Transformers trained on trees and clauses perform better (by 4.0 F_1 and 4.4 F_1, respectively) than LSTMs trained on the data in the same formats. A Transformer trained on trees performs slightly better (by 0.8 F_1) than the same model trained on clauses and is overall best among models employing DRS-based representations.

4.4.4 Analysis

Component types We further analyzed the parsers' output in order to determine which components of the semantic representation are modeled best. COUNTER (van

	LSTM		Transformer	
	Cls	Tree	Cls	Tree
DRS operator	90.80	91.02	94.05	**95.07**
Role	81.68	83.23	87.87	**88.48**
Concept	81.41	82.91	85.97	**86.87**
Syns-Noun	87.41	87.79	91.57	**91.86**
Syns-Verb	65.27	66.04	71.58	**74.20**
Syns-Adjective	71.74	73.12	74.73	**76.93**
Syns-Adverb	**66.48**	60.00	60.00	54.55

Table 4.2: Fine-grained evaluation (F_1) on the English PMB test set by Cls/Tree-Transformer and Cls/Tree-LSTM. The highest scores are bold.

Noord et al., 2018a) provides detailed break-down scores for DRS operators (e.g., negation), Roles (e.g., Agent), Concepts (i.e., predicates), and Synsets (e.g., "n.01"). Table 4.2 compares the output of our English semantic parsers. Tree models perform better than Clauses on most components except for adverb synsets. All models are better at predicting noun senses compared to verbs, adjectives, and adverbs. The clause format is better when it comes to predicting the senses of adverbs. Nevertheless, all models perform poorly on adverbs which are relatively rare in the PMB. The presuppositions are associated to variables and conditions, so we cannot directly compute the scores independently. In order to investigate the presupposition performances, we remove and replace the presuppositions pointers with their default box labels for all variables and conditions in the outputs of the parsers. Then we compute the F_1 scores between the output DRSs without presupposition and the gold DRSs with presuppositions. Table 4.3 shows the performances on the presupposition prediction. The LSTM and Transformer models have large losses (nearly -19.0 F_1) without presupposition predictions. So our parsers have 19.0 F_1 contributions of presupposition predictions to the DRSs parsing.

Iterative Training Figure 4.3 shows the prediction accuracy during the training with bronze-(D_1), silver-(D_2) and gold-standard(D_3) training data. The black dotted curve shows the accuracy of the model trained on the bronze-, silver- and gold-standard data ($D_1 + D_2 + D_3$), the red dashed curves show the accuracy of the model trained on the

	w/ presup	w/o presup	loss
Tree-LSTM	84.3	65.0	-19.3
Tree-Transformer	88.7	69.1	-19.6

Table 4.3: The performances on presupposition prediction (F_1) on the English PMB test set by Tree-Transformer and Tree-LSTM. Noted that DRSs in clause format ignore the default box label and only keep the presuppositions, so we cannot measure the performance without presupposition by replacing presupposition with default box labels.

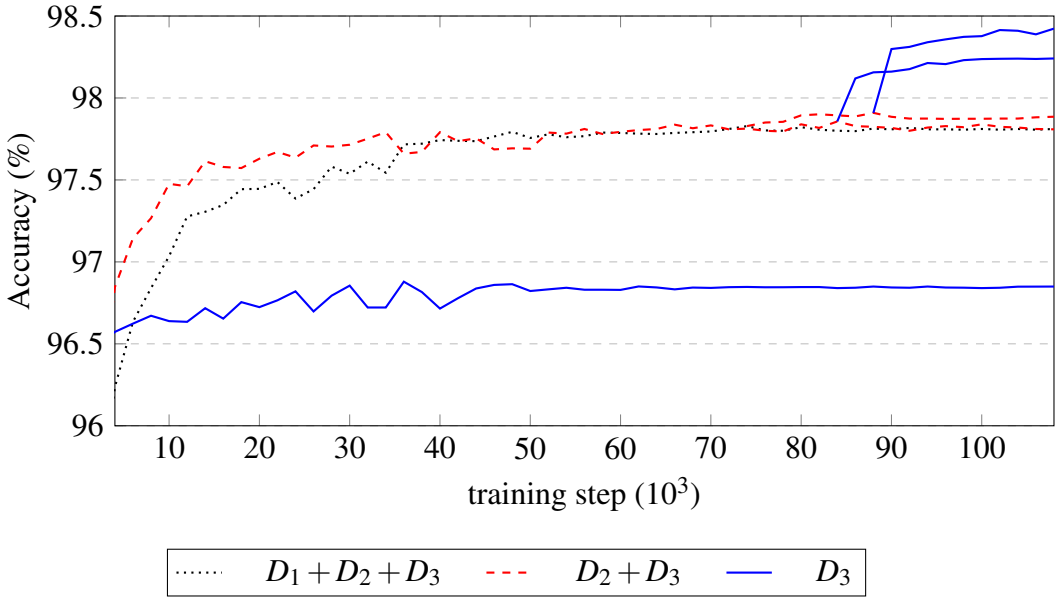


Figure 4.3: Accuracy on development dataset with the iterative training by Tree-Transformer. Blue curves show the training on gold-standard data, red dashed curves show the training on the mix of silver- and gold-standard data, and black dotted curves show the training on the mix of bronze-, silver- and gold-standard data.

silver- and gold-standard data ($D_2 + D_3$), and the blue curves show the accuracy of the model trained on the gold-standard data (D_3). There is an improvement gap between the model trained on $D_1 + D_2 + D_3$ and the model trained only on D_3. The model is quickly converged into local optimisation only trained on the small-scale gold-standard data (D_3). Also, after the convergence on $D_1 + D_2 + D_3$, the models can be further enhanced when they are continuously trained on data $D_2 + D_3$ that has higher quality. As shown in the figure, there is a big jump when the model is continually trained on

gold-standard data (D_3), which helps improve the performance of the model further. One reason is that the small gold-standard data has high quality but low coverage, and the parameter optimization on the set of bronze-, silver- and gold-standard data makes the model to be broad-coverage, and then fine-grained optimization on high-quality data makes the model to be more accurate. If the models are trained on silver- and gold-standard data ($D_2 + D_3$) without bronze-standard data (D_1) and then finetuned on gold-standard data (D_3), the performance is slightly worse then that using bronze-standard data at the beginning of the training.

4.5 Related Work

Recent years have seen growing interest in the development of DRT parsing models. Early seminal work (Bos, 2008) created Boxer, an open-domain semantic parser that produces DRS representations (in box format) by capitalizing on the syntactic analysis provided by a robust CCG parser (Curran et al., 2007). Boxer has been instrumental in enabling the creation of the Groningen Meaning Bank (Bos et al., 2017) and the Parallel Meaning Bank (Abzianidze et al., 2017). Following the available DRS banks, van Noord et al. (2018b) adapt sequence-to-sequence models with LSTM units to parse DRSs in clause format, also following a graph-based representation. Fancellu et al. (2019) represent DRSs as direct acyclic graphs and design a DRT parser with an encoder-decoder architecture that takes as input a sentence and outputs a graph using a graph-to-string rewriting system. In addition, their parser exploits various linguistically motivated features based on part-of-speech embeddings, lemmatization, dependency labels, and semantic tags (van Noord et al., 2019). We propose the alternative tree format for DRS, and adopt the transformer to construct DRS parser, which achieves the state-of-the-art results.

4.6 Summary

In this chapter, we investigated the DRS parsing, including presupposition recover and word sense disambiguation. For the modelling purposes, we introduced two kinds of DRS formats, i.e., clause and tree and design an encoder-decoder parser to this end, showing that the transformer with the DRS in tree format performs slightly better than that in clause format. Also, the iterative training provides a way to adopt the auto-standard training data in various quality, which significantly improve the DRS parser.

The resulting DRS parser with transformers achieves state-of-the-art performance on the PMB 2.2.0 benchmarks.

Until this chapter, we only focus on English semantic parsing. However, there are many other widely-used languages in the world. Next chapter, we move to investigate semantic parsing in non-English languages, answering the research question of whether the DRS parsing can work for non-English texts.

Chapter 5

Universal Discourse Representation Structure

The last chapter introduces semantic parsing in English. Although English is the most widely used language in the world, the number of native English speakers is only about 4.74% of the world population.[1] It is necessary to explore text understanding in multiple languages. However, low-resource languages lack large-scale corpora with gold annotation, which is key to the data-hungry neural models.

There have been many works focusing on the multilingual semantic parsing by constructing corpora for non-English languages, such as multilingual FrameNet annotation (Fung and Chen, 2004; Padó and Lapata, 2005), Universal Conceptual Cognitive Annotation (UCCA; Abend and Rappoport 2013b), Abstract Meaning Representation (AMR; Banarescu et al. 2013; Li et al. 2016a), Universal Decompositional Semantic (UDS; White et al. 2016), and Parallel Meaning Bank (PMB; Abzianidze et al. 2017) which contains annotations for English, German, Dutch, and Italian sentences based on Discourse Representation Theory (DRT; Kamp and Reyle 1993). Renewed interest[2] in DRS parsing has been triggered by the realization that document-level semantic analysis is prerequisite to various applications ranging from machine translation (Kim et al., 2019) to machine reading (Gangemi et al., 2017; Chen, 2018), and generation (Basile and Bos, 2013; Narayan and Gardent, 2014).

Figure 5.1(a) shows the DRS corresponding to an example sentence taken from the PMB, and its Italian translation. The traditional DRSs are depicted as boxes. Each box comes with a unique label (see b_1, b_2, b_3 in the figure) and has two layers. The

[1] https://en.wikipedia.org/wiki/List_of_languages_by_total_number_of_speakers

[2] Details on the IWCS 2019 shared task on Discourse Representation Structure parsing can be found at https://sites.google.com/view/iwcs2019/home.

English: everyone was killed .
Italian: sono stati uccisi tutti .
they were killed all .

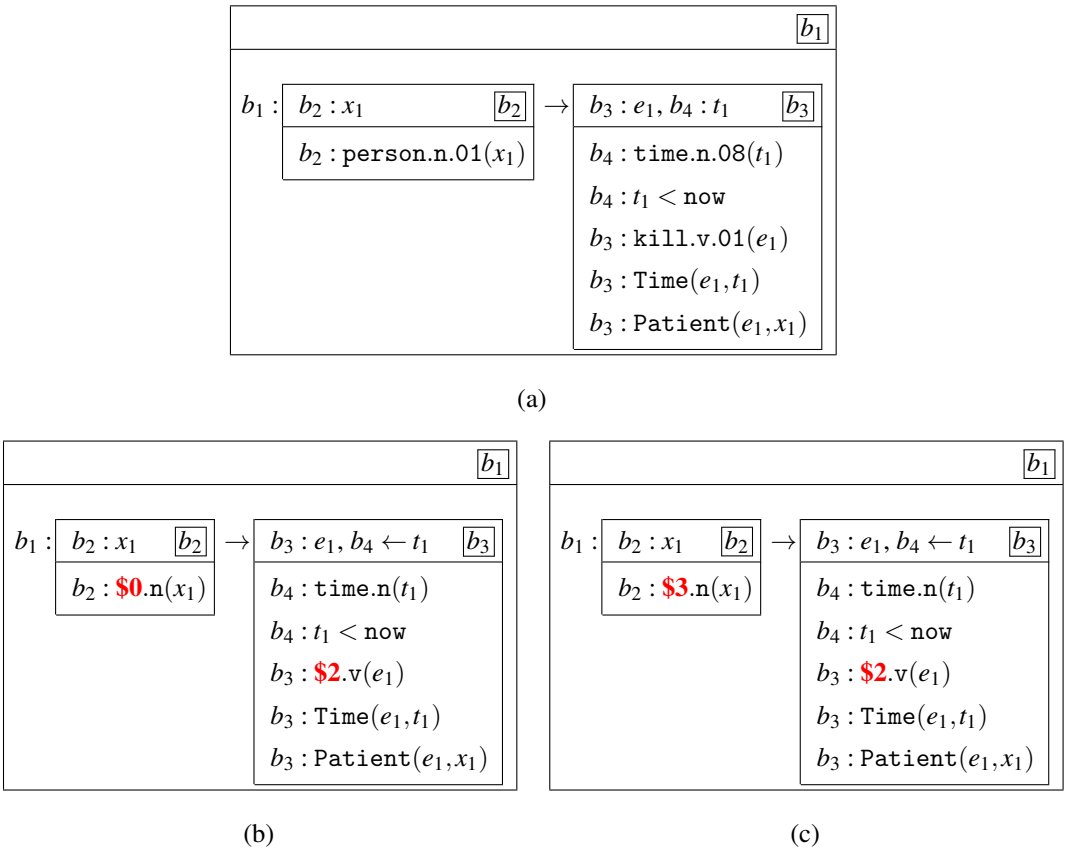


Figure 5.1: (a) DRS of English sentence "*everyone was killed*" and Italian translation "*sono stati uccisi tutti*", taken from the Parallel Meaning Bank; (b) $\mathbb{U}$DRS for English and (c) Italian sentence.

top layer contains discourse referents (e.g., x_1, t_1), while the bottom layer contains conditions over discourse referents. Each referent or condition belongs to a unique box label, showing the referent or the condition is interpreted in that box (e.g., $b_2 : x_1$ and b_2 : person.n.01(x_1)). In PMB, predicates are disambiguated with senses (e.g., n.01 and v.01) provided in WordNet (Fellbaum, 1998). We provided more details on the DRT formalism in Chapter 2.

Despite efforts to create multilingual DRS annotations (Abzianidze et al., 2017), the amount of gold-standard data for languages other than English is limited to a few hundred sentences that are useful for evaluation but not for model training. The creation of such data remains an expensive endeavour requiring expert knowledge, e.g.,

familiarity with the semantic formalism and language at hand. Since it is unrealistic to expect that semantic resources will be developed for many low-resource languages shortly, previous work has resorted to machine translation and bitexts which are more readily available (Damonte and Cohen, 2018; Evang and Bos, 2016; Zhang et al., 2018). **Crosslingual** semantic parsing leverages an existing parser in a *source* language (e.g., English) together with a machine translation system to learn a semantic parser for a *target* language. This chapter aims to develop a cross-lingual DRS parser for languages where no gold-standard training data is available.

Like other related broad-coverage semantic representations (e.g., AMR), traditional DRSs are not directly anchored in the sentences whose meaning they purport to represent. However, PMB provides the linking from the semantic representations to the sentence tokens via CCG derivations. We propose a variant of the DRS formalism, which explicitly anchors semantic representations to words. Specifically, we introduce 𝕌niversal Discourse Representation Structures (𝕌DRSs) where language-dependent symbols are replaced with anchors referring to tokens (or characters, e.g., in the case of Chinese) of the input sentence. 𝕌DRSs adopt the alignment to make DRSs suitable for cross-lingual parsing that relies on semantic and structural equivalences between languages. 𝕌DRSs omit lexical details pertaining to the input sentence and as such are able to capture similarities in the representation of expressions *within* the same language and *across* languages. As shown in Figure 5.1(b) and Figure 5.1(c), "`person`" and "`kill`" are replaced with anchors \$0 and \$2, corresponding to English tokens *everyone* and *killed*, and \$3 and \$2, corresponding to Italian tokens *tutti* and *uccisi*. Also notice that 𝕌DRSs omit information about word senses (denoted by WordNet synsets, e.g., `person.n.01`, `time.n.08` in Figure 5.1(a)) as the latter cannot be assumed to be the same across languages (see Section 5.1 for further discussion).

Our cross-lingual parser takes advantage of 𝕌DRSs and state-of-the-art machine translation to develop semantic resources in multiple languages following two learning schemes. The **Many-to-One** approach works by translating non-English text to English, and then running a relatively accurate English DRS parser on the translated text, while the **One-to-Many** approach translates gold-standard English (training data) to non-English text and trains multiple parsers (one per language) on the translations. In this chapter, we propose (1) 𝕌DRSs to explicitly anchor DRSs to lexical tokens, which we argue is advantageous for cross-lingual parsing; (2) a general cross-lingual semantic parsing framework based on the Transformer architecture and its evaluation on the PMB following the one-to-many and many-to-one learning paradigms; (3) a

large-scale corpus that contains (silver-standard) discourse representation annotations in 102 languages.

5.1 𝕌niversal DRS

How are DRSs used to represent meaning across languages? An obvious idea would be translating the English text to the non-English languages of interest, and the translated non-English languages and English sentence share the same DRSs written with English word senses. For example, English sentence *everyone was killed* and Italian sentence *sono stati uccisi tutti* share the same DRS, which is shown in Figure 5.1(a). However, we cannot simply assume that sense distinctions are preserved across languages.[3] For example, the verb *eat/essen* has six senses in the English WordNet (Fellbaum, 1998) but only one in GermaNet (Hamp and Feldweg, 1997), and the word *good/好* has 23 senses in the English WordNet but 17 senses in Chinese WordNet (Huang et al., 2010). Since word sense disambiguation is language-specific, we relax cross-lingual DRS parsing by excluding the word sense disambiguation. With the assumption that DRS operator (e.g., negation scoped) and semantic roles are consistent across the languages, we propose 𝕌niversal Discourse Representation Structures (𝕌DRSs) by replacing "DRS tokens" such as constants and predicates of conditions, with *alignments* to tokens or spans (e.g., named entities) in the input sentence. An example is shown in Figure 5.2(b), where condition $b_3 : \texttt{kill.v.01}(e_1)$ is generalized to $b_3 : \$2.\texttt{v}(e_1)$.

𝕌DRS representations abstract semantic structures within the same language and across languages. Monolingually, they are generalizations of sentences with different semantic content but similar syntax. As shown in Figure 5.2, sentences "*Tom is eating an apple .*" and "*Jack is cleaning a car .*" are represented by the same 𝕌DRS which can be viewed as a template describing an event in past tense with an agent and a theme. 𝕌DRSs are more compact representations and advantageous from a modelling perspective; they are easier to generate compared to DRSs since multiple training instances are represented by the same semantic structure. Moreover, 𝕌DRSs can be used to capture basic meaning across languages. The 𝕌DRS in Figure 5.2(c) can be used to recover the semantics of the sentence "汤姆 正在 吃 一个 苹果" by substituting index \$0 with 汤姆, index \$2 with 吃, and index \$4 with 苹果 (see Figure 5.2(d)).

[3] http://globalwordnet.org/about-gwa/

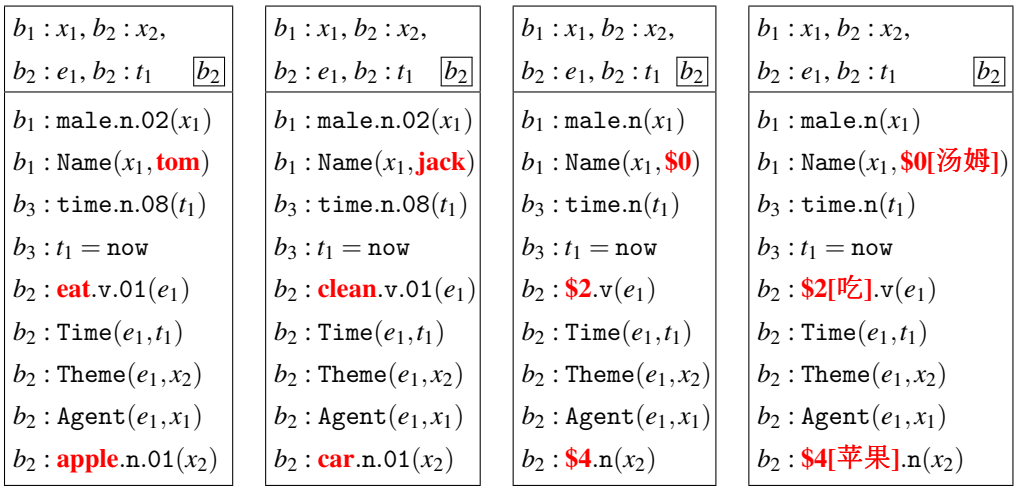


Figure 5.2: (a) DRS of sentence *Tom is eating an apple*; (b) DRS of sentence *Jack is cleaning a car*; (c) 𝕌DRS of both sentences; (d) 𝕌DRS of sentence 汤姆 正在 吃 一个 苹果 constructed via (c) by substituting indices with corresponding words.

5.1.1 Link to Knowledge Bases

An essential distinction between 𝕌DRSs and DRSs is that the former do not represent word senses. Word sense disambiguation (WSD) requires the definitions of the senses in hand, where English WSD is usually taken on English WordNet. However, as discussed earlier, the senses are different across languages in the language-specific knowledge bases. So we view the word sense disambiguation as a post-processing step which can enrich 𝕌DRSs with more fine-grained semantic information according to specific tasks and knowledge resources, such as WordNet in multiple languages[4], BabelNet (Navigli and Ponzetto, 2010), ConceptNet (Speer et al., 2017) and HowNet (Dong et al., 2010). Based on the semantic relations (e.g., synonyms, hyponyms, and meronyms) and the common senses in these knowledge bases, the sentences with their 𝕌DRSs can be interpreted. One example is shown in Figure 5.3.

