THE UNIVERSITY
of EDINBURGH

Graph-based Broad-coverage Semantic Parsing

Chunchuan Lyu

Doctor of Philosophy

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

2021

Abstract

Many broad-coverage meaning representations can be characterized as directed graphs, where nodes represent semantic concepts and directed edges represent semantic relations among the concepts. The task of semantic parsing is to generate such a meaning representation from a sentence. It is quite natural to adopt a graph-based approach for parsing, where nodes are identified conditioning on the individual words, and edges are labeled conditioning on the pairs of nodes. However, there are two issues with applying this simple and interpretable graph-based approach for semantic parsing: first, the anchoring of nodes to words can be implicit and non-injective in several formalisms (Oepen et al., 2019, 2020). This means we do not know which nodes should be generated from which individual word and how many of them. Consequently, it makes a probabilistic formulation of the training objective problematical; second, graph-based parsers typically predict edge labels independent from each other. Such an independence assumption, while being sensible from an algorithmic point of view, could limit the expressiveness of statistical modeling. Consequently, it might fail to capture the true distribution of semantic graphs.

In this thesis, instead of a pipeline approach to obtain the anchoring, we propose to model the implicit anchoring as a latent variable in a probabilistic model. We induce such a latent variable jointly with the graph-based parser in an end-to-end differentiable training. In particular, we test our method on Abstract Meaning Representation (AMR) parsing (Banarescu et al., 2013). AMR represents sentence meaning with a directed acyclic graph, where the anchoring of nodes to words is implicit and could be many-to-one. Initially, we propose a rule-based system that circumvents the many-to-one anchoring by combing nodes in some pre-specified subgraphs in AMR and treats the alignment as a latent variable. Next, we remove the need for such a rule-based system by treating both graph segmentation and alignment as latent variables. Still, our graph-based parsers are parameterized by neural modules that require gradient-based optimization. Consequently, training graph-based parsers with our discrete latent variables can be challenging. By combing deep variational inference and differentiable sampling, our models can be trained end-to-end. To overcome the limitation of graph-based parsing and capture interdependency in the output, we further adopt iterative refinement. Starting with an output whose parts are independently predicted, we iteratively refine it conditioning on the previous prediction. We test this method on semantic role labeling (Gildea and Jurafsky, 2000). Semantic role labeling is the task of predicting the predicate-argument structure. In particular, semantic roles between

the predicate and its arguments need to be labeled, and those semantic roles are interdependent. Overall, our refinement strategy results in an effective model, outperforming strong factorized baseline models.

Lay Summary

Nowadays, many computing systems interact with users through natural language in daily life. Digital assistants (e.g., Alexa, Siri, Cortina) can tell you the weather when you ask *what's the weather today?* Equipped with smart home devices, turning on the lights while lying on the bed can be accomplished by yelling *Alexa, turn on the light*. When calling a customer service center, a dialogue system helps the customer with its goal or direct the customer to the relevant staff. When we type *what is covid?* into QA systems such as WolframAlpha, the system directly provides all the relevant information in structured tables, instead of returning a webpage. These applications are built on top of semantic parsing that converts natural language into machine-interpretable symbolic forms.

This thesis investigates a particular approach of semantic parsing called graph-based parsing. Graph-based parsers are very simple to apply. It identifies concepts in a natural sentence and then figures out their relations. For example, in *Alexa, turn on the light*. *turn on* is a special concept that signifies a command, and both *Alexa* and *light* are two other concepts. Once they are identified, a graph-based parser then decide their relations: *Alexa* is the subject of *turn on* and *light* is the object. Therefore, Alexa knows that it should turn on the light. Such simple parsers should be quite reliable.

However, for more complicated symbolic forms, training a graph-based parser from annotated datasets can be non-trivial and involves many hand-crafted pipelines. Furthermore, a graph-based parser can indeed be too simple to capture all the information in a sentence. This thesis gets rid of pipelines for training graph-based parsers and makes graph-based parsers more powerful.

Acknowledgements

My Ph.D. journey started before my principal supervisor Ivan Titov came to Edinburgh. Before Ivan came, I had been adopted by Shay B. Cohen, who has been supervising me since my MSc year. Since then, I have had the fortune of getting their unconditional and unlimited supports. I am grateful for their insights, patience, and encouragement.

I am also grateful to Wilker Aziz for a discussion during my first year. Otherwise, the way that my AMR parsers got trained could be very different. I thank Hari Parthasarathi for hosting my internship at Amazon in a beautiful summer. I also thank Ramón Fernández Astudillo for hosting my internship at IBM. It is quite a fun experience working with so many people believing in AMR. I am fortunate to study in the EdinburghNLP group with a large number of researchers and students. The discussions are always stimulating. In particular, I thank Adam Lopez and Mark Steedman for feedbacks on my yearly reviews. A super thank you to Matthias Lindemann for proofreading this thesis. Special thanks to Mark Steedman and Daniel Gildea for examining my thesis and providing constructive feedback.

Of course, my Ph.D. years will be very miserable without friends and fellow students. I thank David, Michael, and Xinchi for playing badminton with me. T thank Bo and Bailin for taking me skiing. Many thanks to many friends and fellow students that I met at Edinburgh: Shucong, Zhunxuan, Yanzhi, Yanpeng, Biao, Hao, Zhijiang, Bowen, Mo, Kai, Yang, Xingxing, Li, Hang, Zhifeng, Jianpeng, Esma, Nikos, Jonana, Marco, Shashi, Zheng, Serhii, Arthur, Nicola, Lena, Matthias, Caio, Diego, Matthieu, Maximin, Yao, Shangmin, etc. Special thanks to my flatmate Jiangming for being trapped together due to the Covid.

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.

chunchuan lv

(*Chunchuan Lyu*)

To my parents,
who always forgive my mischief
and forever support my adventures.

Table of Contents

Chapter 1

Introduction

To make computers understand natural language, we arguably need meaning representations that are interpretable to computers. Meaning representations are structured data that capture the meaning in natural language, and can be categorized into two groups: executable meaning representations and broad-coverage meaning representations. Executable meaning representations (Liang, 2016; Cheng et al., 2019) cover specific domains and are executable in the sense that they can be used by the computer to complete certain tasks such as question answering (Liang et al., 2011), and human-computer dialogue (Gupta et al., 2018). In particular, executable meaning representations are widely used in modern digital assistant systems such as Siri, Google Assistant, Alexa, and Cortana. For example, in Google Assistant, "set an alarm for 7 am" will be converted to a domain-specific meaning representation consisting of intent and slot values {Intent: AlarmClock, SLOTS: {EXTRA_HOUR : 7}}. Once the intent and slot values are recognized, the system calls the alarm application API to set the alarm for 7 a.m.

While executable meaning representations have been successful, they require domain-specific environments to be defined. The intent and slot values will not be interpretable without the existence of an alarm app. Yet, meaning representations do not have to be readily executable. In general, natural language meaning can be abstract and is not always grounded in the actual world (e.g., a unicorn). Leaving aside the immediate executability, many have been pursuing broad-coverage meaning representations that aim at representing meaning across broader domains of human language in a unified framework (Baker et al., 1998; Palmer et al., 2005; Ivanova et al., 2012; Hajič et al., 2012; Oepen and Lønning, 2006; Basile et al., 2012; Abend and Rappoport, 2013; Banarescu et al., 2013; Abzianidze et al., 2017). Such pursuits result in many different frame-

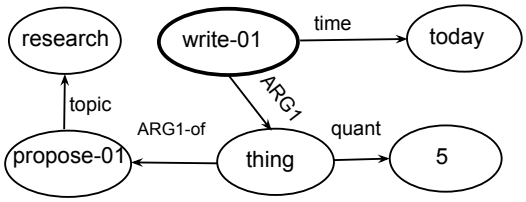


Figure 1.1: Example of Abstract Meaning Representation.

works of broad-coverage meaning representation. Nonetheless, many broad-coverage meaning representations can be formulated as labeled directed graphs (Oepen et al., 2019, 2020). The nodes in the graph are concepts, and edges are semantic relations between them. In particular, as shown in Figure 1.1, the meaning of "write five research proposals today" can be represented as a directed graph with labeled nodes and edges in Abstract Meaning Representation (AMR; Banarescu et al. (2013)), one of the popular meaning representation frameworks.

This thesis is about the *broad-coverage semantic parsing*: the task of mapping a natural language sentence into a broad-coverage meaning representation. We aim at developing general techniques for semantic parsing. As many broad-coverage meaning representations are graphs, we adopt the *graph-based* approach to semantic parsing. A graph-based parser factorizes the prediction of the graph into the predictions of its parts, and has two stages: the concept identification stage and the relation identification stage. In the concept identification stage, the parser scans through the words and produces nodes if the parser identifies a concept (e.g., 'write-01' will be produced when scanning the word 'write'). In the relation identification stage, the parser identifies edges and labels them for all possible pairs of nodes independently. Graph-based parsing is appealing for its simplicity and interpretability. It is simple, as the parser directly produces the graph without relying on auxiliary constructions. It is interpretable because the correspondence between the resulting graph and sentence is always explicit.

However, as we can see in Figure 1.1, the anchoring from nodes to tokens is implicit, e.g., 'write-01' should be anchored to the word 'write', but the anchoring is not given as part of the meaning representation. Such information is not annotated, and training a graph-based parser is impossible without the anchoring. Incorporating the implicit anchoring into an end-to-end training system is particularly challenging, because training neural modules in current graph-based parsers requires gradient-based

optimization. Additionally, factorizing the prediction implies the factorization of scoring over the graph into the summation of the scores over the node and edge predictions. Consequently, such a factorization limits the expressiveness of statistical modeling, as the prediction of nodes and edges are made conditionally independent.[1] Yet, there are underlying interdependencies between the parts of a meaning representation.

The graph generation problems that we study in graph-based semantic parsing is not limited to semantic parsing. Beyond the scope of semantic parsing, graph generation is ubiquitous in many application areas. In computer vision, scene graph generation involves predicting a scene graph for a given image (Xu et al., 2017) for better visual scene understanding. In chemistry, molecular graph generation predicts molecules with designed property for better drug desgin (Jin et al., 2018). In biology, predicting gene interaction graph facilitates inferring functional patterns of genes (Stark et al., 2006; Kishan et al., 2019).