5.1.2 Link to Language Models

𝕌DRSs keep predicates and constants with their original forms (see 吃.v(e_1) in Figure 5.2 (d)), and by directly linking them to the words in sentences, 𝕌DRSs represent multiword expressions as a combination of multiple tokens aiming to assign atomic

[4] http://globalwordnet.org/resources/wordnets-in-the-world/

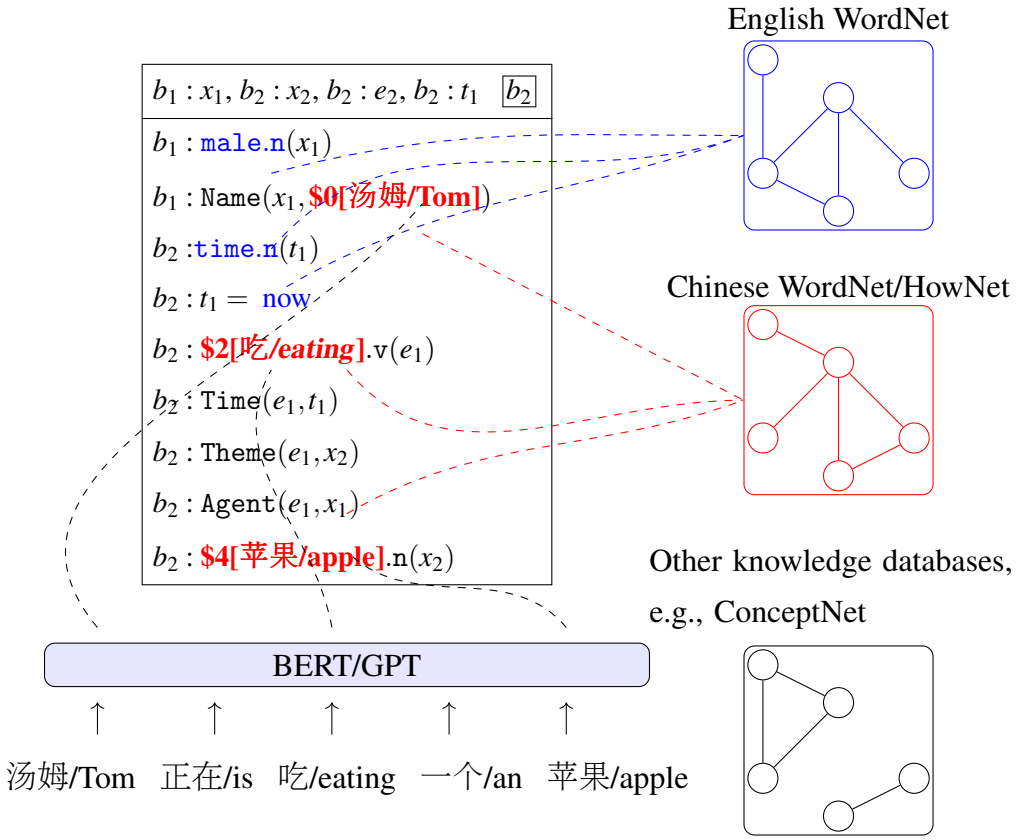


Figure 5.3: The advantages of 𝕌DRSs, where the logic forms in DRS together with the grounded words can enrich the deep contextual word representations, and word senses can be disambiguated according the definitions of various language-specific knowledge graphs.

meanings and avoid redundant lexical semantics. For example, in sentence *Tom picked the graphic card up*, the words *graph card* corresponds to entity \$3-\$4.n(x_2) and the words *picked up* to relation \$1-\$5.v(x_1, x_2). The links make 𝕌DRSs to be portable to large-scale pretrained models, e.g. Elmo (Peters et al., 2018), BERT (Devlin et al., 2019) and GPT (Radford et al., 2019). As shown in Figure 5.3, 𝕌DRSs are capable to bridge deep contextual representations and rich semantic symbols, motivating future researches, such as the interpretation and the probing on pretrained models (Hewitt and Manning, 2019; Kulmizev et al., 2020), and the pretrained models enhancement (Wu and He, 2019; Hardalov et al., 2020; Kuncoro et al., 2020).

5.1.3 Comparison to DRS and DRTS

Discourse Representation Tree Structures (DRTSs) are constructed in tree format that are derived from the original DRS boxes. Although DRS boxes can be transformed into tree formats, they are slightly different from DRTS. The original DRSs/𝕌DRSs are described in box format that can be transformed into clauses and trees for modelling purposes. The only difference between DRSs and 𝕌DRSs is that 𝕌DRSs directly ground the predicates and entities to the corresponding words, while DRSs abstract meaning representation symbols away from the words and adopt word senses in English Wordnet.

5.2 Crosslingual Semantic Parser

As mentioned earlier, PMB (Abzianidze et al., 2017) contains a small number of gold standard DRS annotations in German, Italian, and Dutch. Multilingual DRSs in PMB use English WordNet synsets regardless of the source language. The output of our cross-lingual semantic parser is compatible with this assumption which is also common in related broad-coverage semantic formalisms (Damonte and Cohen, 2018; Zhang et al., 2018). In the following, we present two learning schemes (illustrated schematically in Figure 5.4) for bootstrapping DRT semantic parsers for languages lacking gold standard training data.

5.2.1 Many-to-One Method

According to the Many-to-One approach, target sentences (e.g., in German) are translated to source sentences (e.g., in English) via a machine translation system. Then a relatively accurate source DRS parser (trained on gold-standard data) is adopted to map the target translations to their semantic representation. Figure 5.4(a) provides an example for the three PMB languages.

An advantage of this method is that labelled training data in the target language is not required. However, the performance of the semantic parser on the target languages is limited by the performance of the semantic parser in the source language and machine translation system; moreover, the cross-lingual parser must be interfaced with a machine translation system at run-time, since it only accepts input in English.

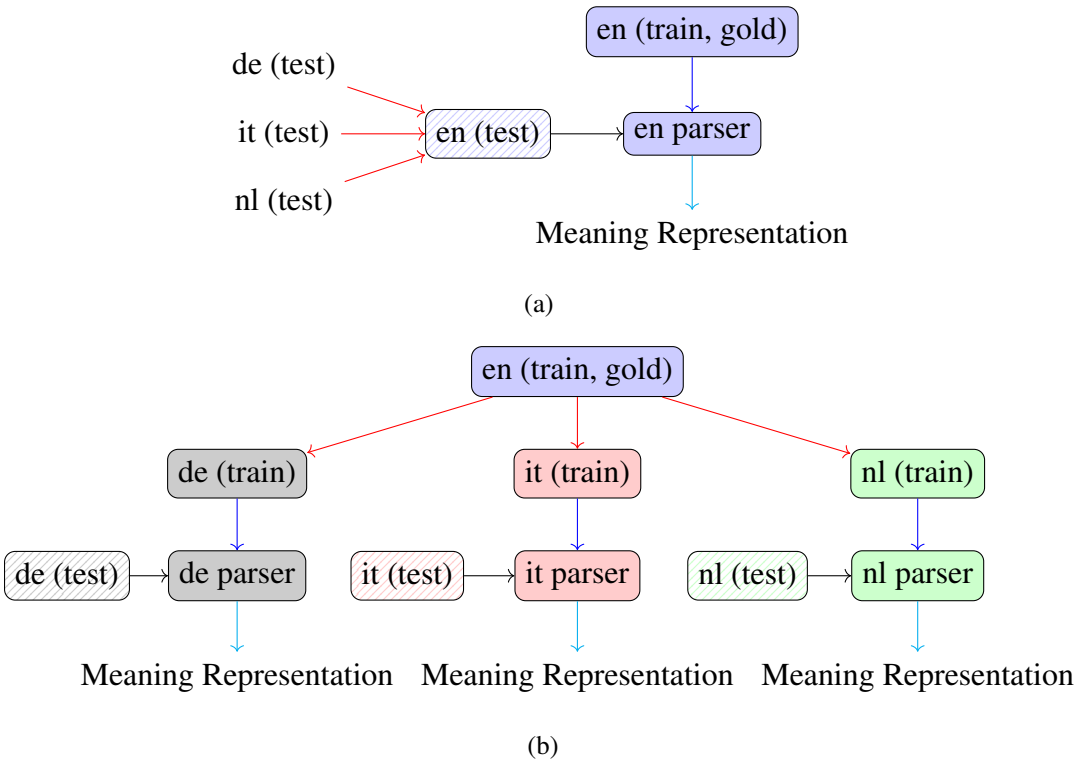


Figure 5.4: Two approaches for learning cross-lingual DRT parsers. Red arrows denote a machine translation engine; blue arrows denote the training of semantic parsing model, and cyan arrows denote the application of trained parser to test data (drawn in dotted background).

5.2.2 One-to-Many Method

The One-to-Many approach constructs training data for the target languages (see in Figure 5.4(b)) via machine translation. The translated sentences are paired with gold DRSs (from the source language) and collected as training data for the target languages. An advantage of this method is that the obtained semantic parsers are sensitive to the linguistic aspects of individual languages (and how these correspond to meaning representations). From a modeling perspective, it is also possible to exploit large amounts of unlabeled data in the target language to improve the performance of the semantic parser. Also notice that the parser is independent of the machine translation engine employed (sentences need to be translated only once) and the semantic parser developed for the source language. In theory, different parsing models can be used to cater for language-specific properties.

The learning schemes just described are fairly general and compatible with either clause or tree DRS formats, or indeed meaning representation schemes which are not

based on DRT. However, the proposed 𝕌DRS representation heavily depends on the order of the tokens in the natural language sentences, and as a result is less suited to the Many-to-One method; parallel sentences in different languages might have different word orders and consequently different 𝕌DRSs (recall that the latter are obtained via aligning non-English tokens to English ones).

5.2.3 Semantic Parsing Model

Like DRSs, 𝕌DRSs can be formated as clauses and trees in similar ways in Chapter 4. We adopt the same neural sequence-to-sequence models with transformers from Section 4.2 as our semantic parsing models.

5.3 Experiments

In this section, we describe the dataset used in our experiments, as well as details concerning the training and evaluation of our models.

5.3.1 Settings

Data Our experiments were carried out on the Parallel Meaning Bank 2.2.0 (Abzianidze et al., 2017), which is annotated with DRSs for English (en), German (de), Italian (it) and Dutch (nl). The dataset contains gold standard training data for English only, while development and test gold standard data is available for all four languages. The PMB also provides silver and bronze standard training data for all languages. Silver data is only partially checked for correctness, while bronze data is not manually checked in any way. Both types of data were built using Boxer (Bos, 2008) with annotation projection pipeline (Abzianidze et al., 2017).

Training We used pre-trained word embeddings obtained on Wikipedia with GloVe (Pennington et al., 2014) for the German, Italian, and Dutch languages.[5] All models share the same hyperparameters. The dimension of the word embeddings is 300, the Transformer encoder and decoder have 6 layers with a hidden size of 300 and 6 heads; the dimension of position-wise feedforward networks is 4,096. The models were trained to minimize a cross-entropy loss objective with an l_2 regularization term.

[5]German https://deepset.ai/german-word-embeddings; Italian http://hlt.isti.cnr.it/wordembeddings/; and Dutch https://github.com/clips/dutchembeddings.

	de			it			nl			all
DRS	Pre	Rec	F_1	Pre	Rec	F_1	Pre	Rec	F_1	avg F_1
Cls	72.1	72.6	72.3	74.2	74.4	74.3	64.3	65.2	64.8	70.5
Cls-m2o	84.5	83.6	84.0	85.1	85.4	85.2	**84.4**	84.0	84.2	84.5
Cls-o2m	81.1	80.2	80.6	81.0	80.6	80.8	76.0	76.4	76.2	79.2
Tree	72.6	72.9	72.8	75.4	75.9	75.7	65.8	66.22	66.0	71.5
Tree-m2o	**84.6**	**83.9**	**84.2**	**86.0**	**85.6**	**85.9**	84.3	**84.4**	**84.4**	**84.9**
Tree-o2m	82.7	81.1	81.9	81.4	81.0	81.2	78.4	77.5	78.0	80.3

Table 5.1: DRS parsing results on German, Italian, and Dutch (PMB test set); best result per metric shown in bold.

We used Adam (Kingma and Ba, 2014) as the learning rate optimizer; the initial learning rate was set to 0.001 with a 0.7 learning rate decay for every 4,000 updates starting after 30,000 updates. The batch size was 2,048 tokens. We adopt iterative training in Chapter 4.

Evaluation We evaluated the output of our semantic parser using COUNTER (van Noord et al., 2018a), a recently proposed metric suited for scoped meaning representations. COUNTER computes F-score on DRSs in clause formats. All the outputs of the models are transformed into clauses with the method introduced in Chapter 4.

5.3.2 Systems

Our cross-lingual experiments were carried out on German, Italian, and Dutch. We built four cross-lingual Transformer-based models:

- **Cls/Tree-m2o** uses the Many-to-One method to translate foreign sentences into English and parse them using an English Transformer trained on clauses or trees.
- **Cls/Tree-o2m** applies the One-to-Many method to construct training data in the target languages for training clause and tree Transformer models.

5.3.3 Results

DRS Parsing Our results on the DRS cross-lingual setting are summarized in Table 5.1. Many-to-One parsers outperform One-to-Many ones, however, the difference

	de			it			nl			all
𝕌DRS	Pre	Rec	F_1	Pre	Rec	F_1	Pre	Rec	F_1	avg F_1
Cls	83.0	82.4	82.7	85.2	85.3	85.2	74.9	75.9	75.4	81.1
Cls-o2m	89.0	88.7	88.8	**88.4**	87.9	88.2	86.1	84.6	85.3	87.4
Tree	83.1	82.8	83.0	85.1	85.3	85.2	75.6	76.5	76.1	81.4
Tree-o2m	**89.5**	**88.7**	**89.1**	89.2	**88.2**	**88.7**	**85.7**	**84.8**	**85.2**	**87.7**

Table 5.2: 𝕌DRS parsing results on German, Italian, and Dutch (PMB test set); best result per metric shown in bold.

is starker for Clauses than for Trees (5.3 vs. 4.6 F_1 points). With the Many-to-One strategy, Tree-based representations are overall slightly better on DRS parsing.

𝕌DRS Parsing Our results on the 𝕌DRS are shown in Table 5.2. Aside from 𝕌DRS parsers trained with the One-to-Many strategy (Cls-o2m and Tree-o2m), we also report the performance of monolingual Transformers (clause and tree formats) trained on the silver and bronze standard datasets provided in PMB. All models were evaluated on the *gold standard* PMB test data. We only report the performance of One-to-Many 𝕌DRS parsers due to the word order issue discussed in Section 5.2.2. Compared to models trained on silver and bronze data, the one-to-many strategy improves performance for both clause (Cls vs Cls-o2m) and tree (Tree vs Tree-o2m) formats (by 5.7 F_1 and 5.7 F_1, on average). Overall, the cross-lingual experiments show that we can indeed bootstrap fairly accurate semantic parsers across languages, without *any* manual annotations on the target language.

5.3.4 Analysis

We further analyzed the parsers' output in order to determine which components of the semantic representation are modeled best. COUNTER (van Noord et al., 2018a) provides detailed break-down scores for DRS operators (e.g., negation), Roles (e.g., Agent), Concepts (i.e., predicates), and Synsets (e.g., "n.01").

In our cross-lingual experiments, we observe that Tree models slightly outperform Clauses across languages. For the sake of brevity, Table 5.3 only reports a break-down of the results for Trees. Interestingly, we see that the bootstrapping strategies proposed here are a better alternative to just training semantic parsers on PMB's silver

	DRS			𝕌DRS	
	Tree	Tree-m2o	Tree-o2m	Tree	Tree-o2m
de					
DRS operator	84.89	**91.75**	90.94	86.26	**91.87**
Role	72.47	**84.37**	82.62	73.93	**83.83**
Concept	69.08	**81.69**	78.21	—	—
Syns-Npun	79.67	**88.71**	86.19	—	—
Syns-Verb	41.38	**63.69**	58.82	—	—
Syns-Adjective	50.32	**69.54**	59.64	—	—
Syns-Adverb	14.29	0.00	**25.00**	—	—
it					
DRS operator	87.28	**91.20**	89.54	87.82	**91.20**
Role	76.18	**88.60**	82.02	79.41	**84.82**
Concept	71.35	**81.63**	77.60	—	—
Syns-Noun	81.71	**87.69**	86.70	—	—
Syns-Verb	44.48	**66.13**	53.91	—	—
Syns-Adjective	52.35	**70.16**	62.11	—	—
Syns-Adverb	0.00	0.00	0.00	—	—
nl					
DRS operator	79.90	**92.98**	89.05	82.69	**93.58**
Role	64.08	**83.54**	77.72	65.57	**77.75**
Concept	63.34	**82.43**	74.63	—	—
Syns-Noun	74.66	**88.59**	82.68	—	—
Syns-Verb	32.33	**67.26**	54.81	—	—
Syns-Adjective	41.38	**64.86**	50.00	—	—
Syns-Adverb	0.00	0.00	**50.00**	—	—

Table 5.3: Fine-grained evaluation (F_1%) on German, Italian, and Dutch (test set); best result per synset shown in bold.

and bronze data (see Tree column in Table 5.3). Moreover, the success of bootstrapping strategy seems to be consistent among languages, with Many-to-One being over-

whelmingly better than One-to-Many. However, Many-to-One fails to predict adverb synsets. Overall, the prediction of synsets is a harder task and indeed performance improves when the model only focuses on operators and semantic roles (compare Tree and Tree-o2m columns in DRS and 𝕌DRS).

5.3.5 Scalability Experiments

We further assessed the scalability of the cross-lingual approach advocated in this paper by obtaining 𝕌DRS parsers for all languages supported by Google Translate in addition to German, Italian, and Dutch (99 in total). Specifically, we applied the One-to-Many bootstrapping method on the English gold-standard PMB annotations to obtain semantic parsers for 99 additional languages. These parsers were trained and tested on *silver-standard* data.

As described in Section 5.2.2 we translated English sentences to the language of interest and paired the translations with the 𝕌DRSs originally developed for English. This procedure was repeated three times for the training, development, and test set. Subsequently, we trained a semantic parser on the training partition, optimized it on the development set, and evaluated it on the test set. For our cross-lingual experiments on German, Italian, and Dutch, the semantic parsers developed via the One-to-Many method were evaluated on gold-standard annotations released with the PMB. In our scalability experiments, the semantic parser is evaluated against silver-standard annotations. We, therefore, caution that the parser's actual performance would be slightly worse if it were compared against gold-standard data. We elaborate on this point shortly.

Table 5.4 presents our results which are clustered according to language family (we only report the performance of Tree 𝕌DRS models for the sake of brevity). All models for all languages used the cross-lingual BERT tokenizer and pre-trained word embeddings (Devlin et al., 2019) as input representations. As can be seen, the majority of languages we experimented with are Indo-European. In this family, the highest F_1 is 86.65 for Kurdish, Latin and Luxembourgish. In the Austronesian family, our parser performs best for Hawaiian and Samoan (F_1 is 86.65). In the Afro-Asiatic family, Hebrew achieves the highest F_1 of 79.44. In the Niger-Congo family, the highest F_1 is 74.57 for Swahili. In the Turkic family, our parser performs best for Turkish (F_1 is 74.68). In the Dravidian, Uralic, Sino-Tibetan, and Tai-Kadai families, Kannada, Estonian, Myanmar and Thai obtain the highest F_1, respectively. The worst parsing perfor-

Language	F_1	Language	F_1	Language	F_1	Language	F_1	Language	F_1	Language	F_1
Indo-European		Icelandic	81.65	Serbian	79.25	Malagasy	69.84	Xhosa	69.33	Chinese	73.42
Afrikaans	82.15	Irish	73.73	Sindhi	60.45	Malay	78.72	Yoruba	68.97	**Myanmar**	**77.10**
Albanian	79.98	**Kurdish**	**86.65**	Sinhala	44.25	Maori	69.84	Zulu	70.02	Tai-Kadai	
Armenian	76.12	**Latin**	**86.65**	Slovak	78.35	**Samoan**	**86.65**	Turkic		Lao	56.66
Belorussian	79.04	Latvian	80.50	Slovenian	79.16	Afro-Asiatic		Azerbaijani	73.72	**Thai**	**78.97**
Bengali	76.89	Lithuanian	79.00	Spanish	81.82	Amharic	41.49	Kazakh	72.95	Others	
Bosnian	79.09	**Luxembourgish**	**86.65**	Swedish	84.17	Arabic	76.81	Kyrgyz	71.25	Basque	71.11
Bulgarian	81.35	Macedonian	79.21	Tajik	69.06	Hausa	68.06	**Turkish**	**74.68**	Esperanto	86.65
Catalan	78.73	Marathi	74.39	Ukrainian	80.65	Hebrew	**79.44**	Uzbek	69.34	Georgian	72.57
Corsican	74.95	Nepali	74.82	Urdu	74.10	Maltese	74.15	Dravidian		Haitian Creole	77.13
Croatian	79.63	Norwegian	83.90	Welsh	76.35	Somali	68.48	**Kannada**	**76.43**	Hmong	86.65
Czech	80.15	Pashto	61.65	Yiddish	75.94	Sudanese	75.55	Malayalam	75.65	Japanese	71.90
Danish	84.09	Persian	73.99	Austronesian		Vietnamese	76.17	Tamil	75.62	Khmer	53.10
French	80.52	Polish	80.84	Cebuano	73.01	Niger-Congo		Telug	75.20	Korean	75.80
Frisian	79.68	Portuguese	83.25	Filipino	74.31	Chichewa	72.84	Uralic		Mongolian	71.18
Galician	79.66	Punjabi	74.80	Frisian	79.68	Igbo	67.61	**Estonian**	**80.35**	*Average*	75.69
Greek	80.96	Romanian	79.46	**Hawaiian**	**86.65**	Shona	70.70	Finnish	80.16		
Gujarati	76.63	Russian	82.22	Indonesian	79.73	Sotho	69.47	Hungarian	75.63		
Hindi	76.11	Scots Gaelic	72.05	Javanese	74.31	Swahili	**74.57**	Sino-Tibetan			

Table 5.4: Parsing results by Tree-o2m for 99 languages individually and on average; languages are grouped per language family and sorted alphabetically; best results in each family are shown in bold.

language	BLEU	F_1
de	66.03	95.30
it	63.00	89.17
nl	67.67	93.84
avg	65.56 ($\pm$2.10)	92.77 ($\pm$2.53)

Table 5.5: Comparison between gold- and silver-standard PMB test sets in German, Italian, and Dutch using BLUE and COUNTER; standard deviations are shown in parentheses.

mance is obtained for Amharic (F_1 of 41.49), while for the majority of languages, our parser is in the 70-86% ballpark. Better parsing performance correlates with a higher quality of machine translation and statistical word alignments. We will make these datasets publicly available as means of benchmarking semantic parsing performance and hoping that some of these might be manually corrected.

We further investigated the quality of the constructed datasets by extrapolating from experiments on German, Italian, and Dutch for which a gold-standard test set is available. Specifically, using the one-to-many method, we constructed silver-standard test sets and compared these with their gold-standard counterparts provided in the PMB. We first assessed translation quality by measuring the BLEU score (Papineni et al., 2002). We also used COUNTER to evaluate the degree to which silver-standard $\mathbb{U}$DRSs deviate from gold-standard ones. As shown in Table 5.5, the average BLEU (across three languages) is 65.12 while the average F_1 given by COUNTER is 92.23. These results indicate that the translation quality is rather good, at least for these three languages, and the PMB sentences. COUNTER scores further show that annotations are transferred relatively accurately, and that silver-standard data is not terribly noisy, where approximately 8% of the annotations deviate from the gold standard. In the next section, we further investigate what these deviations are.

5.3.6 Translation Divergences

Our cross-lingual methods depend on machine translation and alignments, which can be affected by the translation divergences. In this section, we discuss how translation divergences influence our methods. We focus on seven types of divergences high-

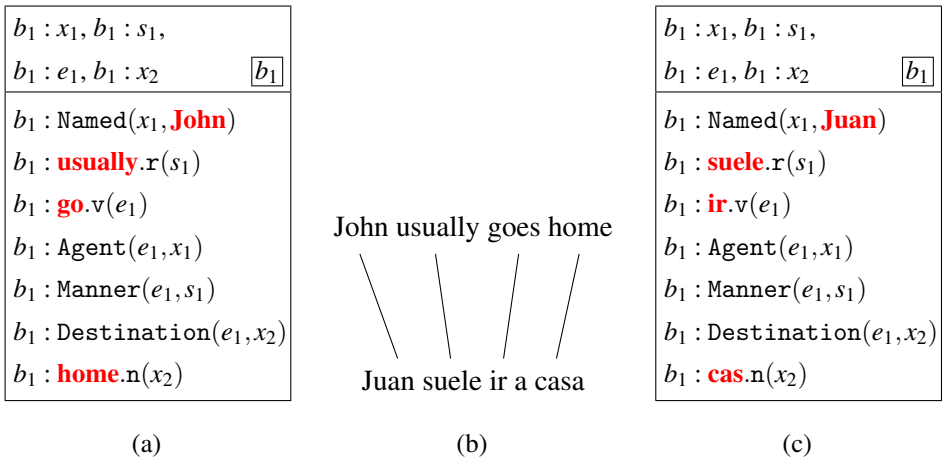


Figure 5.5: Examples of promotional divergence. (a) 𝕌DRS of English sentence *John ususally goes home*; (b) word alignments between the two sentences; (c) Incorrect 𝕌DRS of the Spanish translation *Juan suele ir a casa*, which is constructed by alignments.

lighted in Dorr (1994), i.e. Promotional Divergence, Demotional Divergence, Structural Divergence, Conflational Divergence, Categorical Divergence, Lexical Divergence, and Thematic Divergence, which could (not always) happen when translating one language to another. Then, we show whether the proposed 𝕌DRS representation can handle these and explain why.

Promotional Divergence Promotional Divergence is a divergence where a logical modifier of a main verb position can be changed. For example, consider the English sentence *John usually goes home* and its Spanish translation *Juan suele ir a casa* (John tends to go home), where the modifier (*usually*) is realized as an adverbial phrase in English but as the verb (*sueler*) in Spanish. As shown in Figure 5.5, to obtain the Spanish 𝕌DRS, the English words are replaced with aligned words. However, the adverbial *usually* is replaced with the verb *suele*, which together with the thematic relation Manner qualifies how the action *ir* is carried out. The divergence raises a **Category Inconsistency** in 𝕌DRS, which means the categories (or part-of-speech) of the replacing words are not consistent with the original ones.

Demotional Divergence In Demotional Divergence, a logical head into an internal argument position can be changed. For example, consider the English sentence *I like eating* and its German translation *Ich esse gern* (i eat likingly), where the head (*like*)

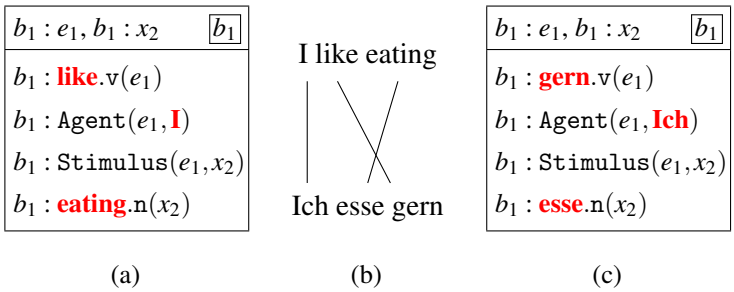


Figure 5.6: Example of demotional divergence. (a) 𝕌DRS of English sentence *I like eating*; (b) word alignments between two sentences; (c) incorrect 𝕌DRS of German translation *Ich esse gern*, which is constructed by alignments.

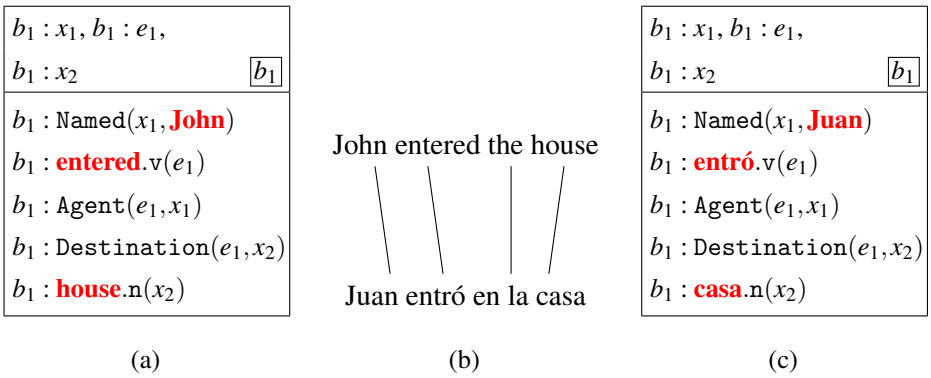
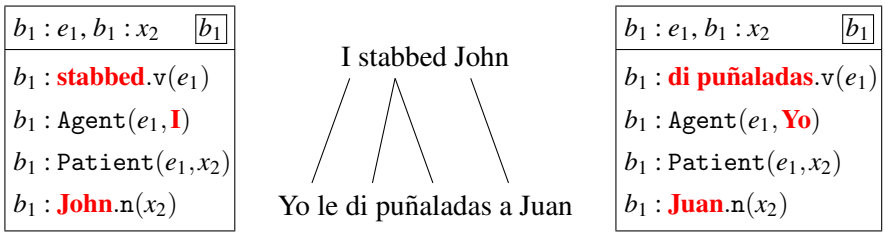


Figure 5.7: Examples of structural divergence. (a) 𝕌DRS of English sentence *John entered the house*; (b) word alignments between two sentences. (c) correct 𝕌DRS of Spanish translation *Juan entró en la casa*, which is constructed by alignments.

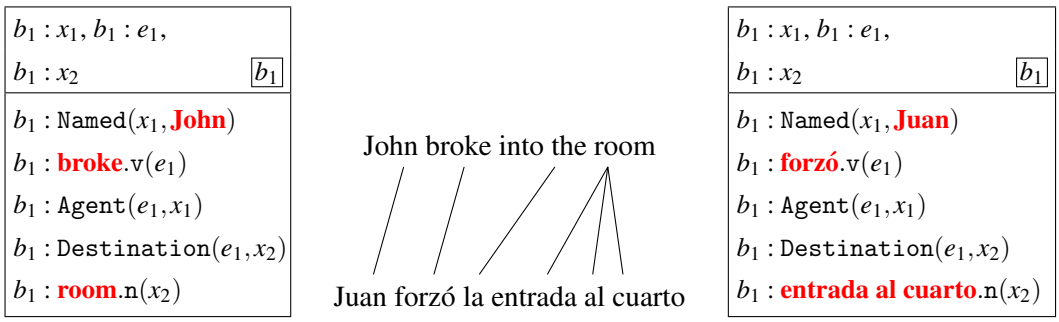
is realized as a verb in English but as the adverbial satellite in German. Figure 5.6 shows the alignments between the two sentences with their 𝕌DRSs. Similar to the promotional divergence, this also leads to **Category Inconsistency** in 𝕌DRS, because German word *gern* should be an adverb, not a verb.

Structural Divergences Structural Divergences are different in that syntactic structure is changed, and as a result, the syntactic relations also become different. For example, for the English sentence *John entered the house*, the Spanish translation is *Juan entró en la casa* (John entered in the house), where the noun phrase (the house) in English becomes a prepositional phrase (en la casa) in Spanish. However, 𝕌DRSs only care about the thematic relation Destination, showing the place the agent goes to with an action, which is shown in Figure 5.7. In this way, these divergences can be resolved.



(a) (b) (c)

Figure 5.8: Example of conflational divergence. (a) 𝕌DRS of English sentence *I stabbed John*; (b) word alignments between two sentences; (c) correct 𝕌DRS of Spanish translation *Yo le di puñaladas a Juan*, which is constructed by alignments.



(a) (b) (c)

Figure 5.9: Example of lexical divergence. (a) 𝕌DRS of English sentence *John broke into the room*; (b) word alignments between two sentences. (c) correct 𝕌DRS of Spanish translation *Juan forzó la entrada al cuarto*, which is constructed by alignments.

Conflational and Lexical Divergence We discuss the two types of divergence together. Words or phrases in the source language can be paraphrased using various descriptions in target languages. In conflational divergence, for example, the English sentence *I stabbed John* is translated into the Spanish as *Yo le di puñaladas a Juan* (I gave knife-wounds to John), which uses the paraphrase *di puñaladas* (gave knife-wounds to) to describe the English word *stabbed*, and the 𝕌DRSs are shown in Figure 5.8. Similarly, the 𝕌DRS for lexical divergence is shown in Figure 5.9. 𝕌DRS resolves these divergences with the many-to-many word alignments, which align the corresponding words in target and source language. For example, *stabbed* is replaced with *di puñaladas* to obtain Spanish 𝕌DRS from the English 𝕌DRS. In this way, these divergences can be resolved.

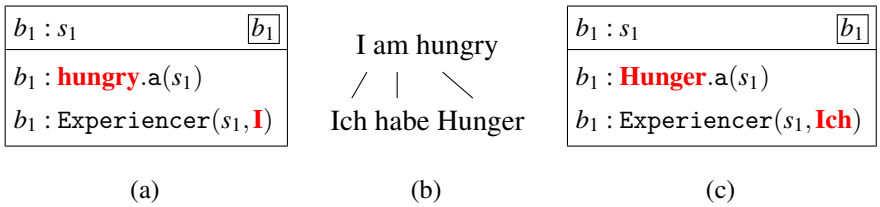


Figure 5.10: Example of catergorical divergence. (a) 𝕌DRS of English sentence *I am hungry*; (b) word alignments between two sentences; (c) incorrect 𝕌DRS of German translation *Ich habe Hunger*, which is constructed by alignments.

Figure 5.11: Example of thematic divergence. (a) 𝕌DRS of English sentence *I like Mary*; (b) word alignments between two sentences; (c) incorrect 𝕌DRS of Spanish translation *María me gusta a mí*, which is constructed by alignments.

Categorical Divergence The lexical categories (or parts of speech) might change from source to target language. For example, the English sentence *I am hungry* is translated to the German as *Ich habe Hunger* (I have hunger), where the adjective *hungry* in English is translated to the noun *Hunger* in German. As shown in Figure 5.10, *Hunger* in the German 𝕌DRS is a state (s_1) with a experiencer (I), it should be adjective, which yields **Category Inconsistency**.

Thematic Divergence Thematic relations are governed by the main verbs of the sentences, so thematic roles could be changed when translating verbs. For example, the English sentence *I like Mary* is translated to Spanish as *María me gusta a mí* (Mary pleases me), where the subject (I) in English is changed to the object (*me*) in Spanish. As shown in Figure 5.11, word alignments can capture semantic content mapping between the two sentences, but the Spanish 𝕌DRS keep the thematic relations governed by the main verb unchanged, which causes **Thematic Inconsistency**.

In total, there are two inconsistencies when non-English 𝕌DRSs are constructed with translation and alignments from English 𝕌DRSs, i.e., **Category Inconsistency**

	de	it	nl	zh
correct	44	46	45	32
trans error	1	0	0	4
trans divergence	1	0	0	2
align error	4	4	5	12

Table 5.6: The number of correct and incorrect German, Italian and Dutch $\mathbb{U}$DRSs . "trans error", "trans divergence" and "align error" show the number of incorrect $\mathbb{U}$DRSs caused by translation errors, translation divergence and alignment errors, respectively

and **Thematic Inconsistency**. **Category Inconsistency** can be addressed with the help of language-specific knowledge base by learning a function $f(s,c) = (s',c')$, where s and c are a translated word and an original category, respectively, and s' and c' are corrected word and category. Addressing the problem **Thematic Inconsistent** is more difficult, which requires to compare the aligned governed verbs between non-English and English at the linguistic level to decide if thematic relations have to be changed.

In order to see how many constructed $\mathbb{U}$DRSs are incorrect and what errors cause the incorrect $\mathbb{U}$DRSs, we randomly sample 50 constructed German, Italian and Dutch $\mathbb{U}$DRSs. The statistics are shown in Table 5.6. We found that most of the incorrect $\mathbb{U}$DRSs are due to the alignment errors. The translation divergences did not often happen when we used the machine translation system. We also sample and analyze 50 constructed $\mathbb{U}$DRSs in Chinese, which are in a different language family to English. There are still not many divergences. The reasons have two parts. One is that the used machine translation system is general from English to non-English without considering the divergences (Goyal and Sinha, 2009; Khan et al., 2018), and the other is that the English sentences in PMB are short, which structures are simple to translate. So we suggest that the translation divergences still have to be considered when the translation is performed on long sentence or by humans.

5.4 Related Work

The idea of leveraging existing English annotations to overcome the resource shortage in other languages by exploiting translational equivalences are by no means new.

A variety of methods have been proposed in the literature under the general framework of *annotation projection* (Yarowsky and Ngai, 2001; Hwa et al., 2005; Padó and Lapata, 2005, 2009; Akbik et al., 2015; Evang and Bos, 2016; Damonte and Cohen, 2018; Zhang et al., 2018) which focuses on projecting existing annotations on source-language text to the target language, while other work focuses on *model transfer* where model parameters are shared across languages (Cohen et al., 2011; McDonald et al., 2011; Søgaard, 2011; Wang and Manning, 2014). Our cross-lingual parsers rely on *translation systems* following two ways commonly adopted in the literature (Conneau et al., 2018; Yang et al., 2019; Huang et al., 2019): translating the training data into each target language (one-to-many) to provide data to train a semantic parser per language, and using a translation system at test time to translate the input sentences to the training language (many-to-one). Our experiments show that the combination of one-to-many and 𝕌DRS representations allows to speed-up meaning bank creation and the annotation process.

5.5 Summary

In this chapter, we introduced 𝕌niversal Discourse Representation Structures (𝕌DRSs) as a variant of canonical DRSs; 𝕌DRSs link elements of the DRS structure to tokens in the input sentence and are ideally suited to cross-lingual learning. Also, 𝕌DRSs are capable of bridging deep contextual word representations with semantic logic in sentences, and they can be interpreted by language-specific knowledge base. We further proposed a general framework for cross-lingual learning based on neural networks and state-of-the-art machine translation and demonstrated it can incorporate various DRT formats (e.g., trees vs. clauses) and is scalable.

Chapter 4 and Chapter 5 present models for mapping the texts written in various languages to their meaning representation in DRS. After understanding the text, the machines should give feedbacks to the human in human-understand languages. In the next chapter, we will discuss how to generate text given the DRT-based meaning representations.

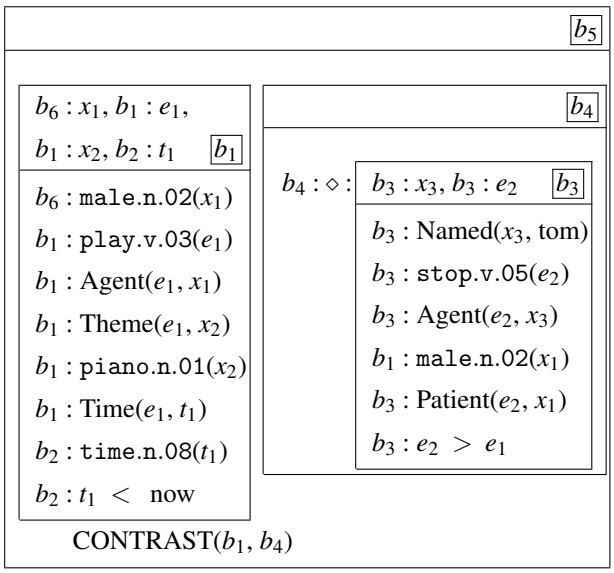
Chapter 6

Text Generation in DRS

It is not uncommon for text generation systems to produce natural language output from intermediate semantic representations (Yao et al., 2012; Takase et al., 2016). The literature presents several examples of generating text from logical forms underlying various grammar formalisms (Wang, 1980; Shieber et al., 1990; Carroll and Oepen, 2005; White et al., 2007), typed lambda calculus (Lu and Ng, 2011), Abstract Meaning Representations (AMR; Flanigan et al. 2016b; Konstas et al. 2017; Song et al. 2018; Beck et al. 2018; Damonte and Cohen 2019; Ribeiro et al. 2019; Zhu et al. 2019; Cai and Lam 2020b; Wang et al. 2020), Discourse Representation Theory (DRT; Basile and Bos 2011; Basile 2015), and Minimal Recursion Semantics (MRS; Horvat et al. 2015; Hajdik et al. 2019). Also, recently, natural language processing tasks involving document analysis have become increasingly popular, such multi-document summarization, dialog generation, discourse machine translation (Kim et al., 2019) and storytelling (Ferraro et al., 2019).

In this work, we propose neural models to generate high-quality text from semantic representations based on Discourse Representation Structures (DRSs). DRSs are the basic meaning-carrying units in Discourse Representation Theory (DRT; Kamp 1981; Kamp and Reyle 1993; Asher and Lascarides 2003), a formal semantic theory designed to handle a variety of linguistic phenomena, including anaphora, presuppositions (Van der Sandt, 1992; Venhuizen et al., 2018), and temporal expressions within and across sentences. DRSs are scoped meaning representations, and they capture the semantics of negation, modals, and quantification.

Compared to other semantic formalism, DRSs cover more linguistic phenomena and organize them together. Given these semantic constraints, the text generation will be more coherent and controllable. The temporal expressions contribute to the verbs



The man is going to play the piano. Tom may stop him.

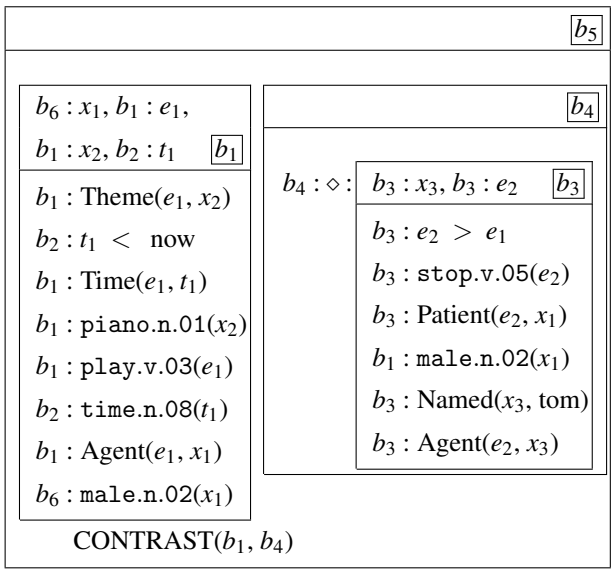
Figure 6.1: DRS in box-format for the discourse "*The man is going to play the piano. Tom may stop him.*".

generation with correct tenses and aspects. The discourse relations contribute to the connectives between word spans within and across sentences. The semantic scopes contribute negation and modals generation. The presuppositions and coreferences contribute to the generation of pronouns and articles.

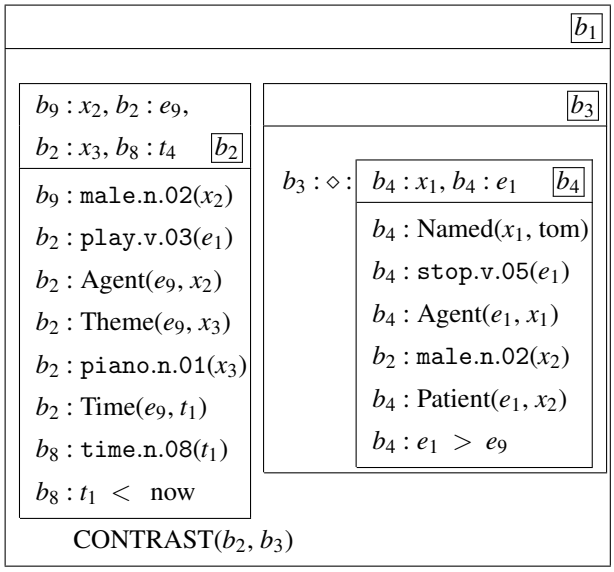
Figure 6.1 displays in box format the meaning representation for a discourse consisting of two sentences. The outermost box is a *segmented* DRS expressing the rhetorical relation CONTRAST between box b_1 representing the first sentence and box b_4 representing the second sentence. Boxes b_1 and b_4 are DRSs, the top layers contain variables (e.g., x_1, x_2) indicating discourse referents and the bottom layers contain conditions (e.g., Named(x_3, tom)) representing information about discourse referents. Variables and conditions have pointers (denoted by b in the figure) pointing to the boxes where they should be interpreted.[1] Predicates are disambiguated to their Wordnet (Fellbaum, 1998) senses (e.g., male.n.02 and play.v.03).

Although there has been considerable activity recently in developing models which analyze a text in the style of DRT (van Noord et al., 2018b, 2019; Liu et al., 2019a, 2018a; Fancellu et al., 2019), attempts to *generate* text from DRSs have been few and far between (however see Basile 2015 and Narayan and Gardent 2014 for notable

[1] Box b_6 is a presuppositional box for the interpretation of *the man* in the context of the two-sentence discourse.



(a)



(b)

Figure 6.2: DRS from Figure 6.1 with (a) shuffled conditions and (b) different variable names.

exceptions). This is primarily due to two properties of DRS-based semantic representations which render generation from them challenging. Firstly, DRS conditions are *unordered* representing a *set* (rather than a list).[2] A hypothetical generator would have to produce the same output text for any DRSs which convey the same meaning

[2]An exception are conditions in segmented DRSs whose order can be retrieved deterministically based on the arguments of rhetorical relations. For example, given the relation BECAUSE(b_1, b_3), we can assume that box b_1 precedes b_4.

but appear different due to their conditions having a different order (see Figures 6.1 and 6.2(a) which are otherwise identical, but the order of conditions in boxes b_1 and b_4 varies). The second challenge concerns *variables* and their prominent status in DRSs. Variables identify objects in discourse (such as entities and predicates), and are commonly used to model semantic phenomena including coreference, control constructions, and scope. In Figure 6.1, variables x, e, s, t, p, and b denote entities, events, states, time, propositions and boxes, respectively. Variable names themselves are arbitrary and meaningless, posing a challenge for learning. Our generator must verbalize different variable names to the same surface form. The meaning representations in Figures 6.1 and 6.2(b) are identical, and both correspond to the same discourse except that the variables have been given different names (b_5 in Figure 6.1 has been named b_1 in Figure 6.2(b), b_1 is now b_2, x_1 is x_2, e_1 is e_9, and so on).

These two problems are further compounded by the way DRSs are displayed, in a box-like format which is intuitive and easy to read but not convenient for modelling purposes. As a result, DRSs are often post-processed in a format that can be handled more easily by modern neural network models. For example, DRS variables and conditions are converted to clauses (van Noord et al., 2018b) or DRSs are modified to trees where each box is a subtree and conditions within the box correspond to children of the subtree, which are shown in previous Chapters.