Thesis Statement This thesis makes the training of a graph-based parser end-to-end differentiable and increases its expressiveness: when the anchoring of a graph to the sentence is unknown, the graph-based approach can be trained end-to-end by inducing the anchoring as a latent variable; while the factorized prediction in graph-based parsing results in limited expressiveness in terms of statistical modeling, statistical interdependency can be captured by further iteratively refining the prediction.

1.1 Thesis Overview

This thesis develops general techniques for broad-coverage semantic parsing. However, we focus on abstract meaning representation parsing and semantic role labeling (SRL). Semantic role labeling is the task of predicting the predicate-argument structure (Gildea and Jurafsky, 2000), which involves edge label prediction (e.g., semantic role 'ARG1' in Figure 1.1) between the predicate (e.g., 'write-01') and its arguments (e.g., 'thing'). We will discuss both AMR and SRL in more detail in the next chapter. We pick AMR parsing for its complexity, stretching the applicability of our end-to-end differentiable training framework. We choose semantic role labeling for its structural simplicity, making it straightforward to perform an investigation of the effectiveness of modeling interdependency between parts.

[1] This is mostly true, but one can not have edges without nodes.

1.1.1 End-to-end Training

Oepen et al. (2019, 2020) represent several broad-coverage meaning representations as graphs, and classified them into three flavors in terms of the anchoring of nodes to words: in Flavor (0), anchoring is explicit and injective (i.e., all nodes are aligned to words, and their anchoring does not overlap); in Flavor (1), anchoring might be implicit for some nodes and non-injective; in Flavor (2), anchoring is implicit and non-injective. AMR belongs to the last flavor, which is the most challenging type from the parsing perspective. Both properties make the training of a graph-based parser problematic, as they prevent us from casting concept prediction as a sequence tagging task. The lack of explicit anchoring means that we do not know which word to condition on when predicting any particular node. The non-injectivity of anchoring is also problematic, as it forces one token to generate several nodes. For example, in our Figure 1.1, the word 'proposal' is supposed to generate the two nodes 'propose-01' and 'thing'. We address both issues with the AMR anchoring.

Latent Alignment We address the implicit anchoring without an external aligner. We assume the anchoring is injective and refer to the injective anchoring as alignment. We propose to model the alignment as a latent variable in a joint probabilistic model for alignment, concept, and relation identification. The model can be trained end-to-end. The assumed injectivity is enforced by a hand-crafted rule-based re-categorization system that collapse nodes like 'propose-01' and 'thing' into one node for the concept identification. This rule-based system is based on a close examination of individual phenomena AMR captures. Experimentally, we show that joint modeling is preferable to using a pipeline of align and parse. We significantly improved the state-of-the-art AMR parsing at the time.

Latent Segmentation and Alignment Such rule-based re-categorization requires expert domain knowledge about the specific meaning representation. This requirement of the domain knowledge makes the adaptation to other formalisms difficult. We further attempt to get rid of such a system, addressing the non-injectivity of anchoring. We recognize that producing multiple nodes from one word can be realized by a local auto-regressive model that produces nodes in a subgraph associated with the word (e.g., the method may associate both 'propose-01' and 'thing' with word 'proposals' in Figure 1.1). Furthermore, as graph segmentation is implicit, we treat both segmentation and alignment as the latent variables in a joint probabilistic model. Again, the resulting model can be trained end-to-end. Experimentally, we observe that inducing

segmentation yields substantial gains over using a naive 'greedy' segmentation heuristic. The performance of our method also approaches that of a model that relies on our hand-crafted re-categorization system.

1.1.2 Beyond Factorized Prediction

The graph-based parser makes factorized predictions of the parts of the meaning representation graph. In graph-based parsing, the identification of an individual concept is conditioned on the anchored word, and the identification of an individual semantic relation is conditioned on the pair of concepts. Those identifications are modeled as conditionally independent of each others. Relying on powerful sentence encoders, such models can be very expressive. However, before the rise of deep learning methods, the most accurate structured prediction methods relied on modeling high-order interactions in the output space. In particular, in semantic role labeling, modeling high-order interdependency between semantic roles was common before the rise of the expressive sentence encoders (Watanabe et al., 2010; Toutanova et al., 2008). Yet, many earlier approaches can not be easily adapted to the deep learning system. We model interactions between argument labeling decisions through *iterative refinement*. Starting with an output produced by a factorized model, we iteratively refine it using a purposefully designed task-specific network. Experimentally, we considered CoNLL 2009 shared task. We outperformed previous best results on five out of seven languages.

1.2 Thesis Structure

The thesis is structured as follows:

- Chapter 2 reviews previous work on SRL and AMR, including applications and existing parsing approaches. We also review other broad-coverage meaning representation formalisms.

- Chapter 3 provides technical backgrounds on the differentiable sampling of the latent structured discrete variable. Differentiable sampling is a recent technique which we heavily rely on to model the latent alignment and subgraph segmentation in AMR parsing.

- Chapter 4 introduces our first graph-based AMR parser. We devise a rule-based re-categorization system to segment the AMR graph into a re-categorized AMR

graph. With the help of the re-categorization system, our parser is trained with a joint probabilistic model for alignment, concept, and relation identification. This chapter is based on Lyu and Titov (2018).

- Chapter 5 replaces the rule-based re-categorization system introduced in Chapter 4 by modeling the subgraph segmentation as another latent variable in a joint probabilistic model. This chapter has been submitted to a conference as a paper (Lyu et al., 2020).

- Chapter 6 introduces *iterative refinement* that models interactions between argument labeling decisions. Overall, our refinement strategy results in an effective model, outperforming strong factorized baseline models. Additionally, our refinement improves the baseline prediction in the out-of-domain setting, avoiding overfitting. This chapter is based on Lyu et al. (2019).

- Chapter 7 summarizes the findings and discusses potential future work.

Chapter 2

Broad-coverage Meaning Representations

In this chapter, we introduce broad-coverage meaning representations and discuss how they can be represented and parsed as graphs. Our discussion will be limited to sentence-level meaning representations. In particular, we focus on *Semantic Roles Labeling* (SRL)[1] and *Abstract Meaning Representation* (AMR), for which we will build parsers. Roughly speaking, both semantic roles and AMR represent "Who did what to whom and how, when and where?" in a sentence (Palmer et al., 2010; Banarescu et al., 2019) as semantic dependency between semantic concepts. Those semantic concepts and semantic dependencies can be regarded as nodes and edges in a graph. Focusing on semantic parsing, we review recent approaches to semantic parsing. In particular, we focus on graph-based approaches. A graph-based parser performs parsing by first tagging the words to predict the nodes, then predicting edges between the nodes.

The remainder of this chapter is organized as following: first, we introduce semantic role labeling and AMR parsing. Next, we briefly review some other broad-coverage meaning representations. It turns out that many of them can be represented as graphs as was done in Oepen et al. (2019, 2020). Then, we discuss applications of SRL and AMR. Finally, we introduce graph-based parsing and other parsing approaches. In particular, we discuss why it is difficult to apply the graph-based approach for AMR parsing and what can be regarded as limitations of graph-based parsing. Naturally, this thesis tries to address those issues.

[1] Technically speaking, SRL is not a representation but the task of predicting the predicate-argument structure.

2.1 Semantic Role Labeling

To capture "who did what to whom and how, when, and where" in a sentence, we need to identify the events and their participants with associated roles, i.e., the predicate-argument structure. A sentence can contain multiple predicates and their corresponding arguments. Each argument corresponds to a syntactic constituent and is assigned a semantic role from the role set defined for that predicate. As a task, semantic role labeling, originally introduced by Gildea and Jurafsky (2000), involves the prediction of semantic roles given the frame and sentence, i.e., identification of arguments and their assignment to semantic roles.

There are two major annotated datasets for semantic role labeling for English: FrameNet (Baker et al., 1998; Ruppenhofer et al., 2006), and the Proposition bank (PropBank) (Palmer et al., 2005). FrameNet is primarily semantics-driven and focuses on developing and annotating fine-grained lexical meaning (Palmer and Sporleder, 2010). Focusing on annotating training data for building an automatic parser, the PropBank annotated the predicate-argument structure for verbs with generic semantic roles. In addition, PropBank contains a set of frame files for each predicate that describe the set of possible semantic roles that can be associated with the predicate. We focus on PropBank.

There were several SRL shared tasks which derived all or some of their data from PropBank: CoNLL-2004, CoNLL-2005, CoNLL-2008 and CoNLL-2009 (Carreras and Màrquez, 2004, 2005; Surdeanu et al., 2008; Hajič et al., 2009).[2] In particular, the latter two shared tasks CoNLL-2008 and CoNLL-2009 use a dependency-based representation of predicate-argument structure (Surdeanu et al., 2008; Hajič et al., 2009), where the semantic roles are assigned to the syntactic head of the argument constituents. We focus on the dependency-based representation of the predicate-argument structure, in particular CoNLL-2009, which includes 7 languages (i.e., Catalan, Chinese, Czech, English, German, Japanese and Spanish).

This predicate-argument structure forms a graph, where the nodes are predicates and their arguments. The graph edges are semantic roles. While the graph formalization is not apparently useful for semantic role labeling per se, it helps the application of the predicate-argument structure for other tasks (see Section 2.4), and making SRL a subset of AMR as we shall see in Section 2.2.

[2] In addition to PropBank, NomBank (Meyers et al., 2004) annotated the predicate-argument structure for nouns. CoNLL-2008 and CoNLL-2009 combine PropBank and NomBank.

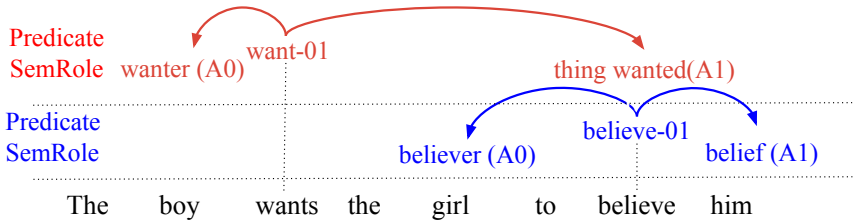


Figure 2.1: Example of Semantic Role Labeling.

As shown in Figure 2.1,[3] the sentence has two predicates *'want'* and *'believe'*. They are further annotated with word senses 'want-01' and 'believe-01', respectively. Let us focus on the predicate 'believe-01'. The argument *'girl'* takes the semantic role of Believer, and *'him'* takes the semantic role of Belief. The set of core semantic roles (i.e., Believer and Belief) are specific to 'believe-01'. Those core semantic roles are subsequently numbered as A0 and A1 in the frame file. The numbered arguments A0 and A1 tend to correspond to the prototypical agent (Dowty, 1991) and the prototypical patient, respectively. In general, there could be at most six arguments for a predicate. A2 to A5 can denote other thematic roles (e.g., beneficiary, goal, source, extent and cause). However, they are not very consistent across predicates (Palmer et al., 2010). In addition, the frame file contains all possible predicate senses (e.g., believe contains only 'believe-01'). The predicate can only take a sense from this sense inventory. In principle, the core semantic roles are restricted to the numbered arguments given the predicate sense. e.g., 'believe-01' has only two arguments, so labeling any token as A2 is illegitimate. Meanwhile, missing arguments are allowed, as arguments can be optional. In addition to those core semantic roles, there are other modifier roles AM-M. e.g., AM-LOC specifies location of the event. The corpus is annotated with whether each word is a predicate, and the sense of the predicate. For each predicate, their arguments are labeled with semantic roles, where the core semantic roles are labeled as A0 to A5.

Concerning the evaluation of the shared tasks, the CoNLL 2008/2009 semantic role labeling shared tasks involve both the identification/disambiguation of the predicate senses and the original semantic role labeling (Surdeanu et al., 2008; Hajič et al., 2009).[4] To make the task simpler, the predicate words (e.g., 'wants' and 'believe') are usually given for benchmarking the performance. This effectively converts the task

[3]This sentence is from AMR guidelines (Banarescu et al., 2017).

[4]The CoNLL-2008 and CoNLL-2009 shared tasks include the predicate identification in the evaluation. Yet, the tasks are still commonly referred as semantic role labeling.

into a classification problem and a sequence labeling problem (i.e., assigning a semantic role to each word, including not being an argument role). The evaluation metric for CoNLL-2009 is the macro-averaged F1 score for both predicate sense disambiguation and semantic role labeling, given the predicate words. It is worth noting that PropBank has 95% inter-annotator agreement (IAA) on classifying semantic roles given the constituents (Palmer et al., 2005), and NomBank has 85% IAA (Meyers et al., 2004). In Chapter 6, we will work on the CoNLL-2009 shared task, where multi-lingual resources are available. This provides a simple yet extensive testing bed.

Concerning the coverage of semantic phenomena, there are clear limitations of the predicate-argument structure even as a 'shallow' meaning representation. It does not capture many semantic phenomena such as co-reference (e.g., 'boy' and 'him' refers to the same person in Figure 2.1), and named entities. A new annotation of the predicate-argument structure QA-SRL has been pursued (He et al., 2015; Klein et al., 2020) with additional semantic roles for question answering. Aside from QA-SRL, the OntoNotes project (Pradhan et al., 2007; Strassel et al., 2011) annotates co-reference and named entities. However, the OntoNotes project does not provide a unified meaning representation to capture all those phenomena. In the next section, we introduce abstract meaning representation that captures many distinct semantic phenomena in a unified graph format.

2.2 Abstract Meaning Representation

Abstract Meaning Representation (AMR; Banarescu et al. (2013)) is a broad-coverage sentence-level meaning representation. AMR encodes, among others, information about the predicate-argument structure, named entities, co-reference, negation, and modality. The meaning representation can be regarded as a rooted labeled directed acyclic graph, where the nodes represent concepts and edges represent semantic relations (e.g., semantic role). We will examine individual semantic phenomena encoded in AMR and some recent developments on AMR sembanking.

Figure 2.2 is an AMR annotation of the same example as in Section 2.1. One can immediately notice that AMR builds on the same predicate frames with PropBank.[5] Furthermore, AMR explicitly shows that the predicate 'believe-01' assigns the se-

[5]However, AMR does not use NomBank. AMR chose to verbalize nouns, as we will see in Section 2.2.1

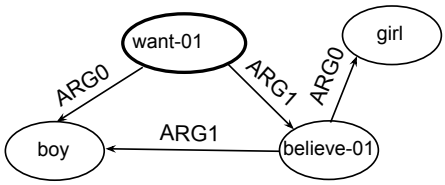


Figure 2.2: Example of Abstract Meaning Representation.

mantic role belief (as annotated by ARG1)[6] to the 'boy' instead of 'him'. Therefore, co-reference is encoded as reentrancy (node with more than one incoming edge). Another piece of information this graph encodes is the focus of the sentence, which is the root node 'want-01' (as annotated by bolder border). Another crucial difference between AMR and SRL is that AMR concepts are not explicitly aligned to individual words. This facilitates the annotation of corpus and encourages varieties in approaches for semantic parsing (Banarescu et al., 2013). Importantly, abstract meaning representation abstracts away from the surface realization. The graph will be the same for the sentence "The boy wants to be believed by the girl".

While AMR is a rooted labeled directed acyclic graph in essence, it has two other representations. The text-friendly representation (PENMAN notation; Kasper (1989)) of our running example is as follows.

```
(w / want-01
    :ARG0 (b / boy)
    :ARG1 (b2 / believe-01
              :ARG0 (g / girl)
              :ARG1 b))
```

Each distinct concept node is replaced by a unique variable, and the concept node lemma is introduced by "/" following the variable. When a variable appears multiple times, all occurrences denote the same concept. The variable name itself (e.g., w for 'want-01') does not carry meaning. The text representation is actually what is being used in the corpus annotation. Alternatively, AMR can be represented as logic triples (Banarescu et al., 2017):

[6] In AMR, ARGx is used instead of Ax as in SRL by convention.

```
Root(w,w) ^
instance(w, want-01) ^
instance(b, boy) ^
instance(b2, believe-01) ^
instance(g, girl) ^
ARG0(w, b) ^
ARG1(w, b2) ^
ARG0(b2, g) ^
ARG1(b2, b)
```

We added a self edge Root(w,w) to denote the focus. This is useful for defining evaluation algorithms of AMR parsing. In addition, in the logic triple language, ARG0(w, b) is the same as ARG0-of(b, w). This equivalence is important for evaluating AMR parsing.

Evaluating the performance of the parser requires comparing two unaligned graphs. It is more convenient to define the score in terms of logical triples. Smatch score (Cai and Knight, 2013) is defined as the F1 score of logical triple overlap between two AMRs. However, as variable names could differ between two AMRs, an integer programming technique is required to approximate the best match. To compare multiple predicated AMRs and the gold AMRs, the macro-averaged F1 is used. The IAA in terms of Smatch score is 0.83 for newswire data and 0.80 for web data. This IAA is significantly lower than that of SRL. Indeed, the low IAA is considered as a weakness of the current AMR sembank. In addition to the aggregated evaluation metric, Damonte et al. (2017) developed AMR-evaluation tools that are based on Smatch but provide fine-grained analysis of the parser performance (e.g., SRL, reentrancy).

So far, AMR appears to be simple. Now we turn to the intricate part of AMR.

2.2.1 Individual Phenomena

We examine some individual semantic phenomena that are captured in AMRs. To have a more thorough picture, one should refer to the AMR guidelines (Banarescu et al., 2017), from which we draw most of our examples. Our close examination is to illustrate the difficulty of parsing, in particular for a graph-based parser that generates AMR nodes from words. As we will see, one word can trigger multiple AMR nodes due to some implicit concepts. We will handle them using a rule-based system in Chapter 4 and a learning-based system in Chapter 5.

Named Entities AMR recognizes a named entity with fine-grained types, and if possible, links it to its Wikipedia page. For example, "Boris Johnson" has the following AMR:

```
(p / person
    :wiki ``Boris_Johnson"
    :name (n / name
              :op1 ``Boris"
              :op2 ``Johnson"))
```

As one can see, there is an extra wiki tag, a 'name' node denoting this is a named entity, and a 'person' node denoting this named entity is a person. Other named entity types include 'country', 'organization', 'location', and 'company'. Note that "Boris" and "Johnson" in the AMR graph are not associated with a variable. They can be treated as the attribute of the node (Oepen et al., 2019, 2020). Other attributes include numbers and polarity tag. Still, we can treat attributes as nodes in the graph. Excluding the wiki tag, there are still four nodes that we need to generate from two words. This presents an obvious challenge to AMR node generation.

Persons & Things AMR uses PropBank frames to denote frames for consistency. A frame could be triggered by a noun or a verb. When it is triggered by a noun, it is usually associated with some kind of person or thing. For example, "the proposal of the teacher" has the following AMR:

```
(t / thing
    :ARG1-of (p / propose-01)
          :ARG0 (p2 / person
                :ARG0-of (t2 / teach-01))))
```

'thing' and 'person' nodes are introduced to capture the nature of participants, and the frames are used to represent the meaning. Consequently, when the meaning of the noun significantly differ from the meaning of the verb, AMR use the original noun. For example, a professor is not a person who professes. So, it is represented as a single concept node 'professor'.

Quantities Just as a named entity has a name and possibly a wiki tag, a quantity is naturally associated with a unit, a number denoting the quantity per unit, and a fine-grained type. In particular, the number is normalized explicitly. For example, "one hundred and ten miles" has the following AMR:

```
(q / distance-quantity
```

```
    :unit (m / mile)
    :quant 110)
```

Other quantity types include 'volume-quantity', 'temporal-quantity', and 'monetary-quantity'. Each is associated with its own set of units.

Other Entities There are entities other than named entities. An entity is associated with its own set of attributes. In particular, we have 'date-entity' for "February 29, 2012":

```
(d / date-entity
     :year 2012
     :month 2
     :day 29)
```

This set of attributes is not fixed, as 'date-entity' can also describe a week day "Friday":

```
(d / date-entity
    :weekday (f / friday))
```

Other entities usually have only a value attribute. For example, "the second planet" are annotated with:

```
(p / planet
    :ord (o / ordinal-entity
            :value 2))
```

Special Frames AMR use some special frames that captures personal role relations. For example, "US President Trump" has the AMR:

```
(p / person
    :wiki ``Donald_Trump"
    :name (n / name :op1 ``Trump")
    :ARG0-of (h / have-org-role-91
                :ARG1 (c / country
                         :wiki ``United_States"
                         :name (n2 / name :op1 ``US"))
                :ARG2 (p2 / president)))
```

'have-org-role-91' frame describes a person's role in an organization. The ARG0 is the office older ARG1 is the organization, and ARG2 is the office title. As we can

see, two words can correspond to a quite large AMR graph. Similarly, we have a 'have-rel-role-91' frame that describes inter-personal relation.