In this chapter, we propose novel solutions to condition ordering and variable naming. We argue that even though DRS conditions appear unordered, they have a *latent* order due to biases in the way the training data is created. To give a concrete example, the Groningen Meaning Bank (GMB; Bos et al. 2017) provides the collection of English texts annotated with DRSs. These annotations were generated with the aid of a CCG parser (Clark and Curran, 2007); atomic DRS conditions were associated with CCG supertags and then semantically combined following the syntactic CCG derivations. Even annotators manually creating DRSs would be prone to follow a canonical order (e.g., listing named entities first, then verbal predicates and their thematic roles, and finally temporal conditions). We propose a graph-based model which learns to recover the latent order of conditions without explicitly enumerating all possible orders that can be prohibitive. We also handle variable names with a method that rewrites arbitrary indices to *relative* ones which are in turn determined by the order of conditions.

Following previous work, we convert DRSs to a more amenable format. Specifically, we consider DRS in tree format as the semantic representation input to our document generation task and generate a sequence of words autoregressively. We adopt an

encoder-decoder framework with a treeLSTM (Tai et al., 2015) encoder and a standard LSTM (Hochreiter and Schmidhuber, 1997) decoder. Problematically, DRS trees are wide, and the number of children for a given node can be as many as 180. Therefore, it becomes memory-consuming and sparse to assign a forget gate for each child as in the case of conventional (N-ary) treeLSTM (Tai et al., 2015). We propose a variant which we call *Sibling* treeLSTM that replaces N forget gates with a *parent* gate and a *sibling* gate. As a result, it reduces memory usage from $O(N)$ to $O(2)$, and is more suitable for modelling wide and flat trees.

Our contributions can be summarized as follows: (1) we formalize the task of neural DRS-to-text generation; (2) we provide solutions for the problems of condition ordering and variable naming, which render generation from DRS-based meaning representations non-trivial; (3) we propose a novel sibling treeLSTM model that can be also generally used to model wide tree structures.

6.1 Problem Formulation

Let S denote a DRS-based meaning representation. Given S as input, the aim of DRS-to-text generation is to output text T with the meaning of S:

$$T^* = \arg\max_{T \in \mathbb{T}} P(T|S,\Theta),$$

where $\mathbb{T}$ is the set of all possible texts, S has an arbitrary order of conditions and indexing of variables and Θ is the set of model parameters.

Our generation model is based on the encoder-decoder framework (Bahdanau et al., 2015) and operates over tree structures. Moreover, prior to training, variable names are rewritten so that their (arbitrary) indices denote relative order of appearance. We propose a novel sibling treeLSTM for encoding tree structures. The decoder is a sequential LSTM equipped with an attention mechanism generating word sequence $T = [t_1, t_2, ..., t_m]$, where m is the length of the text. At test time, DRS conditions are normalized, i.e., they are reordered following a canonical order learned from data and used as input to our generation model.

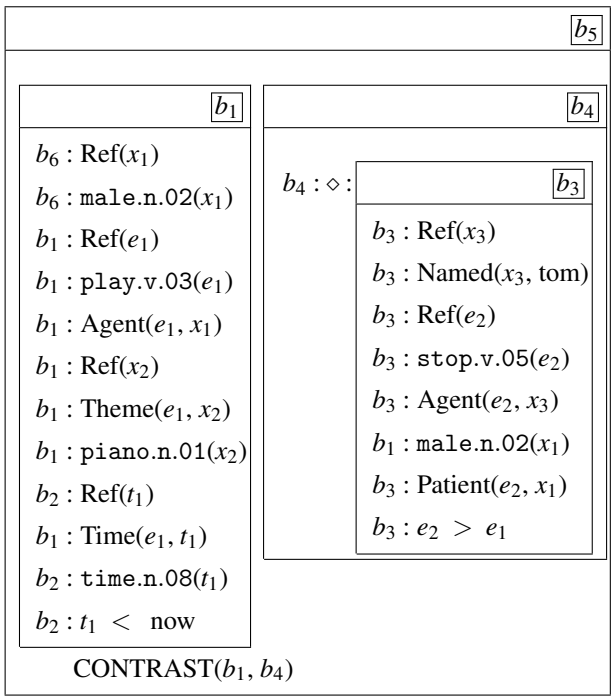
We first revisit the DRS-to-tree conversion and introduce variable renaming procedures (Sections 6.2). We next present the generation models (Section 6.3) and explain how DRS conditions are ordered (Section 6.4).

6.2 Relative Variables

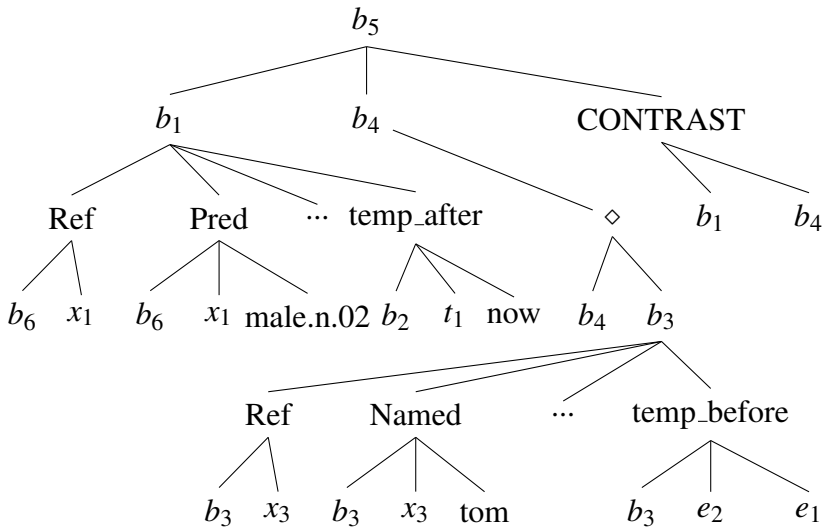
The DRS-to-tree conversion algorithm constructs trees based on DRS conditions in the bottom box layers and DRS variable in the top box layers via introducing *special* conditions $b : \text{Ref}(v)$, where v is a variable index and b is the label of the corresponding box. For example, $b_6 : x_1$ in Figure 6.1 becomes special condition $b_6 : \text{Ref}(x_1)$ and is placed before condition $b_6 : \texttt{male.n.02}(x_1)$ in Figure 6.3(a). The correpsonding tree-style DRS are shown in Figure 6.3(b).

We rename variables with regard to their relative position in a given DRS following a predefined traversal order. The sequence of *box variables* is obtained by traversing DRSs in an outer-to-inner and left-to-right manner, e.g., $[b_5, b_1, b_4, b_3]$ in Figure 6.3(a). For SDRSs, we replace variables in discourse relations with $K_i, i \in \mathbb{N}$, where i denotes th ith box from left to right. For example $\text{CONTRAST}(b_1, b_4)$ in Figure 6.3(a) is rewritten to $\text{CONTRAST}(K_0, K_1)$. Variables and conditions within presupposition boxes are rewritten to $B_i, i \in \mathbb{Z}$, where i denotes the distance of the current box to the presupposition box. For example, $b_1 : \text{Agent}(e_1, x_1)$ is rewritten to $B_0 : \text{Agent}(e_1, x_1)$ because it is in the current box b_1, while $b_1 : \texttt{male.n.02}(x_1)$ is rewritten to $B_{-2} : \texttt{male.n.02}(x_1)$ because it is in box b_3 and two hops away from presupposition box b_1. We use special label O for presupposition boxes pertaining to semantic content outwith the current DRS. For example, $b_6 : \text{Ref}(x_1)$ is rewritten to $O : \text{Ref}(x_1)$ because it introduces a new presupposition box, and $b_6 : \texttt{male.n.02}(x_1)$ is rewritten to $O_0 : \texttt{male.n.02}(x_1)$ because the condition can only be interpreted in this new presupposition box (now O_0 and previously b_6).

We obtain a sequence of *general variables* by traversing conditions as they appear in the DRS. Variables introduced for the first time are denoted by their type (going from left-to-right), while subsequent mentions of the same variables are rewritten with relative indices denoting their distance from the position where they were first introduced. Take Figure 6.3(a) as an example. The sequence of general variables is $[x_1, x_1, e_1, e_1, e_1, x_1, x_2, e_1, x_2, x_2, t_1, e_1, t_1, t_1, t_1, x_3, x_3, e_2, e_2, e_2, x_3, x_1, e_2, x_1, e_2, e_1]$, and is rewritten to $[X, X_0, E, E_0, E_0, X_0, X, E_0, X_0, X_0, T, E_0, T_0, T_0, T_0, X, X_0, E, E_0, X_{-2}, E_0, X_{-2}, E_0, E_{-1}]$. The DRS from Figure 6.3(a) is shown in Figure 6.4(a) with relative variables. The corresponding tree-style DRS are shown in Figure 6.4(b).



(a)

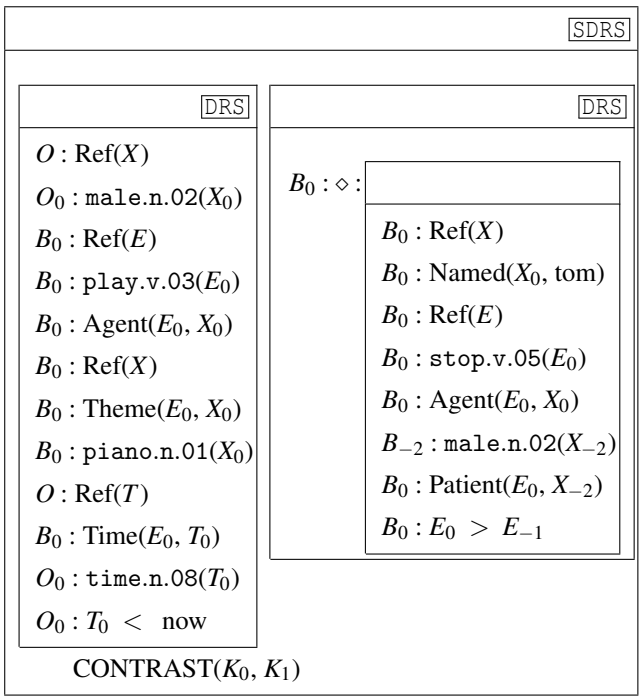


(b)

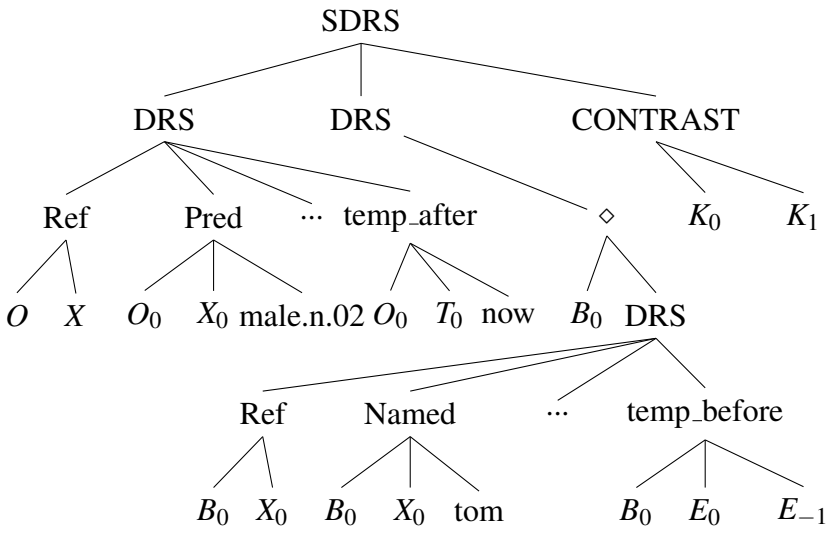
Figure 6.3: (a) Box-style DRS where the variables in top layers are merged to the bottom layers via introducing *special* conditions; (b) Tree-style DRS converted from its box-style DRS by the box-to-tree conversion algorithm in Chapter 5.

6.3 Generation Models

The generation models are based on an encoder-decoder framework by end-to-end training, equipped with standard LSTM-based sequential decoders. We introduce sev-



(a)



(b)

Figure 6.4: (a) Box-style DRS with the relative variables; (b) Tree-style DRS converted from its box-style DRS with relative variables by the algorithm in Chapter 5.

eral encoders prior to our proposed sibling TreeLSTM.

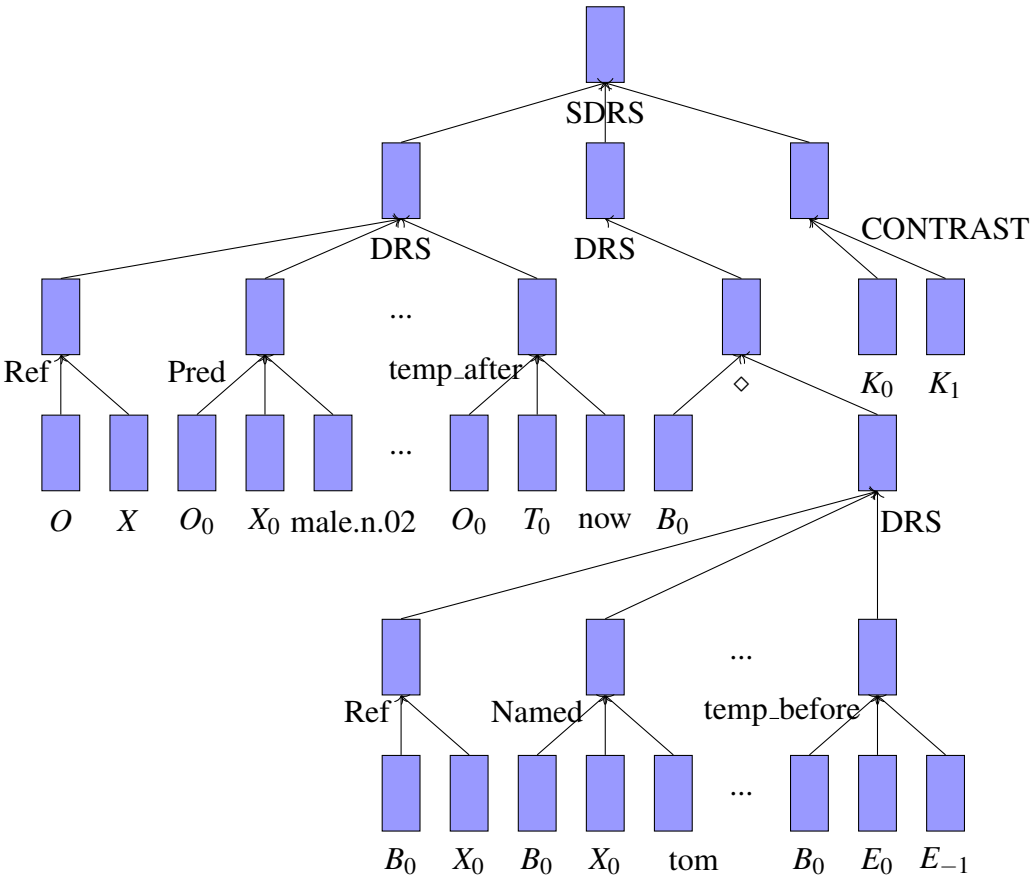


Figure 6.5: The standard TreeLSTM proposed by Tai et al. (2015). Each node in the tree has a hidden representation that is updated by the node input (e.g., B_0) and their child nodes.

6.3.1 Sequential Encoder

We adopt bidirectional LSTM as the baseline encoder. The trees are top-down and left-right linearized as $X = [x_1, x_2, ...x_n]$, where n is the length of the input linearized tree. We encode the sequence with a BiLSTM to obtain hidden representations $H = [h_1, h_2, ..., h_n]$ of the input:

$$[h_1, h_1, ..., h_n] = \text{BiLSTM}([x_1, x_1, ..., x_n]) \tag{6.1}$$

6.3.2 Standard TreeLSTM Encoder

A limitation of sequential encoders is that they only allow sequential information propagation without considering the input structure. The standard TreeLSTM (Tai et al., 2015) is designed to encode text with their tree structures, originally for the syntactic

structures. The hidden representations of the standard TreeLSTM cells are updated from the children nodes, as shown in Figure 6.5. There are two updating ways, i.e. Childsum and N-ary, according to the trees' property.

Childsum TreeLSTM Childsum TreeLSTM is well-suited for trees with high branching factor or whose children are unordered. The hidden representation for the jth node is updated as:

$$\begin{aligned}
\tilde{h}_j &= \sum_{k \in C(j)} h_k \\
i_j &= \sigma(W^i x_j + U^i \tilde{h}_j + b^i) \\
o_j &= \sigma(W^o x_j + U^o \tilde{h}_j + b^o) \\
f_{jk} &= \sigma(W^f x_j + U^f h_k + b^f) \\
u_j &= \tanh(W^u x_j + U^u \tilde{h}_j + b^u) \\
c_j &= i_j \cdot u_j + \sum_{k \in C(j)} f_{jk} \cdot c_k \\
h_j &= o_j \cdot \tanh(c_j),
\end{aligned} \tag{6.2}$$

where x_j is the input representation for jth node, $C(j)$ is the set of children indices under the jth node, and $W \in R^{d \times d}$, $U \in R^{d \times d}$ and $b \in R^d$ are learned parameters, and d is the hidden dimension. For the node j, all the child nodes representations, $h_k, k \in C(j)$, are summed to $\tilde{h}_j$ that is used together with input representation x_j to produce a input gate i_j, a output gate o_j, and the input information u_j. The input gate control the input information flow, the output gate control how much information is used as the hidden representation h_j from the memory c_j. For each child node, a forget gate is produced to control the information coming from the memories of their child nodes. The parameters includes 12 matrices, i.e., $O(8d^2 + 4d)$.

***N*-ary TreeLSTM** N-ary TreeLSTM is well-suited for trees with N children at most and whose children are ordered. The hidden representation for the jth node is updated

as:

$$
\begin{aligned}
i_j &= \sigma(W^i x_j + \sum_{l=1}^{N} U_l^i h_{jl} + b^i) \\
o_j &= \sigma(W^o x_j + \sum_{l=1}^{N} U_l^o h_{jl} + b^o) \\
f_{jk} &= \sigma(W^f x_j + \sum_{l=1}^{N} U_{kl}^f h_{jl} + b^f) \\
u_j &= \tanh(W^u x_j + \sum_{l=1}^{N} U_l^u h_{jl} + b^u) \\
c_j &= i_j \cdot u_j + \sum_{l=1}^{N} f_{jl} \cdot c_{jl} \\
h_j &= o_j \cdot \tanh(c_j),
\end{aligned}
\tag{6.3}
$$

where x_j is the input representation for jth node, and $W \in R^{d \times d}$, $U \in R^{d \times d}$ and $b \in R^d$ are learned parameters. The procedure is similar to Childsum TreeLSTM but the methods of obtaining gates. Instead of summing the child nodes representations, it assigns individual parameters for each child nodes. The parameters include $4 \times N + 8$ matrices, i.e. $O(4d^2 + 4Nd^2 + 4d)$.

6.3.3 Sibling TreeLSTM Encoder

In our case, DRS tree structures are additionally wide (the longer a document, the wider the tree) and relatively flat. To better model these aspects, we propose a TreeLSTM encoder that takes *sibling* information into account. The sibling TreeLSTM is suited for tree with various branches and whose children are ordered. The hidden representations of the sibling TreeLSTM cells are updated from the sibling nodes and the last children nodes, as shown in Figure 6.6. The hidden representation for the jth node is updated as:

$$
\begin{aligned}
i_j &= \sigma(W^i x_j + U_s^i h_{js} + U_p^i h_{jp} + b^i) \\
o_j &= \sigma(W^o x_j + U_s^o h_{js} + U_p^o h_{jp} + b^o) \\
f_{js} &= \sigma(W^f x_j + U_{ss}^f h_{js} + U_{sp}^f h_{jp} + b^f) \\
f_{jp} &= \sigma(W^f x_j + U_{ps}^f h_{js} + U_{pp}^f h_{jp} + b^f) \\
u_j &= \tanh(W^u x_j + U_s^u h_{js} + U_p^u h_{jp} + b^u) \\
c_j &= i_j \cdot u_j + f_{js} \cdot c_{js} + f_{jp} \cdot c_{jp} \\
h_j &= o_j \cdot \tanh(c_j),
\end{aligned}
\tag{6.4}
$$

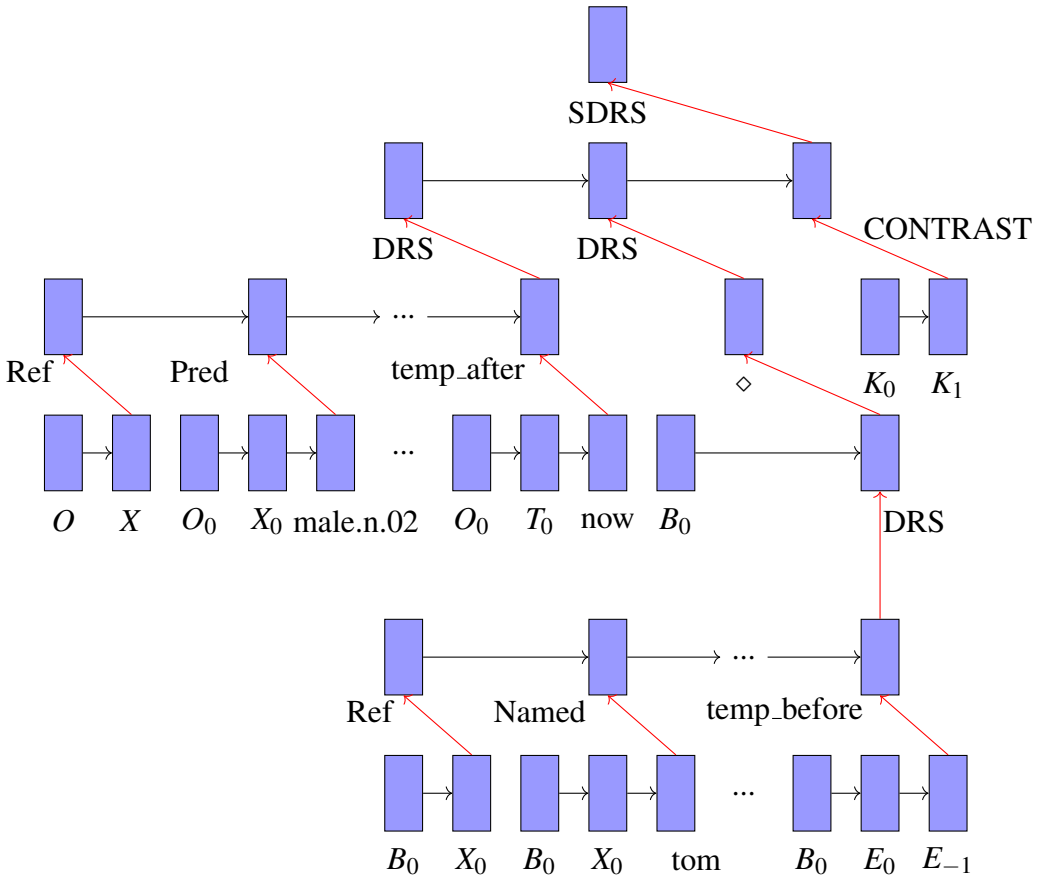


Figure 6.6: The sibling treeLSTM, where the blue boxes are hidden representations of nodes, the black arrows are sibling information flow, and the red arrows are parent information flow.

where x_j is the input representation, h_{js} and h_{jp} are the hidden representation of the previous sibling node of the jth node and the hidden representation of its last children of the jth node, respectively, and $W \in R^{d\times d}$, $U \in R^{d\times d}$ and $b \in R^d$ are learned parameters. For each node j, we obtain its cell input representation u_j, its input gate i_j and output gate o_j, and two forget gates f_{js} and f_{jp} for its neighbor cell and the last child cell, respectively. The memory of the current cell c_j is updated by the gated sum of its cell input representation and the memories of its neighbor and child cells. The hidden representation of current node h_j is computed with its output gate o_j. The parameters include 16 matrices, i.e. $O(12d^2+4d)$.