Negation AMR captures the negation of meaning through a polarity attribute. The negation can be triggered by either keyword like 'not' or a morphological affix such as 'in'. For example, both "the dress is inappropriate" and "the dress is not appropriate" have the following AMR:

```
(a / appropriate-02
  :polarity -
  :ARG1 (d / dress))
```

Modality AMR captures modality through frames. For example, "The boy may go." has the following AMR:

```
(o / permit-01
    :ARG1 (g / go-02
              :ARG0 (b / boy)))
```

This is problematic for string-similarity based alignment, as the trigger word 'may' has no resemblance with the frame 'permit-01'. This is one of the key motivations for a learning-based approach for the alignment that we pursued in Chapter 4.

Acronyms AMR expands acronyms explicitly. For example, "CEO" is annotated as:

```
(o / officer
     :mod (e3 / executive)
     :mod (c7 / chief))
```

Note that they usually appear in a much longer sentence in the corpus. This is very difficult to handle for the graph-based approach. First, same as modality, string-similarity based alignment will not work; second, it is many-to-one alignment like entities; third, they are un-systematic. We cannot categorize all the acronyms without a dictionary.

We have surveyed several aspects of AMR that are challenging for graph-based parsing. There are still many semantic phenomena that is not captured by AMR, including scope, universal quantification, tense, and aspect (Banarescu et al., 2013).

2.2.2 Recent Developments

Aside from the continuing development of the standard AMR sembanking through the LDC2014T12, LDC2015E86, LDC2016E25, LDC2017T10, and LDC2020T02

datasets, there are some other recent developments on the AMR formalism. AMR can be extended to the multi-sentence setting (O'Gorman et al., 2018) through sharing variable names.

AMR or AMR-like formalism has been adopted to specific domains. Bonial et al. (2019, 2020) adopted AMR to the domain of human-robot interaction. Alexa meaning representation is proposed to model sentence meaning that supports inference in the Amazon Alexa system (Kollar et al., 2018). The major advantage of such a domain-specific setting is that the meaning representation is not only grounded to some frames files that are human-readable but is executable by the machine.

Another frontier is the cross-lingual setting. While machine translation is one of the original motivation for developing AMR (Jones et al., 2012; Banarescu et al., 2013), AMR itself is not an interlingua (Banarescu et al., 2013).[7] Consequently, AMR sembanks for various other languages have been developed, including Korean (Choe et al., 2020), Brazilian Portuguese (Sobrevilla Cabezudo and Pardo, 2019), Spanish (Migueles-Abraira et al., 2018), Chinese (Li et al., 2019a) and Turkish (Azin and Eryiğit, 2019).

Both the developments on some specific domains and new languages require training AMR parsers for the new settings. Ideally, the parsers should be trained end-to-end without pipeline approaches, including alignment and graph pre-processing to reduce error propagation. Our Chapter 4 and Chapter 5 on AMR parsing overcome such pipeline approach for the graph-based parsers.

2.3 Other Formalisms

There are other broad-coverage meaning representations. As pointed out by the recent shared tasks on cross framework semantic parsing (Oepen et al., 2019, 2020), many meaning representations can be regarded as directed acyclic graphs. First, there are DELPH-IN MRS Bi-Lexical Dependencies (DM) (Ivanova et al., 2012) and Prague Semantic Dependencies (PSD) (Hajič et al., 2012). They capture semantic dependencies among surface words. Then, there are Elementary Dependency Structures (EDS) (Oepen and Lønning, 2006) and Universal Conceptual Cognitive Annotation (UCCA) (Abend and Rappoport, 2013) whose nodes might not correspond to sur-

[7] However, Xue et al. (2014) disputed this notion to some extent. Later, Damonte and Cohen (2018) and Blloshmi et al. (2020) built parsers for English AMR on Italian, Spanish, German and Chinese sentence.

face words. In EDS, all nodes are aligned, but the alignments could overlap. In UCCA, there are unaligned implicit nodes. They are more similar to AMR, but most alignments are given. A graph-based parser can be applied to all of them, including AMR (Zhang et al., 2019c). However, manual categorization/pre-processing of nodes might be needed. Actually, Cao et al. (2019) provided graph-based parsers for DM, PSD, UCCA, AMR but left EDS to future work due to the complexity. The need to manually handle different graph annotations provides motivation to automatize this process. This is the focus of our Chapter 5.

There are two other major broad-coverage meaning representations that are not designed to be graph-like or dependency focused. The Discourse Representation Structure (DRS) based on Discourse Representation Theory (Kamp, 1993) has been proposed to capture discourse-level scoped meaning representation (Basile et al., 2012; Abzianidze et al., 2017). Distinct from AMR, DRS captures the scope of variables explicitly,[8] and incorporates universal quantification. Like AMR, DRS also abstracts away from the surface realization. Therefore, it also lacks alignment. Interestingly, Discourse Representation Graph (DRG) is proposed for allowing explicit word alignment (Basile and Bos, 2013). Based on DRG, DRS is incorporated into the more recent cross framework representation parsing shared task (Oepen et al., 2020). In addition, a tree-structured representation of DRS has been proposed by Liu et al. (2018, 2019b) to facilitate parsing.

The other broad-coverage meaning representation is the distributed meaning representation as championed by connectionism (Hinton et al., 1986). In essence, distributed meaning representation embed the words, phrases, and sentences into vector space. The distributed representation is the foundation of deep learning in natural language processing. The distributed meaning representations are realized as word2vec (Mikolov et al., 2013), ELMO (Peters et al., 2018a), BERT (Devlin et al., 2019) and RoBERTA (Liu et al., 2019c). In fact, most symbolic meaning representation parsers rely on distributed meaning representations. In this sense, they are at least as broad-coverage as the symbolic ones. Very recently, Wu et al. (2020) shows semantic dependencies can be used to improve the performance of a RoBERTA baseline model on several natural language understanding tasks, suggesting explicit semantic dependency structure is still useful.

[8] In AMR, the variables have no scope.

2.4 Applications

Both SRL and AMR can be potentially beneficial in many semantic-related NLP tasks. Semantic roles and AMRs have been used in machine translation system (Wu and Fung, 2009; Liu and Gildea, 2010). In fact, as a precursor of the AMR sembank (Banarescu et al., 2013). Jones et al. (2012) proposed graphs as intermediate semantic representations for machine translation. More recently, Marcheggiani et al. (2018) exploit the dependency-based representation of predicate-argument structures by feeding them to a graph convolution network (GCN) (Kipf and Welling, 2017) to improve machine translation. Similarly, Song et al. (2019) encodes AMR graph with GCN to improve machine translation.

Another major application is question answering. This traditionally involves symbolic manipulation of the meaning representation. Mitra and Baral (2016) extracts domain-specific meaning representation from AMR to perform inference. In a quite exciting recent development, Kapanipathi et al. (2020) convert AMR to knowledge graph triples and applies neural-symbolic inference for knowledge base question answering, and achieves state-of-the-art performance. Regarding SRL, the QA-SRL (He et al., 2015; Klein et al., 2020) formalism can be used for question answering and has been further applied to open domain information extraction (Stanovsky et al., 2018). This line of applications is very important for symbolic broad-coverage meaning representation because logic inference can be applied either directly on the broad-coverage meaning representation or a converted domain-specific meaning representation with the help of conversion rules. This removes the need to train a parser for a specific domain, demonstrating a qualitative edge over the distributed meaning representation.

Other recent applications of AMR[9] include text summarization (Liu et al., 2015; Takase et al., 2016; Hardy and Vlachos, 2018; Liao et al., 2018), paraphrase detection (Issa et al., 2018) and entity-linking (Pan et al., 2015).

2.5 Broad-coverage Semantic Parsing

We explain the main approaches for semantic parsing and focus on graph-based parsing. In particular, we focus on AMR parsing, but the ideas could be applied to other meaning representations. For a more comprehensive discussion, one can refer to Oepen

[9] As AMR contains the predicate-argument structure, recent applications have been focused on using AMR, instead of SRL.

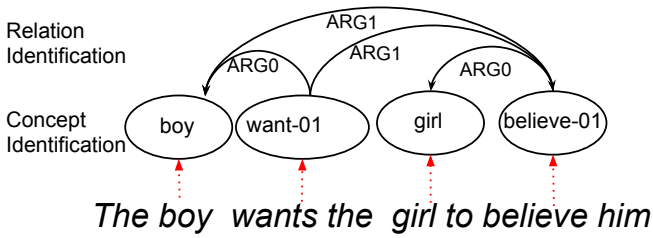


Figure 2.3: Graph-based Parsing for Abstract Meaning Representation.

et al. (2019, 2020). First, we explain graph-based parsing, and discuss other parsing approaches. Then, we compare the graph-based parsing to other approaches, and highlight the problems we face in graph-based parsing.

2.5.1 Graph-based Parsing

The central idea of graph-based parsing is the factorization of the prediction of a graph into its individual nodes and edges, conditioning on the anchoring of nodes to words. Formally, graph-based parsing consists of two stages: the concept identification stage and the relation identification stage. In the concept identification stage, graph-based parsers work as a tagger that identify concept nodes from anchored tokens. In the relation identification stage, edges between nodes are identified and labeled. Concretely, as we can see from Figure 2.3, the nodes 'boy', 'want-01' , 'girl' and 'believe-01' are identified from the word 'boy', 'wants', 'girl' and 'believe' respectively. Then, the relation identification is conditioned on nodes and the original tokens that generate them. Of course, those predictions are conditioned on neural contextualized encoding. We leave those details to the main chapters.[10] Now, let us have a look at alternative parsing approaches.

2.5.2 Alternative Approaches

There are three main alternative approaches for semantic parsing: neural seq2seq, transition system, and grammar-based. The neural seq2seq approach to semantic parsing is first proposed for domain-specific meaning representation (Dong and Lapata, 2016). It applies an auto-regressive model to predict the meaning representation with the help of an attention mechanism (Sutskever et al., 2014; Bahdanau et al., 2015). In AMR parsing, there are two variants of realizations: linearize the AMR graph, and predict nodes

[10] Also, the root identification is not shown, but that can be done as a separate classification on choosing a node.

and edges alternately (Zhang et al., 2019b; Cai and Lam, 2020); predict the AMR as in the (simplified) Penman notation (Konstas et al., 2017; van Noord and Bos, 2017; Xu et al., 2020). The difference between the two is that the former treats nodes and edges differently.

A transition system converts the prediction of the meaning representation into a sequence of actions that produce the meaning representation incrementally. Actions can be performed when traversing through the original sentences (Damonte et al., 2017; Ballesteros and Al-Onaizan, 2017; Guo and Lu, 2018; Naseem et al., 2019; Fernandez Astudillo et al., 2020; Lee et al., 2020) or the dependency tree (Wang et al., 2015b,a; Goodman et al., 2016).

The grammar-based approach produces parts of the meaning representation from individual spans, then combine them to form the graph. The learning, of course, requires a decomposition of the graph into fragments (see Section 5.8 for more). The grammar constraints the combination rules of parts, and hence provides guidance for the decomposition of the graph. In AMR, various grammars have been used, including Hyperedge Replacement Grammar (HRG) (Braune et al., 2014; Peng et al., 2015; Gildea et al., 2019), Combinatory Categorial Grammar (CCG) (Artzi et al., 2015; Beschke and Menzel, 2018) and Apply-Modify (AM) algebra (Groschwitz et al., 2018; Lindemann et al., 2020).

As the grammar-based approach is similar to the graph-based approach, we clarify their differences. They are similar in that sequence tagging is performed to identify semantic elements. In graph-based parsing, the semantic elements are simply the semantic concepts, and all the decisions about semantic relations are postponed to the relation identification stage. However, in grammar-based parsing, those semantic elements are graph fragments that already specify semantic relations. The composition stage then decides the application of semantic relations (i.e., find their argument).[11] Compared to grammar-based parsing that has restrictions on composition rules, graph-based parsing is more flexible regarding possible semantic relations. Meanwhile, grammar-based parsing builds in a stronger inductive bias through such restrictions.

2.5.3 Comparing Graph-based Parsing to Alternatives

Compared to other approaches, the graph-based parsing is appealing because of its simplicity and interpretability. Unlike the grammar-based or transition-based approach,

[11] Of course, CCG works on logic forms, not on graphs directly, but there is a correspondence between CCG lexicons and graph fragments (Artzi et al., 2015).

there is no auxiliary scaffolding (i.e., manually defined actions or grammar rules) that is required. This avoids the errors due to the lack of coverage. The interpretability is in contrast with the neural seq2seq approach that uses soft attention. In graph-based parsing, the nodes are identified from hard aligned words. In spite of the advantages that graph-based parsing offers, it can be hard to apply and has limited expressiveness.

2.5.3.1 Pipeline Training

The graph-based parsing is not always straightforward to apply for two reasons: lack of annotated anchoring of nodes and one-to-many mapping from words to nodes. AMR nodes are not explicitly anchored/aligned to the words in the AMR sembank. Also, as we have shown in Section 2.2.1, there are many semantic phenomena that have several nodes being triggered by one word. Importantly, while this thesis focuses on AMR, those two problems are not limited to AMR. In particular, UCCA and DRS can have nodes without alignment; EDS and DRS can have one-to-many mapping. In AMR parsing, those two problems have been addressed prior to our work through a pipeline approach.

A pipeline involves the pre-alignment of nodes and the segmentation of subgraphs. The alignment is usually provided by a separate statistical aligner (Pourdamghani et al., 2014a). Some have built aligners specifically for training their parsers (Werling et al., 2015; Wang and Xue, 2017). Meanwhile, the one-to-many problem is solved through explicit rule-based segmentation of subgraphs so that one token or one span can generate at most one subgraph (Flanigan et al., 2014; Werling et al., 2015; Foland and Martin, 2017).

In Chapter 4 and Chapter 5, we remove the pipeline in training by modeling the anchoring as a latent structured discrete variable. Crucially, we make the training end-to-end differentiable. To this end, we introduce recent developments on gradient estimation over the latent structured discrete variable in the next chapter.

2.5.3.2 Limited Expressiveness

Before the rise of deep learning models, structured prediction in natural language processing usually involves scoring the high-order features in the output space (Collins, 2002; Roark et al., 2004; Liang et al., 2006; Huang, 2008). However, current state-of-the-art systems (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017, 2018) use powerful sentence encoders (e.g., layers of LSTMs (Li et al., 2019b; He et al.,

2017) or multi-head self-attention (Strubell et al., 2018)) and factorize the prediction into its parts. This factorization strategy is the essence of our graph-based parsing, and our parsers also rely on the expressiveness of the sentence encoder.

Powerful encoders, in principle, can capture long-distance dependencies and hence alleviate the need for modeling high-order interactions in the output. However, capturing these interactions in the encoder would require substantial amounts of data. Even if we have domain knowledge about likely interactions between components of the predicted graphs, it is hard to inject this knowledge in an encoder. Empirically, the trend towards more factorizable models is common for most structured prediction tasks in NLP (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017, 2018).[12]

In broad-coverage semantic parsing, the lack of a large amount of annotated data is likely to be the case in the near future, due to the high cost of annotation. Therefore, capturing high-order output interdependency could still be important for semantic parsing. This thesis alleviates the limited expressiveness of factorized prediction. As AMR has too many varying parts, we choose SRL as our testing bed. We present our method in Chapter 6.

[12]The only major exceptions are language generation tasks, especially machine translation and language modeling, where larger amounts of text are typically available. For those tasks, recent models are usually autoregressive.

Chapter 3

Gradient Estimation over Latent Structured Discrete Variable

This chapter introduces techniques that are needed to make the training of our AMR parsers end-to-end differentiable. Our AMR parsers (Chapter 4 and Chapter 5) have the anchoring of concept nodes to sentence tokens as a discrete latent variable. We induce the anchoring in the deep variational inference framework. The anchoring of different nodes could interact (e.g., injectivity), so the anchoring is structured. Inferring such a latent anchoring requires Monte-Carlo gradient estimation involving sampling structured discrete variables.

Let us define our problem formally. Denote $\mathbf{x} \in \mathcal{X}$ as an observable data point (includes both input and output), $\mathbf{z} \in \mathcal{Z}$ as the latent structure associated with the data point. We focus on modeling a discrete latent variable, therefore we represent it as a binary vector $\mathcal{Z} \subset \{0,1\}^n$. Furthermore, we have a prior probability distribution $P_\theta(\mathbf{z})$[1] over the latent variable and a likelihood function $P_\theta(\mathbf{x}|\mathbf{z})$ of the data conditioning on the latent variable. Now, we are trying to minimize the marginal likelihood $\log P_\theta(\mathbf{x}) = \log \sum_{\mathbf{z}} P_\theta(\mathbf{z}) P_\theta(\mathbf{x}|\mathbf{z})$ w.r.t. θ. The objective is intractable[2], and even harder to optimize with gradient based optimization methods.

In the remainder of this chapter, we review deep variational inference, which enables efficient inference when the marginalization is intractable. Still, deep variational inference involves Monte-Carlo gradient estimation over the sampling of $\mathbf{z}$. We review the differentiable sampling of structured discrete variables to show how to do the Monte-Carlo gradient estimation. In particular, Paulus et al. (2020) proposed *Stochas-*

[1] $P_\theta(\mathbf{z})$ does not have to be parameterized. Also, as it is not conditioned on the observable, we refer to it as prior.

[2] Unless the marginal likelihood can be factorized or $|\mathcal{Z}|$ is small.

tic Softmax as a general method for constructing a differentiable sampling process for structured discrete variables. Furthermore, we introduce *Gumbel-Sinkhorn* (Mena et al., 2018), an instance of stochastic softmax for sampling permutations. We use Gumbel-Sinkhorn in Chapter 4 to model bijective alignment and extend it in Chapter 5 to model segmentation and alignment jointly.

3.1 Deep Variational Inference

On a conceptual level, when the marginal likelihood can no longer be computed, Monte-Carlo methods can estimate the marginal likelihood. However, when performing Monte-Carlo estimation, sampling from the prior could result in a large variance. Instead, we can sample from an approximated-posterior distribution (aka, the variational distribution) to reduce the variance. This process is referred to as variational inference. Deep variational inference takes a further step to parameterize our variational distributions with neural networks. This neural-parameterization estimates variational distributions across many data points effectively.

The variational inference approach optimizes a lower bound on the original marginal likelihood by marginalizing over a simple variational distribution $Q(\mathbf{z}|\lambda)$, which is parameterized by λ. With deep learning, a simple distribution means a distribution that we can efficiently sample from. However, in contrast to using the prior distribution, we expect the variational distribution to result in sampling latent structures such that joint probability of the data point tends to be high. The replacement of the prior distribution by a variational distribution is at the cost of having a biased objective, which is called the evidence lower bound (ELBO). The ELBO can be derived as follows:

$$\log \mathop{\mathrm{E}}_{P_\theta(\mathbf{z})} P_\theta(\mathbf{x}|\mathbf{z}) \tag{3.1}$$

(introduce variational distribution $Q(\mathbf{z}|\lambda)$)

$$= \log \mathop{\mathrm{E}}_{Q(\mathbf{z}|\lambda)} \frac{P_\theta(\mathbf{x},\mathbf{z})}{Q(\mathbf{z}|\lambda)} \tag{3.2}$$

(Jensen's inequality)

$$\geq \mathop{\mathrm{E}}_{Q(\mathbf{z}|\lambda)} \log \frac{P_\theta(\mathbf{x},\mathbf{z})}{Q(\mathbf{z}|\lambda)} \tag{3.3}$$

(re-arrange terms)

$$= \mathop{\mathrm{E}}_{Q(\mathbf{z}|\lambda)} \log P_\theta(\mathbf{x}|\mathbf{z}) - \mathrm{KL}(Q(z|\lambda)||P_\theta(\mathbf{z})) \tag{3.4}$$

where λ parameterizes the variational distribution of the individual data point. Importantly, the ELBO can be a tight lower bound, when the variational distribution equals the true posterior (formally, when $Q(z|\lambda) = P_\theta(\mathbf{z}|\mathbf{x})$). To make the notation compact, we have $\text{ELBO}(\mathbf{x}, \theta, \lambda)$ as:

$$\text{ELBO}(\mathbf{x}, \theta, \lambda) := \mathop{\text{E}}_{Q(z|\lambda)} \log P_\theta(\mathbf{x}|z) - \text{KL}(Q(z|\lambda)||P_\theta(z)) \tag{3.5}$$

Maximizing ELBO is easy when the coordinate ascent[3] w.r.t. λ (E-step) and θ (M-step) can be computed with closed form, as in *Variational Expectation Maximization* (Neal and Hinton, 1999). However, this is not the case when deep models are involved.[4] The optimization in deep learning is based on gradient estimation, however, even with a simple variational distribution, mariginalization is not tractable, sampling is still required. Furthermore, we actually need the gradient of the objective (i.e., the log probability) with respect to the parameters not the objective itself. We explain the Monte-Carlo gradient estimation in the Section 3.1.2.