Finally, the hidden representations of the nodes are achieved through the sibling treeLSTM as $[h_1, h_2, ..., h_{n'}]$, where n' is the number of nodes in the trees. The decoder is a standard LSTM with global attention (Bahdanau et al., 2015).

6.4 Condition Order Recovery

As discussed previously, DRSs at test time may exhibit an arbitrary order of conditions, which our model should be able to handle. Our solution is to reorder conditions prior to generation by learning a latent canonical order from training data (e.g., to recover boxes b_1 and b_3 in Figure 6.1 from boxes b_1 and b_3 in Figure 6.2(a)). More formally, given a set of conditions R_{set}, we obtain an optimal ordering $R = [r_1, r_2, ..., r_n]$:

$$R^* = \arg\max_{R \in \pi(R_{set})} \textsc{Score}_{\mathbb{K}}(R|R_{set}), \tag{6.5}$$

where $\pi(R_{set})$ are all permutations of R_{set}, and R^* is the order with the highest likelihood according to $\textsc{Score}_{\mathbb{K}}$. Here, $\mathbb{K}$ parametrizes $\textsc{Score}$ as *knowledge* we collect from our training data by observing canonical orders of conditions. Unfortunately, the time complexity of calculating Equation (6.5) is $O(n!)$, we must enumerate all possible permutations for a set of conditions with n as large as 180. Since this is prohibitive, we resort to *graph ordering* which allows us to recover the order of the conditions without enumeration.

6.4.1 Graph Construction

We construct a graph from the set of DRS conditions which we break down into graph nodes and edges. Conditions in DRSs can be *simple* or *complex* according to their type of arguments (see Table 6.1). A simple condition might have a relation name with two arguments (e.g., Named(x_3, "tom") and Agent(e_1, x_3)), while a complex condition has a scoped name (e.g., possibility $\diamond$) and takes one or more DRSs as arguments. Simple conditions are denoted by a 3-tuple (l_s, a_0, a_1), where l_s is the condition name (e.g., Named and Agent) and a_0 and a_1 are its first and the second argument, respectively, which could be a variable or constant (e.g., e_1, x_3 and "piano.n.01").[3] Complex conditions are a 2-tuple (l_c, V_r), where l_c is the scope name, and V_r the set of arguments scoped by the condition. For example, the set of arguments for the possibility scope ($\diamond$) in Figure 6.1 is $\{e_1, e_2, x_1, x_3$, "tom", "stop.v.05", "male.n.02"$\}$.

Nodes Construction Condition names become nodes in our graph (see Table 6.2). Simple conditions are further divided into constant and thematic nodes. Constant nodes are constructed by concatenating the relation name in the condition with the

[3] Condition `piano.n.01`(e_1) equals to Pred(e_1, `piano.n.01`)

Category		Examples
simple	constant	Name(x_1, tom), play.v.03(e_1)
	thematic	Agent(e_1, x_3), temp_after(e_2, e_1)
complex		implication($\rightarrow$), negation($\neg$)

Table 6.1: The condition categories with examples.

Nodes	Conditions	Features
Pred male.n.02	male.n.02(x_1)	$a_0 = x_1$
Pred play.v.03	play.v.03(e_1)	$a_0 = e_1$
Agent	Agent(e_1, x_1)	$a_0 = e_1$, $a_1 = x_1$
Theme	Theme(e_1, x_2)	$a_0 = e_1$, $a_1 = x_2$
Pred piano.n.01	piano.n.01(x_2)	$a_0 = x_2$
Time	Time(e_1, t_1)	$a_0 = e_1$, $a_1 = t_1$
Pred time.n.08	time.n.08(t_1)	$a_0 = t_1$
temp_after	temp_after(t_1, now)	$a_0 = t_1$

Table 6.2: Conditions and their corresponding graph nodes

constant argument (e.g., condition male.n.02(x_1) becomes the node with label "Pred male.n.02"). Thematic nodes correspond to the relation name of the thematic condition (e.g., Agent(e_1, x_1) becomes the node with label "Agent"). Complex nodes correspond to the name of complex conditions (e.g.,$\diamond$).

Edge Construction We insert edges between graph nodes if these share arguments. For example, in Figure 6.7, there is an edge connecting node "Pred male.n.02" with Agent as they share argument x_1. We label this edge with a_1 to denote the fact that it is the second argument of Agent. Another edge is drawn between "Pred play.v.03" and Agent (as they share argument e_1) with label a_0 denoting that this is the first argument of Agent. Edges between nodes are bidirectional, with inverse edges bearing the suffix "-of". Edges drawn between constant and complex nodes bear the label "Related", while edges between two constant nodes (with the same variables) bear the label "Equal". The construction is shown in Algorithm 9.

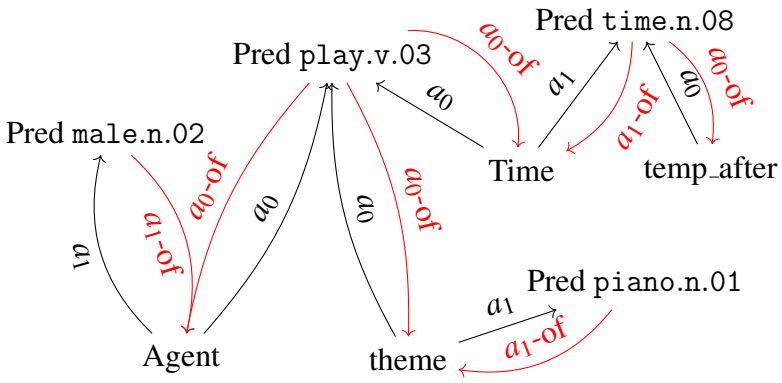


Figure 6.7: graph with inverse edges (shown in red).

6.4.2 Ordering Model

Formally, given graph $G = (R_{set}, E)$, where $R_{set} = \{r_1, r_1, ..., r_n\}$ is the set of nodes and E is the set of edges in G, our model outputs R^* as the optimal order of R_{set}. As shown Figure 6.2, each node is a sequence of words. A BiLSTM is applied to obtain the representation x_i of each node $r_i = [w^i_,..., w^i_m]$:

$$x_i = \text{BiLSTM}([w^i, ..., w^i_m]).$$

Encoder We encode the graph with a Graph Convolutional Recurrent network (GCRN; Seo et al. 2018). For each node r_i, we collect information from neighbor hidden representations with a gate controlling the information flow from neighbors to current nodes:

$$h'^k_i = \sum_j g^k_{ji} \cdot h^{k-1}_j;$$

$$g^k_{ji} = \sigma(f([e_{ji}, h^{k-1}_i, h^{k-1}_j])),$$

where e_{ji} is the embedding of edges from node r_j to r_i, g^k_{ji} is the gate from node r_j to node r_i, h'^k_i is the representation of the information collected from their neighbor nodes, and k is the recurrent step in the GRU. The node hidden representations are updated as:

$$h^k_i = \text{GRUCell}([x_i; h'^k_i; g^{k-1}_G], h'^k_i)$$

$$g^k_G = \text{GRUCell}(\frac{1}{n}\sum_i h^k_i, g^{k-1}_G)$$

where g_G represents the hidden representation of the graph as the average of (hidden) node representations, and GRUCell denotes the gated recurrent cell function. We obtain the hidden representations of nodes in the final recurrent step (K) as $H^K = \{h^K_1, h^K_2, ..., h^K_n\}$.

Algorithm 9 Edges Construction For Graph Ordering

Input: N_t thematic nodes; N_c, complex nodes; N_e, constant nodes

Output: E, set of edges

$E = []$
for n_t, n_e in N_t, N_e **do**
  if $n_t.a_0$ is $n_e.a_0$ **then**
    $E = E \cup \{n_t \xrightarrow{a_0} n_e\}$
    $E = E \cup \{n_e \xrightarrow{a_0\text{-of}} n_t\}$
  end if
  if $n_t.a_1$ is $n_e.a_0$ **then**
    $E = E \cup \{n_t \xrightarrow{a_1} n_e\}$
    $E = E \cup \{n_e \xrightarrow{a_1\text{-of}} n_t\}$
  end if
end for
for n_c, n_e in N_c, N_e **do**
  if $n_e.a_0$ in $n_c.V$ **then**
    $E = E \cup \{n_c \xrightarrow{\text{Related}} n_e\}$
    $E = E \cup \{n_e \xrightarrow{\text{Related-of}} n_c\}$
  end if
end for
for n_e, n'_e in N_e, N_e **do**
  if $n_e.a_0$ is $n'_e.a_0$ and n_e not is n'_e **then**
    $E = E \cup \{n_e \xrightarrow{\text{Equal}} n'_e\}$
  end if
end for

Decoder Our *decoder* adopts a Pointer Network (PN; Vinyals et al. 2015) to obtain the orders with highest probability:

$$\textsc{Score}_{\mathbb{K}}(R|R_{set}) = \text{PN}(R|R_{set}, H^K, \theta)$$

where θ are the parameters of the Pointer Network. We avoid enumerating all possible permutations for a set of nodes by generating their order autoregressively with an LSTM-based Pointer Network:

$$\text{PN}(R|R_{set}, H^K, \theta) = \prod_i P(r_i|r_{<i}, H^K)$$

$$P(r_i|r_{<i}, H^K) = \text{softmax}(v^{\text{T}}\tanh(W[h_i^d; H^K])),$$

Task	train	dev	test
Generation	7,970	992	1,038
Condition Ordering	133,332	16,493	17,624

Table 6.3: GMB dataset statistics; number of documents (generation) and number of different sequences of conditions (ordering).

where h_i^d is the ith step hidden representation of the Pointer Network, and v, W are parameters. The hidden representation h_i^d is updated by the input representation of the $(i-1)$th ordered node: $h_i^d = \text{LSTMCell}(x_{r_{i-1}}, h_{i-1}^d)$. All parameters are optimized with standard back-propagation.

6.5 Experiments

Our experiments were carried out on the Groningen Meaning Bank (GMB; Bos et al. 2017) which provides a large collection of English documents annotated with DRSs. We used the standard training, development, and test splits that come with the corpus distribution. All DRSs in the GMB were preprocessed into the tree-based format discussed in Section 6.2. We also extracted from the training data conditions and their order for training our graph ordering model. Dataset statistics are shown in Table 6.3.

6.5.1 Condition Ordering

Models and Settings Before evaluating our generator, we assess the effectiveness of the proposed condition ordering model (see Section 6.4). Specifically, we compare four kinds of graphs: *NoEdges*, is a graph without edges; *FullEdges*, is a complete graph where each pair of nodes has edges; *SiGraph*, is the proposed graph without bidirectional edges; and *BiGraph*, is the proposed graph with bidirectional edges (see Figure 6.7). We also consider **Counting**, a baseline model which greedily orders pairs of conditions according to their frequency of appearance in the training data.

For all models, the embedding dimension is 50 and the hidden dimension 300. The bidirectional LSTM used for representing the graph nodes has a single layer, and the recurrent step in the GCRN is 2 ($K = 2$). We applied the Adam optimizer (Kingma and Ba, 2014). We use accuracy to measure the percentage of absolute orders which are predicted correctly and Kendall's τ coefficient to measure the relationship between

Models	Acc (%)	τ	Parameters
Counting	14.38	0.5668	—
NoEdges	44.64	0.7488	6.1M
FullEdges	45.79	0.7545	6.1M
SiGraph	59.47	0.8541	6.1M
BiGraph	69.10	0.8919	6.1M

Table 6.4: Results for condition ordering (dev set).

Models	BLEU	Parameters
Seq	71.79	47.4M
ChildSum	72.98 (+1.19)	47.1M
Nary	73.24 (+1.45)	49.2M
Sibling	**74.22 (+2.43)**	49.2M

Table 6.5: Results of Ideal-world generation on dev set (improvements compared to Seq in parentheses).

two lists of ordered items; τ ranges from -1 to 1, where -1 means perfect inversion and 1 means perfect agreement.

Results Table 6.4 summarizes our results. SiGraph performs better than NoEdges (+14.83% accuracy), showing that edge information is necessary to obtain the node representations used for conditions ordering. FullEdges performs worse than SiGraph (−13.68% accuracy), underlying the fact that graph structure matters (i.e., edges are helpful when connecting certain pairs of nodes). BiGraph achieves the best ordering performance by a large margin compared to SiGraph (+9.63 % accuracy). One possible reason is that bidirectionality ensures all nodes have incoming edges, making it possible to pass more information.

6.5.2 Ideal-World Generation

Models and Settings We first examine generation performance in an ideal setting where (gold standard) condition orders are given and the indices of variables are fixed. We compared the proposed treeLSTM against *Seq*, a baseline sequence-to-sequence

model which adopts a bidirectional LSTM as its encoder. The trees were linearized in a top-down and left-to-right, $X = [x_1, x_2, ...x_n]$, where n is the length of the tree. We obtained hidden representations $H = [h_1, h_2, ..., h_n]$ of the input with:

$$[h_1, h_2, ..., h_n] = \text{BiLSTM}([x_1, x_2, ..., x_n])$$

In addition, we included various models with tree-based encoders: *ChildSum*, is the bidirectional childsum-treeLSTM encoder of Tai et al. (2015); it operates over right-branch binarized trees; *Nary*, is the bidirectional Nary-TreeLSTM of Tai et al. (2015), again over right-branch binarized trees;[4] and *Sibling*, our bidirectional sibling-TreeLSTM. All of these models were equipped with the same LSTM decoder, global attention (Bahdanau et al., 2015), and the copy strategy of See et al. (2017).The embedding dimension was 300 and the hidden dimension 512. All encoders and decoders have 2 layers. We measure generation quality with case-insensitive BLEU (Papineni et al., 2002).[5]

Results Table 6.5 shows our results on the development dataset. Overall, treeLSTM models performs better (average +1.69 BLEU) than sequence models. Nary performs better (+0.26 BLEU) than ChildSum because the latter cannot model the order of children. Sibling performs best (74.22 BLEU), because it not only encodes the tree structure but also keeps track of sequential information.

6.5.3 Real-World Generation

Models and Settings We finally present our results in a more realistic setting where both problems of condition ordering and variable naming must be addressed. We recover condition order using four approaches: a *Naive* method that has no special-purpose ordering mechanism; the order of conditions is random in the development/test sets and fixed in the training set; *Random*, the order of conditions is random in the training, development, and test sets; *Counting*, the order of conditions is recovered by the Counting method; *GraphOrder* recovers the order of conditions with *BiGraph*. All comparison systems employ variable renaming as introduced in Section 6.2. We report experiments with a sequence-to-sequence generator and our sibling TreeLSTM.

[4] We experimented with n-ary ($n > 2$) trees, but found that binary trees perform best. Right-branch binary trees are also empirically better than left-branch ones.

[5] https://github.com/OpenNMT/OpenNMT-py

Models	BLEU
Seq+Naive	4.61
Seq+Random	24.34 (16.77)
Seq+Counting	45.17
Seq+GraphOrder	55.57
Sibling+Naive	6.98
Sibling+Random	43.43 (0.26)
Sibling+Counting	49.54
Sibling+GraphOrder	**58.73**

Table 6.6: Real-world generation (dev set). For Random, we report average results after shuffling 5 times (variance shown in parentheses).

Models	BLEU	Parameters
Graph	45.72	30.1M
Seq+GraphOrder	55.28	32.4M + 6.1M
Sibling+GraphOrder	59.26	34.5M + 6.1M

Table 6.7: Real-world generation (test set).

Results Table 6.6 summarizes our results on the development set. Naive performs poorly, indicating that both Seq and Sibling models are sensitive to the order of conditions. Random, has higher variance with Seq (+16.51) compared to Sibling. Hidden representations for each timestep in Seq are heavily influenced by all previous steps, which are sequentially encoded; subtrees are encoded as a unit in Sibling, which is a more global representation for capturing patterns. Overall, we observe that the order of conditions plays a key role in the generation: both Seq and Sibling models improve when the ordered conditions are explicitly incorporated (either with Counting or GraphOrder). We observe that the combination of Sibling with GraphOrder achieves the best results (58.73 BLEU).

Table 6.7 presents our results on the test set. We compare our Sibling encoder against a sequential one. Both models are interfaced with GraphOrder. We also compare to a previous graph-to-text model (Song et al., 2018; Damonte and Cohen, 2019) which has been used for generating from AMRs. We converted DRSs to graphs fol-

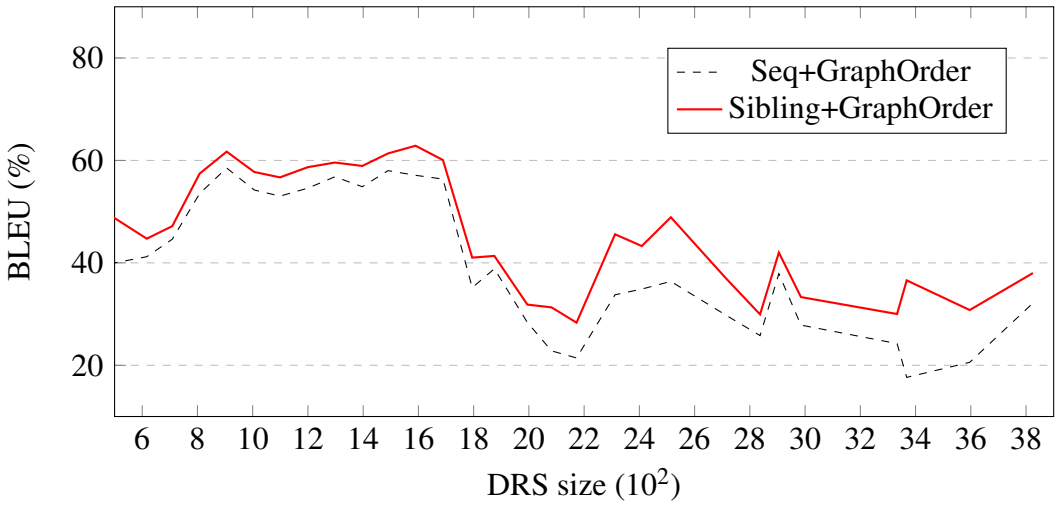


Figure 6.8: BLEU score against DRS size (test set).

lowing the method of Liu et al. (2020); graphs were encoded with a GCRN (Seo et al., 2018) and decoded with an LSTM. As can be seen, Sibling+GraphOrder outperforms all comparison systems achieving a BLEU of 59.26. However, compared to the ideal-world generation (see Table 6.5) there is still considerable room for improvement.

Figure 6.8 shows model performance on the test set against DRS size (i.e., the number of nodes in a DRS tree). Perhaps unsurprisingly, we see that generation quality deteriorates with bigger DRS size (i.e., with >1,600 nodes).

Although BLEU score has frequently been reported as correlating well with human judgement in machine translation task and text generation task, it only outputs the word overlap between the generated text and the gold text. In data-to-text generation tasks, BLEU score fails to measure the performance of hard-constraints text generation, e.g., measuring if the generated texts are coherent to the given semantics. So we manually analyze our outputs by examples.

Figure 6.9 shows examples of text generation from the test set. In the first example, the model generates the word *because* from the rhetorical relation, BECAUSE(b_{10}, b_{12}). Temporal information (highlighted in blue in the figure) is also accurately reflected in the generated text (*sell* is inflected to its present tense form). In addition, the model tends to over-generate (e.g., the word *dollar* is mentioned twice) and sometimes misses out on important determiners (e.g., *some*). In the second example, the model generates the word *themselves* referring to the entities mentioned before, e.g., x_{29} equals to x_{27} which refers to *inmates*, resolving the coreference. In the third example, the model generates the modal verb *must* in accordance with the scope operator

DRS	... DRS(b_{10} Pred(b_{10} x_{24} "speculator.n.01") Pred(b_{10} x_{25} "dollar.n.01") Pred(b_{10} e_8 "sell.v.01") Agent(b_{10} e_8 x_{24}) Theme(b_{10} e_8 x_{25}) Pred(b_4 t_1 "now.r.01") Equ(b_{10} x_{26} t_1) temp_includes(b_{10} t_5 x_{26}) temp_overlap(b_{10} e_8 t_5)) SDRS(b_{12} DRS(b_{11} Pred(b_3 x_5 "thing.n.12") Pred(b_{11} e_9 "expect.v.01") Agent(b_{11} e_9 x_5) ... BECAUSE(b_{10} b_{12})))
Gold	... **some** speculators are selling dollars because they expect ...
Ours	... , the dollar . speculators are selling dollars because they expect ...
DRS	... Pred(b_{16} e_{11} "be.v.00") Agent(b_{16} e_{11} x_{26}) Ref(b_{16} x_{27}) Card(b_{16} x_{27} 7) Ref(b_{16} x_{28}) Pred(b_{16} x_{27} "inmate n.01") Ref(b_{16} x_{29}) Equ(b_{16} x_{27} x_{29}) Ref(b_{17} x_{30}) Pred(b_{17} x_{30} "group.n.01") Ref(b_{16} e_{12}) Pred(b_{16} e_{12} "disguise.v.01") Theme(b_{16} e_{12} x_{29}) ...
Gold	... were among seven inmates who disguised themselves ...
Ours	... were among seven inmates disguised themselves ...
DRS	.... DRS(b_4 NEC(b_4 DRS(b_5 IMP(b_5 DRS(b_6 Ref(b_6 x_{11}) Ref(b_6 x_{10}) subset_of(b_6 x_{11} x_{10}) Ref(b_6 x_{12}) subset_of(b_6 x_{12} x_{10}) Ref(b_6 x_{13}) Pred(b_6 x_{13} "food.n.01") In(b_6 x_{11} x_{13}) Pred(b_6 x_{11} "goods.n.01") Ref(b_6 s_2) Topic(b_6 s_2 x_{12}) Pred(b_6 s_2 "manufactured.a.01") Pred(b_6 x_{12} goods n.01)) DRS(b_7 Ref(b_7 e_4) Pred(b_7 e_4 "import.v.01") Theme(b_7 e_4 x_{10}) Pred(b_3 t_1 "now.r.01") Ref(b_7 t_3) temp_included(b_7 e_4 t_3) temp_before(b_7 t_1 t_3)))$_{\text{IMP}}$)$_{\text{NEC}}$) ...
Gold	... all food and manufactured goods must be imported ...
Ours	... all food and manufactured goods must be imported ...