In the remainder of this section, we optimize the objective for the entire dataset, instead of a single data point.[5] Let us turn to *Amortized Variational Inference* that enables variational inference over a data set.

3.1.1 Amortized Variational Inference

Amortized inference is introduced in Gershman and Goodman (2014). Their key intuition is that the inference procedure in a new situation can exploit past inference experiences. Ritchie et al. (2016) further proposed to use neural networks to parameterize conditional distribution as a way to realize this idea in the deep learning context. This leads to *Amortized Variational Inference*. This term is hence adopted as the standard name, but the technique existed earlier (Dayan et al., 1995).

From a technical perspective, amortized variational inference (aka, deep variational inference) is introduced to approximately perform the E-step (i.e., optimize ELBO w.r.t. λ) in the classical variational expectation maximization. The key idea is to replace $\lambda^* = \arg\max_\lambda \text{ELBO}(\mathbf{x}, \theta, \lambda)$ with $\lambda := \text{ENC}_\phi(\mathbf{x})$, an encoder neural network

[3] Coordinate ascent alternatively maximizes the ELBO with respect to groups of parameters while keeping the other fixed.

[4] When deep models are involved, computing the gradient (but not maximizing) of the ELBO w.r.t. θ might still be possible. This happens when the original marginal likelihood is tractable, and the E-step computes the true posterior (Kim et al., 2018).

[5] When the data set is huge, optimization also becomes an issue for classical models, which leads to the development of *Stochastic Variational Inference* (Hoffman et al., 2013).

with parameters ϕ that estimates the optimum parameters. The parameters ϕ can be used across all data points. Therefore, denoting $\mathbf{x}^{(i)}$ as the ith data point, we are optimizing $\sum_i \text{ELBO}(\mathbf{x}^{(i)}, \theta, \lambda = \text{ENC}_\phi(\mathbf{x}^{(i)}))$ w.r.t. both θ and ϕ.

As variational parameters are estimated by a neural encoder, the variational distribution is not directly chosen to optimize the ELBO. On the one hand, such a direct estimation strategy comes with the cost of a looser bound. In particular, Cremer et al. (2018) shows the approximation gap (aka, amortization gap) caused by amortization could be significant comparing to the gap between the ELBO of the optimum posterior parameters and likelihood. On the other hand, without amortization, it is also expensive to run the E-step until convergence for each data point. In addition, Shu et al. (2018) argues that amortization could be leveraged to introduce regularization on the posterior estimation.

Conceptually, amortized variational inference works well with the modern deep learning philosophy of optimizing a common objective with gradient-based methods. Yet, computing $\nabla_\lambda \text{ELBO}(\mathbf{x}^{(i)}, \theta, \lambda)$ is not easy for discrete $\mathbf{z}$.

3.1.2 Monte-Carlo Gradient Estimation

Even with the amortized variational inference that provides an efficient estimation of the posterior distribution, the actual computation of ELBO is still intractable when neural networks are involved. As we can see in equation 3.4, weighted summation of $\log P_\theta(\mathbf{x}|\mathbf{z})$ over $\mathbf{z}$ is required. However, with deep neural networks being used, this is infeasible unless the marginal likelihood can be factorized or $\mathcal{Z}$ is small.

While Monte Carlo estimation can be applied to estimate an integral, using it to estimate gradients is not trivial. We focus on one data point, and optimize w.r.t. λ and θ. We want to get estimators $\widehat{\eta}_\theta(\mathbf{x})$ and $\widehat{\eta}_\lambda(\mathbf{x})$ of their corresponding expected gradient over sampled $\mathbf{z}$. The Monte Carlo gradient estimator $\widehat{\eta}_\theta(\mathbf{x})$ can be simply derived by exchanging gradient and expectation:[6]

$$\begin{aligned} \eta_\theta(\mathbf{x}) &:= \nabla_\theta \mathop{\mathrm{E}}_{Q(\mathbf{z}|\mathbf{x};\lambda)} \log \frac{P_\theta(\mathbf{x},\mathbf{z})}{Q(\mathbf{z}|\mathbf{x};\lambda)} \\ &= \mathop{\mathrm{E}}_{Q(z|\mathbf{x};\lambda)} \nabla_\theta \log P_\theta(\mathbf{x},\mathbf{z}) \end{aligned} \tag{3.6}$$

This means, we get K samples of $\mathbf{z}$ from $Q(\mathbf{z}|\mathbf{x};\lambda)$, take gradients w.r.t. $\log P_\theta(\mathbf{z},\mathbf{x})$,

[6] There are some technical conditions regarding exchangbility of gradient and expectation, but they are usually satisfied in machine learning.

and average gradients over samples.

$$\widehat{\eta}_\theta(x) = \frac{1}{K}\sum_{i=1}^{K} \nabla_\theta \log P_\theta(\mathbf{x}, \mathbf{z}_i) \tag{3.7}$$

$$\mathbf{z}_i \sim Q(\mathbf{z}|\mathbf{x};\lambda) \tag{3.8}$$

However, the expectation of gradient is not the gradient of expectation when the distribution is parameterized. For Monte Carlo gradient estimation with respect to parameters of the sampling distribution, there are primarily two approaches: score function gradient estimator (Glynn, 1990; Kleijnen and Rubinstein, 1996) (aka, REINFORCE (Williams, 1992)) and the reparameterization of the variational distribution. Readers could refer to Mohamed et al. (2020) for more discussions. Importantly, the score function gradient estimator requires evaluation of $Q(\mathbf{z}|\lambda)$, which is not feasible for our purpose (see Section 3.2.3). In fact, sometimes even the sampling can be hard to define when the variable is structured. In the next section, we explain how differentiable sampling works for structured discrete random variable, which is a generalization of reparmeterization that works for continuous random variables (Williams, 1992; Titsias and Lázaro-Gredilla, 2014; Rezende et al., 2014; Kingma and Welling, 2014) and categorical variables (*Gumbel-Softmax* (Jang et al., 2017; Maddison et al., 2017)).

3.2 Differentiable Sampling

When discrete variables have non-trivial constraints, both sampling and differentiability become an issue. Intuitively, differentiable sampling defines a sampling process that converts a fixed distribution into a distribution over structured discrete variables. Crucially, such conversion is differentiable and parameterized. Therefore, a sample from this process can be differentiable w.r.t. the parameters.

Formally, we are interested in estimating $\nabla_\lambda \mathop{\mathrm{E}}_{P(\mathbf{z}|\lambda)} \mathcal{L}(\mathbf{z})$ with $\widehat{\eta}_\lambda := \nabla_\lambda \mathcal{L}(\mathbf{z}(\lambda, \varepsilon))$ where $\varepsilon \sim P(\varepsilon)$, λ is our parameter, $\mathcal{L}$ defines the loss given a latent structure[7] and a function $\mathbf{z}$ produces desired samples from parameter λ and random perturbation ε. To make the notation compact, we overload $\mathbf{z}$ to represent both the variable or the function that produces the variable. For $\nabla_\lambda \mathcal{L}(\mathbf{z}(\lambda, \varepsilon))$ to be well defined, both $\nabla_\mathbf{z} \mathcal{L}$ and $\nabla_\lambda \mathbf{z}$ should exist. Consequently, $\mathcal{L}$ need to be defined for the continuous-valued $\mathbf{z}$. If $\mathbf{z}$ is used by a model as a latent variable, the relaxation of $\mathcal{L}$ involves relaxation

[7] $\mathcal{L}(\mathbf{z})$ could be $\log P_\theta(\mathbf{x}|\mathbf{z})$ as in the earlier discussion, but we use $\mathcal{L}(\mathbf{z})$ to be more compact.

of the model. In Chapter 5, such model relaxation becomes non-trivial. In this section, we focus on the differentiable sampling process that generates continuous $\mathbf{z}$.

In the remaining of this section, first, we review the stochastic softmax (Paulus et al., 2020) that formulates differentiable sampling of many structured variables through optimization with a smoothed convex objective. Then, while stochastic softmax defines a differentiable sample w.r.t. λ, we still need a way to compute the gradient. We discuss the gradient computation when the stochastic softmax is used for sampling. Last, we introduce Gumbel-Sinkhorn (Mena et al., 2018)) that provides a differentiable sampling of permutations.