Figure 6.9: DRS example from test set with gold and automatically generated text by ours (Sibling+GraphOrder). Temporal information marked in blue, rhetorical relations marked in red, co-reference marked as green, and scope marked in brown.

NEC (a shorthand for Necessity, $\Box$). Also, the model generates *all* for *food* and *goods* corresponding to the Implication (IMP) condition (i.e., $\forall x(P(x) \rightarrow Q(x))$).

We provide one example output of our final model (*Sibling+GraphOrder*) for the DRS from GMB test dataset. The DRS in tree format with condition orders given by *GraphOrder* is shown in Figure 6.10, Figure 6.11 and Figure 6.12, where the DRS in Figure 6.11 is to expand the nonterminal b_{33} in Figure 6.10, and the DRS in Figure 6.12 is to expand the nonterminal b_{28} in Figure 6.11. The corresponding document generated by **Sibling** is:

> *the u.s. dollar hit a record low against the euro tuesday . it took a dollar , but 48 cents to buy one euro , and a series of problems including the key of the u.s. housing sector in which has been battered by the slowing n.01 ,*

including continuing the u.s. economy , the dollar . speculator are selling dollars because they expect that the u.s. central bank will try to stimulate the economy by cutting interest rates soon . u.s. lower interest rates can cut the return on investments . the falling dollar is prompting oil-rich nations around the persian gulf to consider ending the practice of linking the value of its currency to those of the dollar and instead supplement the u.s. currency . such a move would reduce demand for dollars and weaken the u.s. currency .

6.6 Related Work

Much previous work has focused on text generation from formal representations of meaning on isolated sentences or queries. The literature offers a collection of approaches to generating from AMRs, most of which employ neural models and structured encoders (Song et al., 2018; Beck et al., 2018; Damonte and Cohen, 2019; Ribeiro et al., 2019; Zhu et al., 2019; Cai and Lam, 2020b; Wang et al., 2020). Other work generates text from structured query language (SQL) adopting either sequence-to-sequence (Iyer et al., 2016) or graph-to-sequence models (Xu et al., 2018).

Basile (2015) was the first to attempt generation from DRT-based meaning representations. He proposes a pipeline system that operates over graphs and consists of three components: an alignment module learns the correspondence between surface text and DRS structure, an ordering module determines the relative position of words and phrases in the surface form and a realizer generates the final text. Narayan and Gardent (2014) simplify complex sentences with a two-stage model which first performs sentence splitting and deletion operations over DRSs and then uses a phrase-based machine translation model for surface realization.

Our work is closest to Basile (2015); we share the same goal of generating from DRSs, however, our model is trained end-to-end and can perform long-form generation for documents and sentences alike. We also adopt an ordering component, but we order DRS conditions rather than lexical items, and propose a model capable of inferring a global order. There has been a long-standing interest in information ordering within NLP (Lapata, 2003; Chen et al., 2016; Gong et al., 2016; Logeswaran et al., 2018; Cui et al., 2018; Yin et al., 2019; Abend et al., 2015). Our innovation lies in conceptualizing condition ordering as a graph scoring task which can be further realized with graph neural network models (Wu et al., 2020).

6.7 Summary

In this chapter, we have focused on *document-level* generation from formal meaning representations. We have adopted DRT as our formalism of choice and highlighted various challenges associated with the generation task. We have introduced a novel sibling treeLSTM for encoding DRSs rendered as trees and shown it is particularly suited to trees with wide branches. We have experimentally demonstrated that our encoder coupled with a graph-based condition ordering model outperforms strong comparison systems.

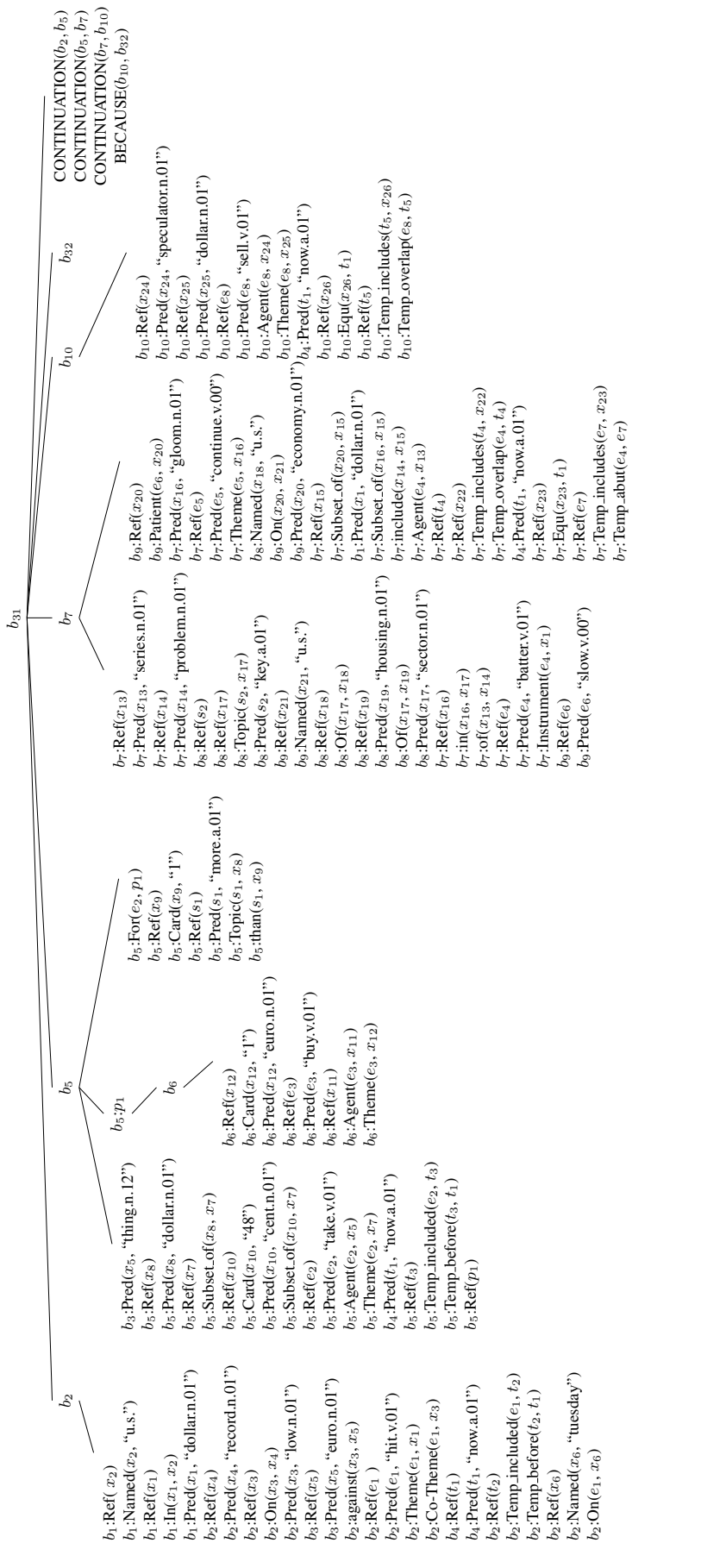


Figure 6.10: A partial DRS in tree format with condition orders recovered by **GraphOrder**.

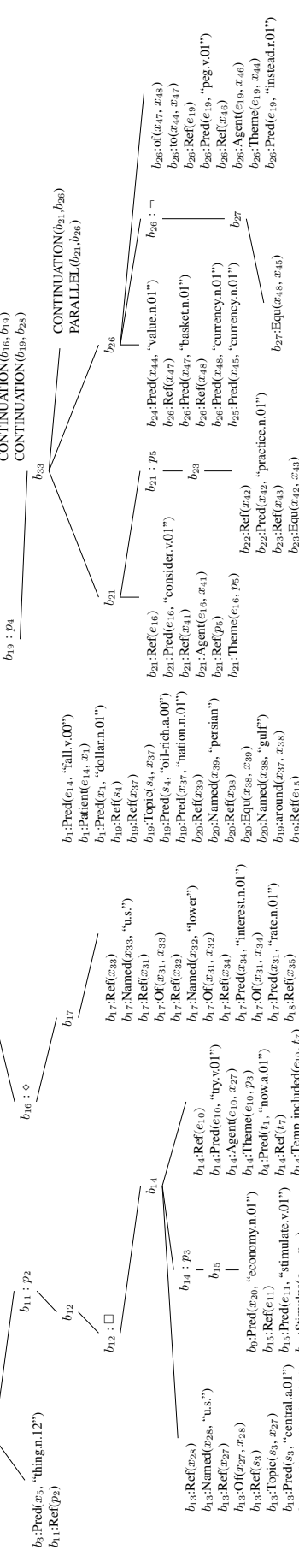


Figure 6.11: A partial DRS expanding the non-terminal node b_{32} of DRS in Figure 6.10.

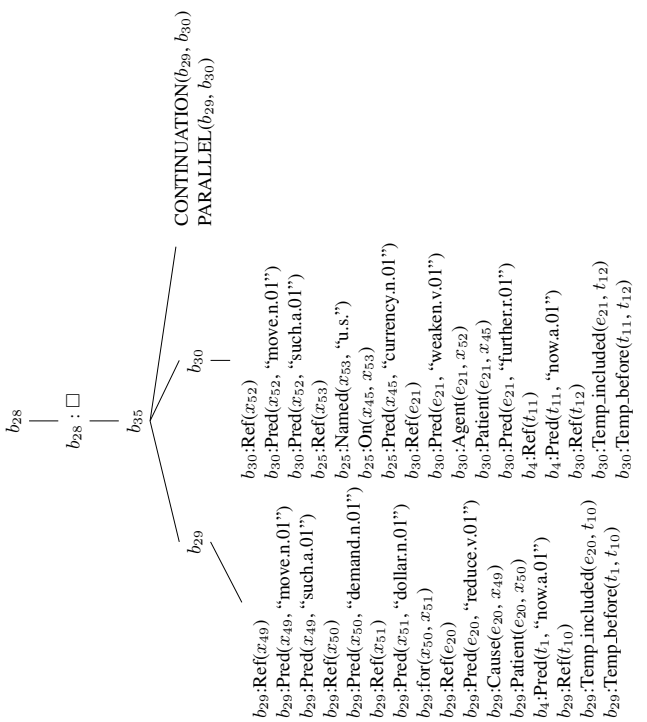


Figure 6.12: A partial DRS expanding the non-terminal node b_{28} of DRS in Figure 6.10.

Chapter 7

Interlingual Machine Translation

Recent neural machine translation systems (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015; Vaswani et al., 2017) are, for the most part, built on parallel corpora, without considering specific syntax transfer and what the meaning between languages is preserved. Although several approaches adopt syntactic trees or semantic representations as features, they generate sequences of words in the target language directly from word sequences in the source language. One disadvantage is that for each pair of languages, we have to construct a parallel bilingual corpus (i.e., n^2 models for n languages). In this chapter, we revisit the interlingual machine translation that has two steps. The source language is parsed to an interlingual formalism at the first step, and the target language is generated from the interlingual formalism at the second step (Richens, 1958; Hutchins and Somers, 1992). The benefits of the interlingual machine translation are that linguistic and translation problems are more clearly and usefully expressed in terms of a standard language (Richens, 1958); for each language, we construct a parser and generator to and from interlingual formalism, respectively (i.e., $2 * n$ models for n languages); this only requires a monolingual analyzer for specific language and interlingual formalism.

In the previous chapters, we developed the DRS parsers that automatically generate discourse representation structures for a given text and DRS-to-text generators that produce text according to discourse representation structures. Based on the success of these models, we explore interlingual machine translation taking DRS as the bridge semantic formalism, which splits the machine translation into two steps. The first step is monolingual semantic parsing which obtains the interlingual formalism for the source language, and the second step is monolingual text generation which produces the target language according to the interlingual formalism obtained in the first step.

We propose a general framework of neural interlingual machine translation that adopts DRT as interlingual formalism. Experiments on the Parallel Meaning Bank show that this approach outperforms baseline models but still lags behind commercial machine translation systems. We aim to renew the research interest on interlingual machine translation involving linguistics in the era of deep learning. Also, the interlingual machine translation framework puts the parsers and generators developed in this thesis together to perform a real-world task.

7.1 Background

Machine translation is a long-standing natural language processing task, which aims to design a system for automatically and accurately translating one language to another. The classical approaches to machine translation are rule-based methods, which generate translations on the basis of morphological, syntactic, and semantic analysis of both the source and the target languages. Thanks to the availability of large-scale parallel corpus, example-based machine translation (Nagao, 1984) that simulates the process of human translation by analogy, and statistical machine translation (Brown et al., 1993; Wu, 1997; Koehn et al., 2003; Galley et al., 2004; Liu et al., 2006; Chiang, 2007; Koehn, 2009) that statistically and automatically obtains synchronous translation rules have become popular. Recently, due to the success of deep learning, neural machine translation methods are mainstream approaches, which are built on the sequence-to-sequence model (Sutskever et al., 2014) equipped with long short-term memory unit (Hochreiter and Schmidhuber, 1997) or transformer layers (Vaswani et al., 2017), also together with extra mechanisms such as attention (Bahdanau et al., 2015), copying strategy (See et al., 2017), and coverage (Tu et al., 2016).

In the earlier stage, rule-based machine translation approaches are categorized into direct, transfer and interlingual methods (Hutchins and Somers, 1992; Dorr et al., 2004). We argue that the three categories cover rule-based machine translation and most machine translation systems, including statistical and neural machine translation, summarized in Figure 7.1.

Nearly all of the recent neural machine translation systems (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015; Vaswani et al., 2017) are examples of direct translation, which generates sequences of words in the target language directly from word sequences or strings in source languages. Before neural methods became popular, machine translation was based on the bilingual rules lexical

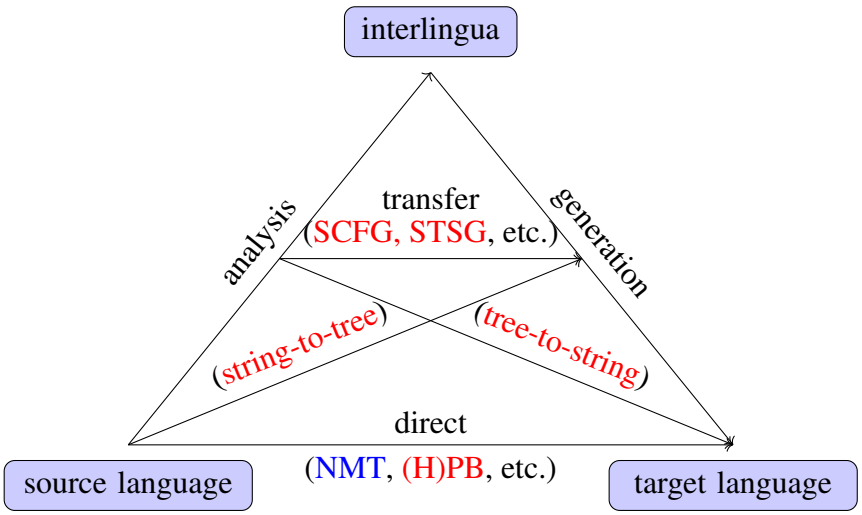


Figure 7.1: Bernard Vauquois' pyramid (Vauquois, 1968) showing comparative depths of intermediary representation with interlingual machine translation at the peak, followed by transfer-based, then direct translation. Statistical machine translation systems are marked as red and neural machine translation systems marked as blue. SCFG stands for Synchronous Context-Free Grammar, STSG for Synchronous Tree-Substitution Grammar, (H)PB for (hierarchical) phrase-based systems, and NMT for neural machine translation. String-to-tree is a set of systems which only adopt syntax in the source language. Tree-to-string is a set of systems which only adopt syntax in the target language.

or syntactic, which here manually designed or automatically extracted from bilingual corpus (statistical MT) (Koehn, 2009). This is a branch of transfer-based systems, where one transfer method (rules) is only for one pair of languages. However, interlingual machine translation has two steps. One is the source language parsing, and the other is the target language generation (Richens, 1958; Hutchins and Somers, 1992). When a new language appears, being able to parse the language to interlingua is sufficient; it only requires a monolingual analyzer for the specific language and interlingual formalism. Interlingual machine translation opens the question of how to define interlingual formalism.

7.2 Interlingual MT

Interlingual machine translation has two steps: source languages are analyzed (parsed) to the interlingual formalism, and then target languages are generated according to this

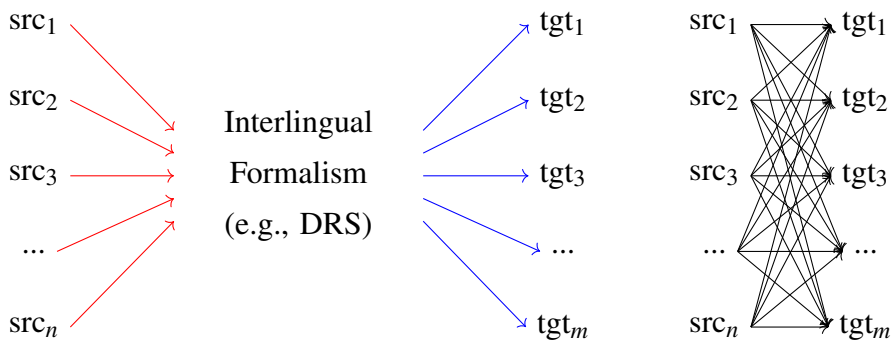


Figure 7.2: Framework of the interlingual machine translation (left), where source languages are analyzed (red) to the interlingual formalism, and the target languages are generated (blue) according to the interlingual formalism, and framework of the machine translation without interlingua (right), where models are trained for each pair of languages.

formalism. The general framework of interlingual MT is shown in Figure 7.2, and can be formulated as:

$$\begin{aligned} D^* &= \arg\max_{D \in \mathbb{D}} P(D|S,\Theta) \\ T^* &= \arg\max_{T \in \mathbb{T}} Q(T|D^*,\Phi), \end{aligned} \tag{7.1}$$

where S is a source text, T is a target text, D is a interlingual formalism, D^* and $\mathbb{D}$ are the optimized interlingual formalism and a set of candidate interlingual formalism, respectively, T^* and $\mathbb{T}$ are the optimized target text and a set of candidate target texts. Θ and Φ are the trainable parameters in the analyzer and the generator, respectively.

7.2.1 Interlingual DRS

Under the assumption that the texts with the same meaning share the same discourse representation structures (DRS) no matter in which language the texts are written, we consider DRSs as our interlingual formalism though they are described using only English symbols. According to the lossless box-to-tree conversion algorithm introduced in Chapter 4, DRS boxes are converted into trees used as our interlingual DRS.

7.2.2 Source Analyzer

We take a DRS parser as our source analyzer that outputs DRSs for a given input. The DRS parser is based on the encoder-decoder architectures equipped with Transformer

(Chapter 4), where the input is a sequence of words and the output is a bracketed linearized DRS in tree format. In order to make the language-independent source analyzer, different languages have their own DRS parsers designed with the cross-lingual methods introduced in Chapter 5.

7.2.3 Target Generator

The purpose of a target generator is to produce a target text according to the inter-lingual DRSs that represent the meaning of source input. So the target generation can be regarded as the DRS-to-text generation task introduced in Chapter 6. We adopt DRS-to-text generation framework. Instead of RNN-based models, we adopt the transformers for the generation task, where the input is a bracketed linearized DRS in tree format, and the output is a sequence of words.

7.3 Experiments

In this section, we describe the dataset used in our experiments, as well as details about our experiment settings. We focus on German-English, Italian-English, and Dutch-English machine translation.

7.3.1 Settings

Data Our experiments were carried out on the Parallel Meaning Bank 2.2.0 (Abzianidze et al., 2017), which has English (en), German (de), Italian (it) and Dutch (nl) sentences with their DRS annotations. The corpus comes with small-scale gold-standard data parallel in four languages that we use as a test set, and the rest are training data. The details are shown in Table 7.1.

Training and Evaluation We train separate DRS parsers for German, Italian and Dutch with the cross-lingual methods introduce in Chapter 5, and train a single DRS-to-text generator for English. All models share the same hyperparameters. The dimension of word embeddings is 300, the transformer encoder and decoder have 6 layers with an hidden size of 300, and the self- and the context-attentions adopt multi-head attention mechanism with 6 heads. The dimension of the position-wise feed-forward layer is 4096. We used Adam (Kingma and Ba, 2014) as the learning rate optimizer

languages	train			dev	test
	parse	generation	parallel		
de-en	131,525	249,168	72,150	1,163	1,966
it-en	72,308	249,168	40,479	2,071	1,058
nl-en	28,099	249,168	13,095	2,133	996

Table 7.1: Statistics on the data that we used in the experiments, where parse shows the number of sentences that we used to train a parser for a source language, generation shows the numebr of sentences that we used to train a generator to produce target language, parallel shows the number of sentences in the source language having a corresponding translation in the target language, and used to train a neural machine translation model. All data in dev and test are parallel.

with an initial learning rate of 0.001 with a 0.7 learning rate decay. We evaluated the translation quality using BLEU score (Papineni et al., 2002).