3.2.1 Stochastic-Softmax

The stochastic softmax (Paulus et al., 2020) is a generalization the Gumbel-Softmax trick (Jang et al., 2017; Maddison et al., 2017) to the structured case, and builds on top of the perturb-and-MAP methods (Papandreou and Yuille, 2011). Stochastic softmax samples a differentiable discrete structure in two steps: perturbing the logits and maximizing a regularized linear objective over the continuous space:

Definition 1 (Stochastic-Softmax)**.** *For a fixed distribution $P(\varepsilon)$, stochastic-softmax implicitly defines a distribution $P(\mathbf{z}|\lambda)$ over* $\text{conv}(\mathcal{Z}) \subset \mathbb{R}^n$ *through the following sampling process:*

$$\forall 0 \leq i < n, \varepsilon_i \sim \mathcal{G}(\varepsilon; 0, 1) \tag{3.9}$$

$$\mathbf{z}(\lambda, \varepsilon, \tau) = \underset{\mathbf{z} \in \text{conv}(\mathcal{Z})}{\arg\max} \langle \lambda + \varepsilon, \mathbf{z} \rangle - \tau \langle \mathbf{z}, \log \mathbf{z} \rangle \tag{3.10}$$

where $\text{conv}(\mathcal{Z}) = \{ \sum_{0 \leq m < |\mathcal{Z}|} a_m \mathbf{z}^m | a_m \geq 0, \sum_m a_m = 1 \}$ is the convex hull of set $\mathcal{Z}$ ($\mathbf{z}^m$ represents an element in $\mathcal{Z}$). Importantly, we have introduced an entropic regularizor, weighted by $\tau > 0$ ('the temperature').[8] This entropic regularizor makes the objective strongly concave, and $\mathbf{z}(\lambda, \varepsilon, \tau)$ differentiable with respect to λ (Barratt, 2018; Agrawal et al., 2019; Paulus et al., 2020). Typically we have $P(\varepsilon) = \mathcal{G}(\varepsilon; 0, 1)$, the standard Gumbel distribution (Gumbel, 1954), which has a probability density function $\exp(-\varepsilon - \exp(-\varepsilon))$, and a cumulative distribution function $\exp(-\exp(-\varepsilon))$.[9]

[8] In general, we can have a strongly differentiable concave regularizor other than entropic regularizor.

[9] In general, we have $\varepsilon \sim \mathcal{G}(\varepsilon; \mu, \beta) \overset{d}{=} \mu + \beta\varepsilon$ where $\varepsilon \sim \mathcal{G}(\varepsilon; 0, 1)$. Furthermore, a Gumbel variable from the standard Gumbel distribution can be sampled with the inverse transform sampling: $\varepsilon \sim \mathcal{G}(\varepsilon; 0, 1) \overset{d}{=} -\log(-\log(u))$ where $u \sim \text{Uni}(0, 1)$.

On the theoretical side, the relaxation of $\mathcal{Z}$ to its convex hull is tight. Consequently, under mild conditions, the convex hull relaxation ensures the stochastic softmax at 0 temperature can sample an element from $\mathcal{Z}$ with probability 1 (Paulus et al., 2020). Furthermore, setting the temperature at 0 reduces our stochastic softmax approach to perturb-and-map models (Papandreou and Yuille, 2011) that approximate the Gibbs distribution (Hazan et al., 2013; Tomczak, 2016). Three main practical problems remain: find the convex hull, solve the optimization problem, and compute the gradient.

In general, it is hard to find those convex hulls (i.e., represent them by a set of linear inequalities). Fortunately, for a lot of applications (e.g., permutations, context-free grammars, spanning trees), their convex hulls are known (Mena et al., 2018; Rush et al., 2010; Paulus et al., 2020). We discuss permutations in Section 3.2.3. As for optimization, since every convex hull can be represented by a set of linear inequalities (Korte and Vygen, 2018, page. 69), such convex optimization problem can be solved by standard package like cvxpy (Diamond and Boyd, 2016). However, the efficiency of solving them could be a problem for NLP applications. We discuss the Sinkhorn algorithm that efficiently finds the optimum relaxed permutation in Section 3.2.3. Now, we address the gradient computation issue.

3.2.2 Gradient Computation

To compute the gradient $\nabla_\lambda \mathbf{z}(\lambda, \varepsilon, \tau)$, there are three main approaches: analytic gradient, numerical approximation and unrolled optimization.

In analytic gradient computation, implicit function theorem gives the analytic gradient at the maximum of a strongly concave objective (Barratt, 2018). In particular, CvxpyLayer (Agrawal et al., 2019) provides a PyTorch (Paszke et al., 2019) extension to CVX that automatically solves the optimization problems and provides a backpropagation functionality. The main drawback is that the standardized procedure is too slow for most NLP applications.[10]

Numerical approximation (Paulus et al., 2020) can be performed directly on the final loss $\mathcal{L}$ w.r.t. λ, and is given by:

$$\mathbf{z} := \mathbf{z}(\lambda, \varepsilon, \tau) \tag{3.11}$$

$$\nabla_\lambda \mathcal{L}(\mathbf{z}) \approx \frac{\mathbf{z}(\lambda, \varepsilon + \sigma \nabla_{\mathbf{z}} \mathcal{L}(\mathbf{z}), \tau) - \mathbf{z}(\lambda, \varepsilon - \sigma \nabla_{\mathbf{z}} \mathcal{L}(\mathbf{z}), \tau)}{2\sigma} \tag{3.12}$$

[10] The current standardized solver involves sparse matrix operation, which is unnecessary for many special problems. Also, in NLP, problem size always varies. This leads to repetitive constructions of the computation graph.

where σ is a small number. This numerical approximation requires two additional calls of the stochastic softmax. As numerical approximation can be vulnerable to numerical issues, this is perhaps best saved as a last resort.

In unrolled optimization, a differentiable optimization process[11] that solves the problem is treated as a forward function, whose gradient can be computed by automatic differentiation packages (e.g., PyTorch). In practice, optimization algorithms need to stop at finite steps. This might lead to a GPU memory issue if the number of steps is large. Fortunately, with the recent gradient checkpoint utility in PyTorch, intermediate steps in an iterative optimization algorithm can be redone during the backpropagation so that the GPU memory will not be a bottleneck for the unrolled optimization. The unrolled optimization will be the gradient computation method used in the later chapters.

3.2.2.1 Straight-Through Estimator

One limitation of the differentiable sampling is that it creates a gap between training (where a continuous relaxation is used) and testing (where everything is discrete). A naive solution is to set the τ to be a small value. However, this will increase the variance of the Monte-Carlo gradient estimation.[12] One popular way to bridge this gap is to use the Straight-Through (ST) gradient estimator (Bengio et al., 2013a; Jang et al., 2017), i.e., use a discrete solution in the forward computation pass, but backpropagate with the relaxed computation. In our structured ST gradient estimator, we use $\mathbf{z}(\lambda, \varepsilon)$ in the forward pass, and set $\nabla_\lambda \mathcal{L}(\mathbf{z}(\lambda, \varepsilon)) := \nabla_\lambda \mathcal{L}(\mathbf{z}(\lambda, \varepsilon, \tau))$. Conceptually, the ST gradient estimator samples a discrete structure in the forward pass, and keeps the gradient well behaved.

3.2.3 Gumbel-Sinkhorn

We have a set of permutations $\mathcal{Z} = \{\mathbf{z} \in \{0,1\}^{n \times n} | \forall j, \sum_i \mathbf{z}_{ij} = 1; \forall i, \sum_j \mathbf{z}_{ij} = 1\}$ that are represented by a binary-valued matrix with two sets of normalization constraints. Crucially, if we were to explicitly define $P(\mathbf{z}) \propto \exp\langle \lambda, \mathbf{z} \rangle$, the exact sampling procedure is still unclear. Moreover, the computation of the partition function is intractable (Valiant, 1979).Therefore, the score function gradient estimator cannot be applied for gradient

[11] Each operation in the optimization steps need to be differentiable, otherwise unrolled optimization cannot be applied.

[12] Intuitively speaking, at low temperature, the gradient will vanish most of the time, but very large when λ is near the decision boundary of two discrete $\mathbf{z}$.

estimation due to the inability to compute the probability mass function. Now, let us sample a differentiable permutation.

First, the convex hull of valid permutations is simply its LP relaxation (Birkhoff, 1946). In otherwords, we have $\text{conv}(\mathcal{Z}) = \{\mathbf{z} \in \{0,1\}^{n \times n} | \forall j, \sum_i \mathbf{z}_{ij} = 1; \forall i, \sum_j \mathbf{z}_{ij} = 1\}$. The Gumbel-Sinkhorn distribution can be characterized as:

$$\forall 0 \leq i, j < n, \varepsilon_{ij} \sim \mathcal{G}(\varepsilon; 0, 1) \tag{3.13}$$

$$\begin{aligned} \mathbf{z}(\lambda, \varepsilon, \tau) = & \arg\max_{\mathbf{z} \geq 0} \langle \lambda + \varepsilon, \mathbf{z} \rangle - \tau \langle \mathbf{z}, \log \mathbf{z} \rangle \\ & \text{s.t. } \forall j \sum_i \mathbf{z}_{ij} = 1, \\ & \forall i \sum_j \mathbf{z}_{ij} = 1 \end{aligned} \tag{3.14}$$

While this problem is a standard convex optimization problem, an efficient solver is needed. Sinkhorn algorithm (Sinkhorn, 1964) solves this problem efficiently, and its steps are differentiable. It works as following (Mena et al., 2018):

$$\mathbf{z}^{(0)}(\lambda, \varepsilon, \tau) := \exp\left(\frac{\lambda + \varepsilon}{\tau}\right) \tag{3.15}$$

$$\mathbf{z}^{(t)}(\lambda, \varepsilon, \tau) := \mathcal{T}_c(\mathcal{T}_r(\mathbf{z}^{(t-1)}(\lambda, \varepsilon, \tau))) \tag{3.16}$$

where exp acts component-wise. Dropping the time step, $\mathcal{T}_c(\mathbf{z})_{ij} = \frac{\mathbf{z}_{ij}}{\sum_i \mathbf{z}_{ij}}$ and $\mathcal{T}_r(\mathbf{z})_{ij} = \frac{\mathbf{z}_{ij}}{\sum_j \mathbf{z}_{ij}}$ are the row-wise and column-wise normalization respectively. This alternating normalization algorithm converges to the optimum:

Theorem 1 ((Mena et al., 2018)). *$\mathbf{z}^{(t)}$ in Equation 3.16 has a limit, and $\lim_{t \to \infty} \mathbf{z}^{(t)} = \mathbf{z}(\lambda, \varepsilon, \tau)$ in Equation 3.14*

While the Sinkhorn's method appears to be simple and easy to implement, its proof from Mena et al. (2018) is not very intuitive. We provide an alternative proof on Appendix A by deriving it as a special case of Bregman's method (Bregman, 1967). Importantly, Bregman's method allows for a generalization that can handle other linear equality constraints, which we rely on in Chapter 5.