7.3.2 Systems

We build three machine translation systems in our experiments:

- **Direct**: a standard transformer-based machine translation system (Vaswani et al., 2017) as our baseline. The system is equipped with standard self-attention and context-attention in the encoder-decoder framework, which is the direct machine translation system.
- **Inter**: our proposed neural DRS interlingual machine translation system. In the full pipeline of interlingual machine translation, the interlingual DRS is automatically analyzed.
- **Google**: a strong commercial machine translation system from Google Translate.[1]

7.3.3 Results

Table 7.2 shows the results on the test set. Our interlingual machine translation outperforms the baseline by 3.03 BLEU on average, especially by 4.61 BLEU for German-

[1] https://translate.google.co.uk/

models	de-en	it-en	nl-en	average
Direct	38.87	43.58	40.25	40.90
Inter	43.48	44.11	44.61	44.07
Google	61.77	67.44	65.88	65.03
Inter-gold	67.06	67.59	68.51	67.72

Table 7.2: BLEU score (%) on test dataset by three models, where Inter-gold is Inter given the gold interlingual DRS.

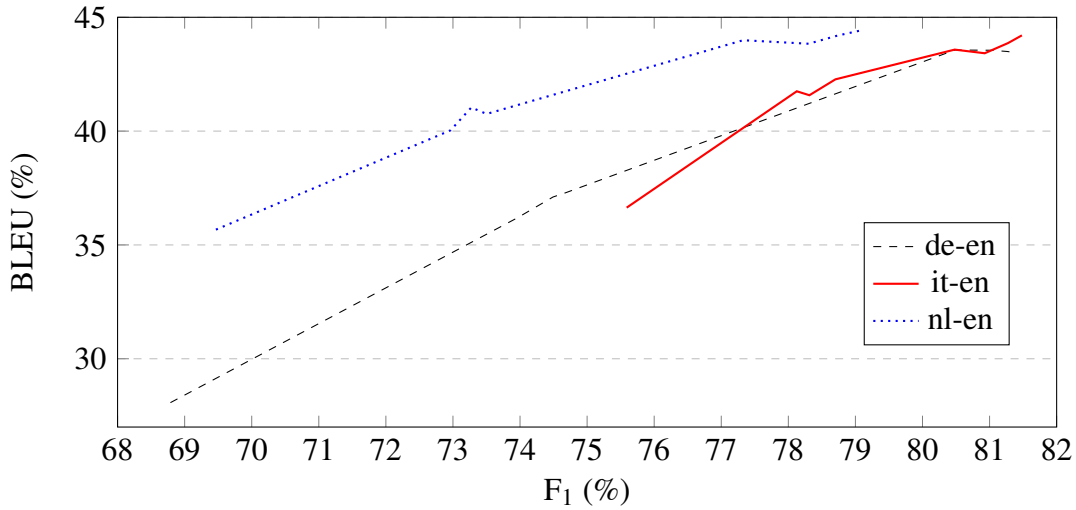


Figure 7.3: BLEU score against the quality of interingual DRS (F_1 score) on German-English (de-en), Italian-English (it-en), and Dutch-English (nl-en) language pairs.

to-English translation, 0.53 BLEU for Italian-to-English translation, and 4.36 BLEU for Dutch-to-English translation. However, Google Translate performs best by a large margin in the three language pairs. It is not surprising. Google Translate is a commercial system trained on vast scale data.

The Quality of Interlingual DRS Our interlingual machine translation system takes DRSs as the interlingual representation, which is automatically analyzed by a neural DRS parser. In order to see if the quality of the DRSs affects the performance of the machine translation system, we generate various-quality interlingual DRSs by using our DRS parser in different checkpoints during training. The Figure 7.3 shows the relation between the quality of the interlingual DRSs and the performances of the ma-

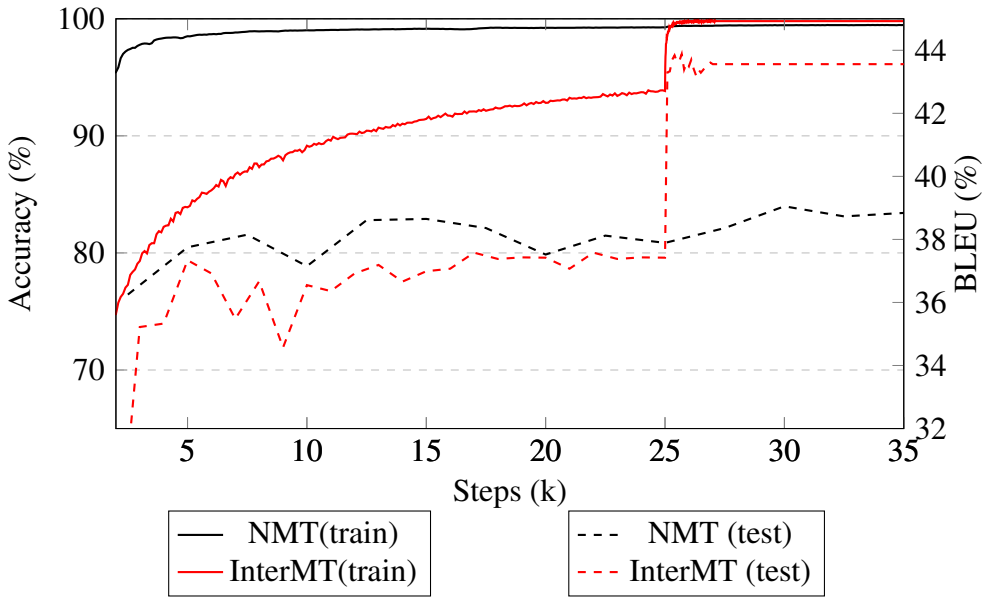


Figure 7.4: The learning curves of the neural machine translation system (direct translation) and DRS-interlingual machine translation on the German-English translation task (de-en).

chine translation system. Translation performances correlate positively with the quality of the interlingual DRS. As shown in table 7.2, given the gold interlingual DRSs for German, Italian and Dutch, our machine translation system outperforms the Google Translate.

Learning Curves Figure 7.4 shows the learning curves of the neural machine translation (NMT) and the DRS-interlingual machine translation (InterMT). NMT is fast to reach the convergence at the first 10k training steps, and the NMT performances on the test data cannot be further improved because the model is already fit to the training data. InterMT starts with low accuracy, and the accuracy is increased smoothly by taking more training steps. Due to the iterative training strategy, the performance has a big jump when we start training the model on gold-standard data after 25k steps. We can see that the gap between training and testing in InterMT is smaller than that in NMT.

7.4 Summary

In this chapter, we revisit interlingual machine translation with neural semantic parsers (Chapter 4 and 5) and text generators (Chapter 6) and take discourse representation structures as the interlingual meaning representations. Experimental results show that we achieve performance comparable to the standard end-to-end transformer-based machine translation methods on the Parallel Meaning Bank. The proposed DRS-based machine translation system outperforms the strong commercial Google Translate system when gold interlingual meaning representations are provided, which shows that developing an excellent semantic parser is necessary to build a better interlingual machine translation system.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

This thesis aimed to develop computational models for text analysis and generation following the principles and representational conventions of Discourse Representation Theory. A key feature of DRT is that it is a linguistically-motivated theory representing the meaning of text within and across sentences for various languages. By considering the modelling problem and applications, we developed a box-to-tree conversion algorithm, which transforms original box-style DRSs to tree-style DRSs that is amenable to structural modelling. Secondly, we proposed Universal Discourse Representation Structures which help anchor semantic symbols to their words in a language-independent way, allowing to bridge deep contextual word representations with semantic formalisms, and the representations can be interpreted by language-specific knowledge bases. Based on the formalisms and conversions, we have made used of the available annotated resources and shown that neural models are a viable alternative to manually-designed features and grammars. We enhance the neural models in the following ways. Firstly, 3-step decoding allows generating DRSs in coarse-to-fine steps being able to capture structural patterns. Secondly, the proposed one-to-many and the many-to-one methods address the problem of DRS parsing in low-resource languages. Thirdly, the sibling TreeLSTM is particularly suited to trees with wide branches.

Discourse Representation Theory can depict the meaning of text within and across sentences covering more linguistic information In this thesis, we compared DRT with other semantic formalism by discussing how they deal with various linguistic phenomena, such as predicate-argument, sense differentiation, scope, presupposition,

temporal expression, coreference and rhetorical relation. We showed that DRT is more expressive in that it can cover all the linguistic information (Chapter 2). Together with the available corpora being released, it is worth to explore the DRT analysis with deep learning.

Lossless conversion from box-style format to tree-style format helps to model DRS Based on the available DRS resources, we explored the DRS parsing task that outputs DRSs for input text. However, the original DRS format is box-like, which is intuitive and easy to read but not particularly amendable to structural modeling. We proposed a box-to-tree conversion algorithm to transform box-style DRSs to tree-style DRSs, and based on this conversion, we defined Discourse Representation Tree Structures (DRTSs) in Chapter 3. Furthermore, we improved the conversion by incorporating sense differentiation and presupposition without lossing generality in Chapter 4. The tree-style DRSs can be linearized and then fed to sequence-to-sequence neural models equipped with the long-short term memory and transformers.

3-step decoding with state trackers can significantly improve the performance of DRS parsing and ensure the well-formed DRSs The tree-style DRSs have explicit structural information that is hard to capture by traditional sequence modelling. As for structure modelling, we introduced a 3-step decoder that generates coarse DRS structures (i.e., the skeleton of tree) and then conditionally produces the relations and the variables. The sequential decoders and the 3-step decoders have no guarantee that tree output will be well-formed. We proposed the constraint-based inference using state trackers, where state trackers identified valid predictions for each step, and similar to finite-state machines, they were updated according to a valid prediction. Experimental results showed that the 3-step decoder with state trackers achieved significant improvement over competitive baselines.

Iterative training can effectively adopt the vary-quality annotations The creation of data with gold-standard DRS annotations remains an expensive endeavour requiring expert knowledge, e.g., familiarity with the semantic formalism and language at hand. There are only small-scale gold-standard data available for training, but large amounts of data with various qualities can be automatically obtained from the output of trained parsers and manual correction. We proposed an iterative training approach that trains the parsers on the possibly noisy data set and then fine-tunes on high-quality data

in Chapter 4. The experimental results showed that the parser could be significantly improved with iterative training.

***One-to-Many* and *Many-to-One* methods can significant improve DRS parsing for low-resource languages** DRS resources are mostly limited to English. In order to address DRS parsing for low-resource (non-English) languages, we proposed two cross-lingual methods in Chapter 5. The many-to-one approach works by translating non-English text to English and then running a relatively accurate English DRS parser on the translated text, while the one-to-many approach translates English to non-English text and trains multiple parsers (one per language) on the translations. The first method requires an online translation system, but one parser is for all languages. The second method requires an independent parser for each language with an offline translation system. Experimental results showed that the two methods could significantly improve low-resource DRS parsing compared to baselines using limited training data.

𝕌niversal Discourse Representation Structures can help to represent meanings of text across languages by grounding the symbol to words Traditional Discourse Representation Structures can represent meaning across languages by assuming that if non-English and English sentences have the same meaning, they also share the same DRSs written with English word senses. However, we argue that sense distinctions are not completely preserved across languages. In Chapter 5, we propose a 𝕌niversal Discourse Representation Structures (𝕌DRSs), which explicitly anchor DRSs to lexical tokens with the advantage of linking to deep contextual word representations in pre-trained language models and various monolingual and cross-lingual knowledge bases. Furthermore, we constructed a large-scale corpus that contains (silver-standard) 𝕌DRS annotations in 102 languages by machine translation and word alignments. We discussed translation divergences that could influence the constructions of 𝕌DRSs in non-English languages. By manual inspection, we found that most errors come from word alignments. Although there were not many divergences in our construction, translation divergences might have to be considered when the translation focuses on longer sentences.

DRS-to-text generation can produce high-quality texts at document level Previous work on text generation from intermediate semantic representations has mostly

focused on isolated sentences. In Chapter 6, we proposed a DRS-to-text generation task aiming to produce high-quality texts (multiple sentences) from semantic representations based on DRSs. A challenge in DRS-to-text generation lies in that DRS conditions are unordered, representing a set, not a list, and variables names themselves are arbitrary and meaningless. We proposed a graph neural model with a pointer net to recover the condition order in DRSs, and then renamed variables with their relative indices according to a predefined traversal. Experimental showed that condition order recovery and variable renaming play a key role in improving DRS-to-text generation.

Sibling treeLSTM can help the modeling for wide and flat trees The standard treeLSTM is designed to encode text with their tree structures, originally for the syntactic structures. We proposed the sibling treeLSTM that is more suitable for modelling wide and flat trees with various branches and ordered children. Instead of modelling each child, sibling treeLSTM models one neighbour node and one parent node, recurrently keeping the information flow from the neighbour nodes with reduced parameters. We take the sibling treeLSTM to our DRS-to-text generation task, and experimental results showed that the generation with a sibling treeLSTM outperforms that with a standard treeLSTM.

Discourse Representation Structures can be used in interlingual machine translation With the success of DRS parsing and DRS-to-text generation, we explored interlingual machine translation in Chapter 7. Here a cross-lingual DRS parser analyzes the source languages to a DRS-based meaning representation (interlingual DRS), and the fully-trained generator produces the target language accordingly. Interlingual machine translation only requires a parser and a generator for each language ($2n$) instead of the parallel corpus for each pair of languages (n^2) that is required in the recent-popular framework of machine translation (direct machine translation). Experimental results showed that interlingual machine translation outperforms direct machine translation in our setting, but cannot reach the performance of the commercial Google translate system. Furthermore, we found that the quality of interlingual DRS heavily affects the translations, and that interlingual machine translation achieves competitive results to Google translate if gold DRSs are provided, showing that developing a good parser is necessary to build a better interlingual machine translation system.

8.2 Future Work

This thesis explores the understanding and generation of natural language with discourse representation theory, motivating the design of neural models for discourse semantic analysis. In the future, it would be essential to explore the scalability of our models. The discourse representation structures depict the meaning of natural languages, so it is necessary to study how to exploit the discourse representation structures in downstream NLP tasks. We discuss some promising directions around discourse representation structures.

Data Collection More annotated training data usually can help neural model learning. Although there are several available corpora with DRS annotations, they are relatively small compared to the datasets used in other NLP tasks. Also, most annotations in Groningen Meaning Bank are not gold-standard, and texts are short in the Parallel Meaning Bank. The problem becomes serious for low-resource languages. The main reason is that DRS annotation requires expert knowledge. It is necessary to design a simplified way of annotating data (Bos, 2020), encouraging more people to join with crowdsourcing methods. DRSs have complicated structures with variables and conditions, but they have explicit patterns. For example, a verb predicate usually appears together with the thematic relation Agent and Patient. Annotators can choose the correct patterns instead of writing them down verbatim.

Semi-Supervised Learning All models presented in the thesis are trained on texts paired with DRS annotations. The models are inevitably limited to the training data domain, e.g., news, causing the problem of dealing with unseen words and training data patterns. In order to alleviate this bias, we can start with a small set of manual grammars or annotations as seeds to initialize the models, and then the grammars can be expanded iteratively by clustering on a large scale of raw data. The seed grammars should be highly diverse to cover several genres and domains.

Multi-view Semantics There are many ways to understand natural language. Although discourse representation structures depict more linguistic phenomena than other meaning formalisms (see Chapter 2), complementary formalism focuses on different aspects of semantics. For example, DELPH-IN MRS-derived Bi-Lexical Dependency (DM) and Prague Semantic Dependencies (PSD) both construct arcs with semantic label between words surface. However, they use the different structures determined by

linguistic signal alone and by the utterance context, respectively. The Universal Conceptual Cognitive Annotation (UCCA) framework aims to design semantic formalism across languages. The Abstract Meaning Representation (AMR) has the assumption that sentences with the same meaning share the same representations with necessary paraphrasing. These representation goals determine the different semantic formalism. In data-driven training, they provide various supervised signals. We can build these semantics with their corpora by multi-view learning and multi-task learning. For example, in the encoder-decoder framework, by sharing encoders, we can take all these available semantic signals to improve the performances of semantic parsing in these formalisms, including discourse representation structure parsing.

Pretrained Models Recently, systems built on large-scale pre-trained models achieve significant improvements in many natural language processing tasks. These pre-trained models are trained on raw text with the objective of predicting marked text or autoregressively generating the text. However, they hardly capture long-distance dependencies. Discourse representation structure can represent the meaning of the entire text (e.g., sentences and documents). Incorporating discourse representation structure learning into pre-trained models can improve the expressive ability of the pre-tained model. We can use the fully-trained DRS parsers to generate the DRSs for the texts automatically. The generated DRSs can be aligned to the word forms (refer to the Universal DRS shown in Chapter 5). If the objective is masked text prediction, we can also mask the corresponding DRSs and then predict them. Also, we can sequentially generate DRSs in tree format simultaneously when autoregressively generating the corresponding text.

Applications Discourse representation structures represent the meaning of text within and across sentences. Natural language applications over the documents can benefit from DRSs. For example, most document comprehension question answering systems are built on black-box transformers without any explicit inference, and it is difficult to target the errors made by the models. We can use the DRSs representations together with texts as the input of the QA systems to trace the parts of the meaning the models fail to adopt. This feedback can help the design of the models. Also, text summarization over multiple long documents requires the ability to model a large number of texts, considering linguistic phenomena such as discourse relation inference and anaphora coreference, while DRSs provide this information. We can parse documents

to their DRSs, summarize them to small DRSs, which retain the critical meaning of the documents, and then generate summaries from DRSs. The procedure would be similar to the interlingual machine translation (Chapter 7), where the natural language processing happens at the level of semantic representations rather than word forms.

Bibliography

Omri Abend, Shay B. Cohen, and Mark Steedman. 2015. Lexical event ordering with an edge-factored model. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1161–1171, Denver, Colorado. Association for Computational Linguistics.

Omri Abend and Ari Rappoport. 2013a. UCCA: A semantics-based grammatical annotation scheme. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers*, pages 1–12, Potsdam, Germany. Association for Computational Linguistics.

Omri Abend and Ari Rappoport. 2013b. Universal Conceptual Cognitive Annotation (UCCA). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–238, Sofia, Bulgaria. Association for Computational Linguistics.

Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos. 2017. The Parallel Meaning Bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 242–247, Valencia, Spain. Association for Computational Linguistics.

Alan Akbik, Laura Chiticariu, Marina Danilevsky, Yunyao Li, Shivakumar Vaithyanathan, and Huaiyu Zhu. 2015. Generating high quality proposition Banks for multilingual semantic role labeling. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 397–407, Beijing, China. Association for Computational Linguistics.

David Alvarez-Melis and Tommi S. Jaakkola. 2017. Tree-structured decoding with doubly-recurrent neural networks. In *Proceedings of the 5th International Conference on Learning Representation (ICLR)*, Toulon, France.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710, Lisbon, Portugal. Association for Computational Linguistics.

Nicholas Asher. 1993. Reference to abstract objects in English: a philosophical semantics for natural language metaphysics. *Studies in Linguistics and Philosophy. Kluwer, Dordrecht*.

Nicholas Asher and Alex Lascarides. 2003. *Logics of conversation*. Cambridge University Press.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, San Diego, California.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 86–90.

Miguel Ballesteros and Yaser Al-Onaizan. 2017. AMR parsing using stack-LSTMs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1269–1275, Copenhagen, Denmark. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Valerio Basile. 2015. *From Logic to Language: Natural Language Generation from Logical Forms*. Ph.D. thesis, University of Groningen, Netherlands.

Valerio Basile and Johan Bos. 2011. Towards generating text from discourse representation structures. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 145–150, Nancy, France. Association for Computational Linguistics.

Valerio Basile and Johan Bos. 2013. Aligning formal meaning representations with surface strings for wide-coverage text generation. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 1–9, Sofia, Bulgaria. Association for Computational Linguistics.

Valerio Basile, Johan Bos, Kilian Evang, and Noortje Venhuizen. 2012. Developing a large semantically annotated corpus. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3196–3200, Istanbul, Turkey. European Language Resources Association (ELRA).

Rachel Bawden, Rico Sennrich, Alexandra Birch, and Barry Haddow. 2018. Evaluating discourse phenomena in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1304–1313, New Orleans, Louisiana. Association for Computational Linguistics.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 273–283, Melbourne, Australia. Association for Computational Linguistics.

Johan Bos. 2008. Wide-coverage semantic analysis with Boxer. In *Semantics in Text Processing. STEP 2008 Conference Proceedings*, pages 277–286. College Publications.

Johan Bos. 2015. Open-domain semantic parsing with boxer. In *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*, pages 301–304, Vilnius, Lithuania. Linköping University Electronic Press, Sweden.

Johan Bos. 2016. Squib: Expressive power of Abstract Meaning Representations. *Computational Linguistics*, 42(3):527–535.

Johan Bos. 2020. Simplified discourse representation structures for natural language analysis. *Semantics Archive*.

Johan Bos, Valerio Basile, Kilian Evang, Noortje J. Venhuizen, and Johannes Bjerva. 2017. The groningen meaning bank. In Nancy Ide and James Pustejovsky, editors, *Handbook of Linguistic Annotation*, volume 2, pages 463–496. Springer.

Keith Brown. 2005. *Encyclopedia of language and linguistics*, volume 1. Elsevier.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Jan Buys and Phil Blunsom. 2017. Robust incremental neural semantic graph parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1215–1226, Vancouver, Canada. Association for Computational Linguistics.

Deng Cai and Wai Lam. 2019. Core semantic first: A top-down approach for AMR parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3799–3809, Hong Kong, China. Association for Computational Linguistics.

Deng Cai and Wai Lam. 2020a. AMR parsing via graph-sequence iterative inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1290–1301, Online. Association for Computational Linguistics.

Deng Cai and Wai Lam. 2020b. Graph transformer for graph-to-sequence learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7464–7471.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752, Sofia, Bulgaria. Association for Computational Linguistics.

John Carroll and Stephan Oepen. 2005. High efficiency realization for a wide-coverage unification grammar. In *Second International Joint Conference on Natural Language Processing: Full Papers*.

Danqi Chen. 2018. *Neural reading comprehension and beyond*. Ph.D. thesis, Standford University.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar. Association for Computational Linguistics.

Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2016. Neural sentence ordering. *arXiv preprint arXiv:1607.06952*.

Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2017. Learning structured natural language representations for semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 44–55, Vancouver, Canada. Association for Computational Linguistics.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Christos Christodouloupoulos and Mark Steedman. 2015. A massively parallel corpus: the bible in 100 languages. *Language resources and evaluation*, 49(2):375–395.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Shay B. Cohen, Dipanjan Das, and Noah A. Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 50–61, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Simon Colton, Jacob Goodwin, and Tony Veale. 2012. Full-face poetry generation. In *3rd International Conference on Computational Creativity, ICCC 2012*, pages 95–102. University College Dublin.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 2-3(3):281–332.

Baiyun Cui, Yingming Li, Ming Chen, and Zhongfei Zhang. 2018. Deep attentive sentence ordering network. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4340–4349, Brussels, Belgium. Association for Computational Linguistics.

James Curran, Stephen Clark, and Johan Bos. 2007. Linguistically motivated large-scale NLP with C&C and boxer. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 33–36, Prague, Czech Republic. Association for Computational Linguistics.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Marco Damonte and Shay B. Cohen. 2018. Cross-lingual Abstract Meaning Representation parsing. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1146–1155, New Orleans, Louisiana. Association for Computational Linguistics.

Marco Damonte and Shay B. Cohen. 2019. Structural neural encoders for AMR-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3649–3658, Minneapolis, Minnesota. Association for Computational Linguistics.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for Abstract Meaning Representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 536–546, Valencia, Spain. Association for Computational Linguistics.

Donald Davidson. 1967. The logical form of action sentences. *The logic of decision and action*, pages 81–95.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852.

Li Deng and Dong Yu. 2014. Deep learning: methods and applications. *Foundations and trends in signal processing*, 7(3–4):197–387.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742, Melbourne, Australia. Association for Computational Linguistics.

Zhendong Dong, Qiang Dong, and Changling Hao. 2010. HowNet and its computation of meaning. In *Coling 2010: Demonstrations*, pages 53–56, Beijing, China. Coling 2010 Organizing Committee.

Bonnie J. Dorr. 1994. Machine translation divergences: A formal description and proposed solution. *Computational Linguistics*, 20(4):597–633.

Bonnie J. Dorr, Eduard H. Hovy, and Lori S. Levin. 2004. Machine translation: Interlingual methods. *Brown K (ed) Encyclopedia of language and linguistics*.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics.

Kilian Evang. 2019. Transition-based DRS parsing using stack-LSTMs. In *Proceedings of the IWCS Shared Task on Semantic Parsing*, Gothenburg, Sweden. Association for Computational Linguistics.

Kilian Evang and Johan Bos. 2016. Cross-lingual learning of an open-domain semantic parser. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 579–588, Osaka, Japan. The COLING 2016 Organizing Committee.

Federico Fancellu, Sorcha Gilroy, Adam Lopez, and Mirella Lapata. 2019. Semantic graph parsing with recurrent neural network DAG grammars. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2769–2778, Hong Kong, China. Association for Computational Linguistics.

Christiane Fellbaum. 1998. *WordNet: An Electronic Database*. MIT Press, Cambridge, MA.

Francis Ferraro, Ting-Hao Huang, Stephanie Lukin, and Margaret Mitchell. 2019. Proceedings of the second workshop on storytelling. In *Proceedings of the Second Workshop on Storytelling*.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016a. CMU at SemEval-2016 task 8: Graph-based AMR parsing with infinite ramp loss. In

Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), pages 1202–1206, San Diego, California. Association for Computational Linguistics.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016b. Generation from Abstract Meaning Representation using tree transducers. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 731–739, San Diego, California. Association for Computational Linguistics.

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6(1):15–28.

William Foland and James H. Martin. 2017. Abstract Meaning Representation parsing using LSTM recurrent neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 463–472, Vancouver, Canada. Association for Computational Linguistics.

Qiankun Fu, Yue Zhang, Jiangming Liu, and Meishan Zhang. 2020. DRTS parsing with structure-aware encoding and decoding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6818–6828, Online. Association for Computational Linguistics.

Pascale Fung and Benfeng Chen. 2004. BiFrameNet: Bilingual frame semantics resource construction by cross-lingual induction. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 931–937, Geneva, Switzerland. COLING.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 273–280, Boston, Massachusetts, USA. Association for Computational Linguistics.

Aldo Gangemi, Valentina Presutti, Diego Reforgiato Recupero, Andrea Giovanni Nuzzolese, Francesco Draicchio, and Misael Mongiovì. 2017. Semantic Web Machine Reading with FRED. *Semantic Web*, 8(6):873–893.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 1–9, Prague. Association for Computational Linguistics.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288.

Alessandra Giordani and Alessandro Moschitti. 2009. Semantic mapping between natural language questions and SQL queries via syntactic pairing. In *International Conference on Application of Natural Language to Information Systems*, pages 207–221. Springer.

Jingjing Gong, Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2016. End-to-end neural sentence ordering using pointer network. *arXiv preprint arXiv:1611.04953*.

Naman Goyal and Jacob Eisenstein. 2016. A joint model of rhetorical discourse structure and summarization. In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 25–34, Austin, TX. Association for Computational Linguistics.

Pawan Goyal and R. Mahesh K. Sinha. 2009. Translation divergence in English-Sanskrit-Hindi language pairs. In *International Sanskrit Computational Linguistics Symposium*, pages 134–143. Springer.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Zhijiang Guo and Wei Lu. 2018. Better transition-based AMR parsing with a refined search space. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1712–1722, Brussels, Belgium. Association for Computational Linguistics.

Valerie Hajdik, Jan Buys, Michael Wayne Goodman, and Emily M. Bender. 2019. Neural text generation from rich semantic representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 2259–2266, Minneapolis, Minnesota.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. 2012. Announcing Prague Czech-English Dependency Treebank 2.0. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3153–3160, Istanbul, Turkey. European Language Resources Association (ELRA).

Birgit Hamp and Helmut Feldweg. 1997. GermaNet - a lexical-semantic net for German. In *Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*.

Momchil Hardalov, Ivan Koychev, and Preslav Nakov. 2020. Enriched pre-trained transformers for joint slot filling and intent detection. *arXiv preprint arXiv:2004.14848*.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Matic Horvat, Ann Copestake, and Bill Byrne. 2015. Hierarchical statistical semantic realization for minimal recursion semantics. In *Proceedings of the 11th International Conference on Computational Semantics (IWCS)*, pages 107–117, London, UK. Association for Computational Linguistics.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1587–1596.

Chu-Ren Huang, Shu-Kai Hsieh, Jia-Fei Hong, Yun-Zhu Chen, I-Li Su, Yong-Xiang Chen, and Sheng-Wei Huang. 2010. Chinese wordnet: Design, implementation, and application of an infrastructure for cross-lingual knowledge processing. *Journal of Chinese Information Processing*, 24(2):14–23.

Haoyang Huang, Yaobo Liang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, and Ming Zhou. 2019. Unicoder: A universal language encoder by pre-training with multiple cross-lingual tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2485–2494, Hong Kong, China.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

William John Hutchins and Harold L Somers. 1992. *An introduction to machine translation*, volume 362. Academic Press London.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural language engineering*, 11(3):311–325.

Angelina Ivanova, Stephan Oepen, Lilja Øvrelid, and Dan Flickinger. 2012a. Who did what to whom? a contrastive study of syntacto-semantic dependencies. In *Proceedings of the Sixth Linguistic Annotation Workshop*, pages 2–11, Jeju, Republic of Korea. Association for Computational Linguistics.

Angelina Ivanova, Stephan Oepen, Lilja Øvrelid, and Dan Flickinger. 2012b. Who did what to whom? a contrastive study of syntacto-semantic dependencies. In *Proceedings of the Sixth Linguistic Annotation Workshop*, pages 2–11, Jeju, Republic of Korea. Association for Computational Linguistics.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2016. Summarizing source code using a neural attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2073–2083, Berlin, Germany. Association for Computational Linguistics.

Katja Jasinskaja and Elena Karagjosova. 2015. Rhetorical relations. *The companion to semantics. Oxford: Wiley*.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany. Association for Computational Linguistics.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA. Association for Computational Linguistics.

Hans Kamp. 1981. A theory of truth and semantic representation. In *Formal Methods in the Study of Language*, volume 1, pages 277–322. Mathematisch Centrum, Amsterdam.

Hans Kamp and Uwe Reyle. 1993. *From Discourse to Logic; An Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Kluwer, Dordrecht.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137.

Lauri Karttunen. 1974. Presupposition and linguistic context. *Theoretical linguistics*, 1(1-3):181–194.

Shahnawaz Khan, Usama Mir, Salam S Shreem, and Sultan Alamri. 2018. Translation divergence patterns handling in english to urdu machine translation. *International Journal on Artificial Intelligence Tools*, 27(05):1850017.

Yunsu Kim, Duc Thanh Tran, and Hermann Ney. 2019. When and why is document-level context useful in neural machine translation? In *Proceedings of the Fourth Workshop on Discourse in Machine Translation (DiscoMT 2019)*, pages 24–34, Hong Kong, China. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, Banff, Canada.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2008. A large-scale classification of english verbs. *Language Resources and Evaluation*, 42(1):21–40.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

Tomáš Kočiský, Gábor Melis, Edward Grefenstette, Chris Dyer, Wang Ling, Phil Blunsom, and Karl Moritz Hermann. 2016. Semantic parsing with semi-supervised sequential autoencoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1078–1087, Austin, Texas. Association for Computational Linguistics.

Philipp Koehn. 2009. *Statistical machine translation*. Cambridge University Press.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 127–133.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157, Vancouver, Canada. Association for Computational Linguistics.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526, Copenhagen, Denmark. Association for Computational Linguistics.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Artur Kulmizev, Vinit Ravishankar, Mostafa Abdou, and Joakim Nivre. 2020. Do neural language models show preferences for syntactic formalisms? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4077–4091, Online. Association for Computational Linguistics.

Adhiguna Kuncoro, Lingpeng Kong, Daniel Fried, Dani Yogatama, Laura Rimell, Chris Dyer, and Phil Blunsom. 2020. Syntactic structure distillation pretraining for bidirectional encoders. *Transactions of the Association for Computational Linguistics*, 8:776–794.

Tom Kwiatkowksi, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 1223–1233.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1512–1523, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Mirella Lapata. 2003. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 545–552, Sapporo, Japan. Association for Computational Linguistics.

Phong Le and Willem Zuidema. 2012. Learning compositional semantics for open domain semantic parsing. In *Proceedings of COLING 2012*, pages 1535–1552, Mumbai, India. The COLING 2012 Organizing Committee.

Bin Li, Yuan Wen, Weiguang Qu, Lijun Bu, and Nianwen Xue. 2016a. Annotating the little prince with Chinese AMRs. In *Proceedings of the 10th Linguistic Annotation Workshop held in conjunction with ACL 2016 (LAW-X 2016)*, pages 7–15, Berlin, Germany. Association for Computational Linguistics.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020. A unified MRC framework for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5849–5859, Online. Association for Computational Linguistics.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2016b. Gated graph sequence neural networks. In *International Conference on Learning Representations*.

Percy Liang, Michael Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 590–599, Portland, Oregon, USA. Association for Computational Linguistics.

Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 599–609, Berlin, Germany. Association for Computational Linguistics.

Jiangming Liu, Shay B. Cohen, and Mirella Lapata. 2018a. Discourse representation structure parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 429–439, Melbourne, Australia. Association for Computational Linguistics.

Jiangming Liu, Shay B. Cohen, and Mirella Lapata. 2019a. Discourse representation parsing for sentences and documents. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6248–6262, Florence, Italy. Association for Computational Linguistics.

Jiangming Liu, Shay B. Cohen, and Mirella Lapata. 2020. Dscorer: A fast evaluation metric for discourse representation structure parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4547–4554, Online. Association for Computational Linguistics.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 609–616, Sydney, Australia. Association for Computational Linguistics.

Yang Liu, Ivan Titov, and Mirella Lapata. 2019b. Single document summarization as tree induction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1745–1755, Minneapolis, Minnesota. Association for Computational Linguistics.

Yijia Liu, Wanxiang Che, Bo Zheng, Bing Qin, and Ting Liu. 2018b. An AMR aligner tuned by transition-based parser. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2422–2430, Brussels, Belgium. Association for Computational Linguistics.

Lajanugen Logeswaran, Honglak Lee, and Dragomir Radev. 2018. Sentence ordering and coherence modeling using recurrent neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Wei Lu and Hwee Tou Ng. 2011. A probabilistic forest-to-string model for language generation from typed lambda calculus expressions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1611–1622, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Chunchuan Lyu and Ivan Titov. 2018. AMR parsing as graph prediction with latent alignment. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 397–407, Melbourne, Australia. Association for Computational Linguistics.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 62–72, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Yusuke Miyao, Stephan Oepen, and Daniel Zeman. 2014a. In-house: An ensemble of pre-existing off-the-shelf parsers. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 335–340, Dublin, Ireland. Association for Computational Linguistics.

Yusuke Miyao, Stephan Oepen, and Daniel Zeman. 2014b. In-house: An ensemble of pre-existing off-the-shelf parsers. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 335–340, Dublin, Ireland. Association for Computational Linguistics.

Makoto Nagao. 1984. A framework of a mechanical translation between japanese and english by analogy principle. *Artificial and human intelligence*, pages 351–354.

Shashi Narayan and Claire Gardent. 2014. Hybrid simplification using deep semantics and machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 435–445, Baltimore, Maryland. Association for Computational Linguistics.

Roberto Navigli and Simone Paolo Ponzetto. 2010. BabelNet: Building a very large multilingual semantic network. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 216–225, Uppsala, Sweden. Association for Computational Linguistics.

Rik van Noord, Lasha Abzianidze, Hessel Haagsma, and Johan Bos. 2018a. Evaluating scoped meaning representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Rik van Noord, Lasha Abzianidze, Antonio Toral, and Johan Bos. 2018b. Exploring neural methods for parsing discourse representation structures. *Transactions of the Association for Computational Linguistics*, 6:619–633.

Rik van Noord, Antonio Toral, and Johan Bos. 2019. Linguistic information in neural semantic parsing with multiple encoders. In *Proceedings of the 13th International Conference on Computational Semantics - Short Papers*, pages 24–31, Gothenburg, Sweden. Association for Computational Linguistics.

Stephan Oepen and Jan Tore Lønning. 2006a. Discriminant-based MRS banking. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

Stephan Oepen and Jan Tore Lønning. 2006b. Discriminant-based MRS banking. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

Sebastian Padó and Mirella Lapata. 2005. Cross-linguistic projection of role-semantic information. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 859–866, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Sebastian Padó and Mirella Lapata. 2009. Cross-lingual annotation projection for semantic roles. *Journal of Artificial Intelligence Research*, 36:307–340.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Terence Parsons. 1990. Events in the semantics of english: A study in subatomic semantics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Carl Pollard and Ivan A. Sag. 1994. *Head-driven phrase structure grammar*. University of Chicago Press.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse TreeBank 2.0. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Ella Rabinovich, Noam Ordan, and Shuly Wintner. 2017a. Found in translation: Reconstructing phylogenetic language trees from translations. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 530–540, Vancouver, Canada. Association for Computational Linguistics.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017b. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1139–1149, Vancouver, Canada. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392.

Leonardo F. R. Ribeiro, Claire Gardent, and Iryna Gurevych. 2019. Enhancing AMR-to-text generation with dual graph representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3183–3194, Hong Kong, China. Association for Computational Linguistics.

Richard H Richens. 1958. Interlingual machine translation. *The Computer Journal*, 1(3):144–147.

Rob A. Van der Sandt. 1992. Presupposition projection as anaphora resolution. *Journal of semantics*, 9(4):333–377.

Karin Kipper Schuler. 2005. *VerbNet: A broad-coverage, comprehensive verb lexicon*. Ph.D. thesis, University of Pennsylvania.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. 2018. Structured sequence modeling with graph convolutional recurrent networks. In *Neural Information Processing Systems (NeurIPS)*, pages 362–373, Montréal, Canada. Springer.

Petr Sgall, Eva Hajicová, Eva Hajicová, Jarmila Panevová, and Jarmila Panevova. 1986. *The meaning of the sentence in its semantic and pragmatic aspects*. Springer Science & Business Media.

Stuart M. Shieber, Gertjan van Noord, Fernando C. N. Pereira, and Robert C. Moore. 1990. Semantic-head-driven generation. *Computational Linguistics*, 16(1):30–42.

Karin Sim Smith. 2017. On integrating discourse in machine translation. In *Proceedings of the Third Workshop on Discourse in Machine Translation*, pages 110–121, Copenhagen, Denmark. Association for Computational Linguistics.

Anders Søgaard. 2011. Data point selection for cross-language adaptation of dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 682–686, Portland, Oregon, USA. Association for Computational Linguistics.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for AMR-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1616–1626, Melbourne, Australia. Association for Computational Linguistics.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Mark Steedman. 2001. *The Syntactic Process*. The MIT Press.

Mark Stevenson and Yorick Wilks. 2003. Word sense disambiguation. *The Oxford handbook of computational linguistics*, pages 249–265.

Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gür, Zenghui Yan, and Xifeng Yan. 2016. On generating characteristic-rich question sets for QA evaluation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 562–572, Austin, Texas. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112. Curran Associates, Inc.

Ida Szubert, Marco Damonte, Shay B. Cohen, and Mark Steedman. 2020. The role of reentrancies in Abstract Meaning Representation parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2198–2207, Online. Association for Computational Linguistics.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China. Association for Computational Linguistics.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on Abstract Meaning Representation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1054–1059, Austin, Texas. Association for Computational Linguistics.

Lappoon R. Tang and Raymond J. Mooney. 2000. Automated construction of database interfaces: Intergrating statistical and relational learning for semantic parsing. In *2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 133–141, Hong Kong, China. Association for Computational Linguistics.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2214–2218, Istanbul, Turkey. European Language Resources Association (ELRA).

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–85, Berlin, Germany. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Bernard Vauquois. 1968. A survey of formal grammars and algorithms for recognition and transformation in mechanical translation. In *IFIP Congress*.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Noortje J. Venhuizen, Johan Bos, and Harm Brouwer. 2013. Parsimonious semantic representations with projection pointers. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers*, pages 252–263, Potsdam, Germany. Association for Computational Linguistics.

Noortje J. Venhuizen, Johan Bos, Petra Hendriks, and Harm Brouwer. 2018. Discourse semantics with information structure. *Journal of Semantics*, 35(1):127–169.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Neural Information Processing Systems (NeurIPS)*, pages 2692–2700, Montréal, Canada.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *Proceedings of the 31st International Conference on Machine Learning*, volume 37.

Juen-tin Wang. 1980. On computational sentence generation from logical form. In *Proceedings of the 8th International Conference on Computational Linguistics (COLING)*, pages 405–411, Tokyo, Japan.

Mengqiu Wang and Christopher D. Manning. 2014. Cross-lingual projected expectation regularization for weakly supervised learning. *Transactions of the Association for Computational Linguistics*, 2:55–66.

Qingyun Wang, Xiaoman Pan, Lifu Huang, Boliang Zhang, Zhiying Jiang, Heng Ji, and Kevin Knight. 2018. Describing a knowledge base. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 10–21, Tilburg University, The Netherlands. Association for Computational Linguistics.

Tianming Wang, Xiaojun Wan, and Hanqi Jin. 2020. AMR-to-text generation with graph transformer. *Transactions of the Association for Computational Linguistics*, 8:19–33.

Yaushian Wang, Hung-Yi Lee, and Yun-Nung Chen. 2019. Tree transformer: Integrating tree structures into self-attention. In *Proceedings of the 2019 Conference on*

Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 1061–1070, Hong Kong, China. Association for Computational Linguistics.

Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2016. Universal decompositional semantics on Universal Dependencies. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1713–1723, Austin, Texas. Association for Computational Linguistics.

Michael White, Rajakrishnan Rajkumar, and Scott Martin. 2007. Towards broad coverage surface realization with CCG. In *Proceedings of the Workshop on Using Corpora for NLG: Language Generation and Machine Translation (UCNLG+MT)*, pages 267–276, Copenhagen, Denmark.

Yuk Wah Wong and Raymond Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 439–446, New York City, USA. Association for Computational Linguistics.

Yuk Wah Wong and Raymond Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 960–967, Prague, Czech Republic. Association for Computational Linguistics.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Shanchan Wu and Yifan He. 2019. Enriching pre-trained language model with entity information for relation classification. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2361–2364. ACM.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S. Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*.

Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Proceedings of the 54th Annual Meeting*

of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1341–1350, Berlin, Germany. Association for Computational Linguistics.

Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, and Vadim Sheinin. 2018. SQL-to-text generation with graph-to-sequence model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 931–936, Brussels, Belgium. Association for Computational Linguistics.

Yinfei Yang, Yuan Zhang, Chris Tar, and Jason Baldridge. 2019. PAWS-x: A cross-lingual adversarial dataset for paraphrase identification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3687–3692, Hong Kong, China.

Xuchen Yao, Gosse Bouma, and Yi Zhang. 2012. Semantics-based question generation and implementation. *Dialogue and Discourse*, 2(3):11–42.

David Yarowsky and Grace Ngai. 2001. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Second Meeting of the North American Chapter of the Association for Computational Linguistics*.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China. Association for Computational Linguistics.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450, Vancouver, Canada. Association for Computational Linguistics.

Yongjing Yin, Linfeng Song, Jinsong Su, Jiali Zeng, Chulun Zhou, and Jiebo Luo. 2019. Graph-based neural sentence ordering. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 5387–5393. AAAI Press.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018a. Qanet: Combining local convolution with global

self-attention for reading comprehension. In *International Conference on Learning Representations*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, and Shanelle Roman. 2018b. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921.

Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. 2019. Graph transformer networks. In *Advances in Neural Information Processing Systems*, pages 11983–11993.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the 13th National Conference on Artificial Intelligence*, pages 1050–1055, Portland, Oregon.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *PProceedings of the 21st Conference in Uncertainty in Artificial Intelligence*, pages 658–666, Edinburgh, Scotland, UK.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019. AMR parsing as sequence-to-graph transduction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 80–94, Florence, Italy. Association for Computational Linguistics.

Sheng Zhang, Xutai Ma, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme. 2018. Cross-lingual decompositional semantic parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1664–1675, Brussels, Belgium. Association for Computational Linguistics.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680, Doha, Qatar. Association for Computational Linguistics.

Xingxing Zhang, Liang Lu, and Mirella Lapata. 2016. Top-down tree long short-term memory networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language*

Technologies, pages 310–320, San Diego, California. Association for Computational Linguistics.

Jingyu Zhao, Feiqing Huang, Jia Lv, Yanjie Duan, Zhen Qin, Guodong Li, and Guangjian Tian. 2020. Do RNN and LSTM have long memory? In *Proceedings of the 37th International Conference on Machine Learning*.

Kai Zhao and Liang Huang. 2015. Type-driven incremental semantic parsing with polymorphism. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1416–1421, Denver, Colorado. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

Jie Zhu, Junhui Li, Muhua Zhu, Longhua Qian, Min Zhang, and Guodong Zhou. 2019. Modeling graph structure in transformer for better AMR-to-text generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5459–5468, Hong Kong, China. Association for Computational Linguistics.

Bowei Zou, Guodong Zhou, and Qiaoming Zhu. 2014. Negation focus identification with contextual discourse information. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 522–530, Baltimore, Maryland. Association for Computational Linguistics.