Now, we use unrolled optimization for gradient computation, and the gradient estimator for $\nabla_\lambda \mathop{\mathrm{E}}_{P(\mathbf{z}|\lambda)} \mathcal{L}(\mathbf{z})$ is: $\widehat{\eta}_\lambda := \nabla_\lambda \mathcal{L}(\mathbf{z}^{(t)}(\lambda, \varepsilon))$ where $\varepsilon_{ij} \sim \mathcal{G}(\varepsilon; 0, 1)$ and $\mathbf{z}^{(t)}(\lambda, \varepsilon)$ as in Equation 3.16.

After the hard work in this chapter, we will use the deep variational inference framework and differentiable sampling for training AMR parsers in the next two chapters.

Chapter 4

AMR Parsing with Latent Alignment

In this chapter, we train a graph-based parser without relying on pre-fixed alignments between nodes in the graph and words in the sentence. We demonstrate that such alignments can be treated as latent variables in a joint probabilistic model and induced in such a way as to be beneficial for AMR parsing. Intuitively, in our probabilistic model, every node in a graph is assumed to be aligned to a word in a sentence: the sentence is encoded with an neural encoder, and each concept is predicted based on the corresponding encoder state. Similarly, graph edges (i.e., relations) are predicted based on representations of concepts and aligned words (see Figure 4.2). As alignments are latent, exact inference requires marginalizing over latent alignments, which is infeasible. Instead, we use deep variational inference as introduced in Chapter 3. Using discrete latent variables in deep learning has proven to be challenging (Mnih and Gregor, 2014; Bornschein and Bengio, 2015). We use a differentiable sampling of the alignment, relying on Gumbel-Sinkhorn distribution (Mena et al., 2018) as introduced in Chapter 3. This yields a computationally-efficient approximate method for estimating our joint probabilistic model of concepts, relations and alignments.

We assume injective alignments from concepts to words: every node in the graph is aligned to a single word in the sentence, and every word is aligned to at most one node in the graph. This is necessary for two reasons. First, it lets us treat concept identification as sequence tagging at test time. For every word, we would predict the corresponding concept or predict $\emptyset$ to signify that no concept should be generated at this position. Secondly, Gumbel-Sinkhorn distribution can only work under this assumption.[1] This constraint, though often appropriate, is problematic for certain AMR constructions (e.g., named entities and many others as we introduced in Chapter 2).

[1] The alignment between all words and nodes (including $\emptyset$ that were to predict) is bijective.

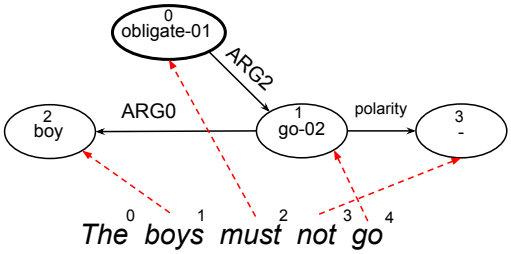


Figure 4.1: An example of AMR, the dashed lines denote latent alignments, *obligate-01* is the root. Numbers indicate depth-first traversal order.

In order to deal with these cases, we introduce AMR concepts re-categorization (see Section 4.4) that combines several concept nodes into one.[2] Similar re-categorization strategies have been used in previous work (Flanigan et al., 2014; Werling et al., 2015; Foland and Martin, 2017; Peng et al., 2017).

Intuitively, as shown in Figure 4.1, while most alignments can be extracted through string matching, the alignment is not always trivial. In this example, *obligate-01* should be aligned to *must*, but we can not know this through string matching. Therefore, learning/inducing alignment is necessary. Furthermore, we believe that using discrete alignments, rather than attention-based models (Bahdanau et al., 2015) is crucial for AMR parsing. AMR banks are significantly smaller than parallel corpora used in machine translation (MT), and hence it is important to inject a useful inductive bias. The injectivity of alignment provides such inductive bias. First, it encodes the observation that concepts are mostly triggered by single words (especially, after re-categorization). Second, it implies that each word corresponds to at most one concept (if any). This encourages competition: alignments are mutually-repulsive. In our example, *obligate* is not lexically similar to the word *must* and may be hard to align. However, given that other concepts are easy to predict, alignment candidates other than *must* and *the* will be immediately ruled out. While the discrete and injective alignment is more interpretable and provides stronger inductive bias, an external aligner is required to obtain it. Meanwhile, neural auto-regressive model (Konstas et al., 2017) that relies on attention (Bahdanau et al., 2015) can learn the correspondence between nodes and words automatically. Our goal is to combine the best of two worlds: to use discrete alignments and induce them while optimizing for the end goal (similarly to the attention component of neural auto-regressive models).

Experimentally, we consider LDC2016E25 (R2) dataset and a smaller LDC2015E86

[2] After re-categorization, we have more words than nodes for most cases. For exceptions, we append ∅ words to the sentence.

(R1) dataset. The resulting parser achieves 74.4% Smatch score on the standard test set when using LDC2016E25 training set,[3] an improvement of 3.4% over the previous best result (van Noord and Bos, 2017). We also demonstrate that inducing alignments within the joint model is indeed beneficial. When, instead of inducing alignments, we follow the standard approach and produce them on pre-processing, the performance drops by 0.9% Smatch.

Our main contributions in this chapter are:

- We introduce a joint probabilistic model for alignment, concept and relation identification;
- We demonstrate that a continuous relaxation can be used to estimate the model effectively;
- We introduce a rule-based re-categorization system that converts AMR subgraph into a node.

The work in this chapter was published at ACL 2018 (Lyu and Titov, 2018). The code can be accessed from https://github.com/ChunchuanLv/AMR_AS_GRAPH_PREDICTION.

4.1 Preliminaries

We will use the following notation throughout the chapter. We refer to words in the sentences as $\mathbf{w} = (w_0, \ldots, w_{n-1})$, where n is sentence length and $w_k \in \mathcal{V}$. The concepts (i.e. labeled nodes) are $\mathbf{c} = (c_0, \ldots, c_{m-1})$, where m is the number of concepts and $c_i \in \mathcal{C}$. For example, in Figure 4.1, $\mathbf{c} = (obligate, go, boy, \text{-})$.[4] We have more words than nodes, so we append one $\emptyset$ at the end of the node list. Note that senses are predicted at post-processing, as discussed in Section 4.5 (i.e. *go* is labeled as *go-02*).

A relation between 'predicate concept' i and 'argument concept' j is denoted by $r_{ij} \in \mathcal{R}$; it is set to $\emptyset$ if j is not an argument of i. For example, in Figure 4.1, $r_{1,2} =$ *ARG0* and $r_{0,2} = \emptyset$. We will use R to denote all relations in the graph.

To represent alignments, we will use $\mathbf{a} = \{a_0, \ldots, a_{m-1}\}$, where $a_i \in \{0, \ldots, n-1\}$ returns the index of a word aligned to concept i. In our example, $a_0 = 2$.

[3] The standard deviation across multiple training runs was 0.16%.

[4] The probabilistic model is invariant to the ordering of concepts, though the order affects the inference algorithm (see Section 4.3). We use depth-first traversal of the graph to generate the ordering.

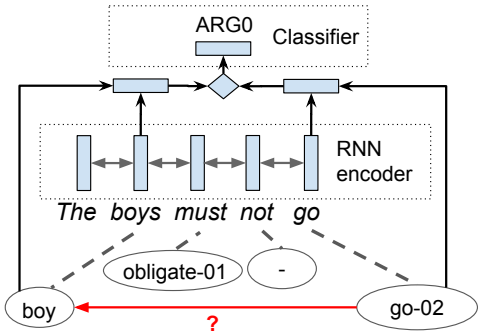


Figure 4.2: Relation identification: predicting a relation between *boy* and *go-02* relying on the two concepts and corresponding RNN states.

4.2 Our model

Our testing-time parser consists of two parts: (1) the concept identification model $P_\theta(\mathbf{c}|\mathbf{a},\mathbf{w})$; (2) the relation identification model $P_\theta(R|\mathbf{a},\mathbf{w},\mathbf{c})$. During training, we need another model to provide an estimation of the alignment: (3) the alignment model $Q_\phi(\mathbf{a}|\mathbf{c},R,\mathbf{w})$.[5] Formally, (1) and (2) together with the uniform prior over alignments $P(\mathbf{a})$ form the generative model of AMR graphs. In contrast, the alignment model $Q_\phi(\mathbf{a}|\mathbf{c},R,\mathbf{w})$, as will be explained below, is approximating the intractable posterior $P_\theta(\mathbf{a}|\mathbf{c},R,\mathbf{w})$ within that probabilistic model.

In other words, we assume the following model for generating the AMR graph:

$$P_\theta(\mathbf{c},R|\mathbf{w}) = \sum_{\mathbf{a}} P(\mathbf{a}) P_\theta(\mathbf{c}|\mathbf{a},\mathbf{w}) P_\theta(R|\mathbf{a},\mathbf{w},\mathbf{c}) \tag{4.1}$$

$$= \sum_{\mathbf{a}} P(\mathbf{a}) \prod_{i=1}^{m} P(c_i|\mathbf{h}_{a_i}) \prod_{i,j=1}^{m} P(r_{ij}|\mathbf{h}_{a_i},\mathbf{c}_i,\mathbf{h}_{a_j},\mathbf{c}_j) \tag{4.2}$$

AMR concepts are assumed to be generated independently relying on the BiLSTM states and surface forms of the aligned words. Similarly, relations are predicted based only on AMR concept embeddings and LSTM states corresponding to words aligned to the involved concepts. Their combined representations are fed into a bi-affine classifier (Dozat and Manning, 2017). See Figure 4.2 for a sketch.

The Equation 4.2 involves intractable marginalization over all valid alignments. We circumvent this marginalization by Monte-Carlo estimation and adopt the deep variational inference framework to reduce sampling variance. To perform gradient-based optimization with discrete latent variables, we use Gumbel-Sinkhorn to sample a differentiable alignment, where real-valued vectors $\hat{\mathbf{a}}_i \in \mathbb{R}^n$ (for every concept i) approximate discrete alignment variables a_i. This relaxation results in low-variance

[5] θ and ϕ denote all parameters of the models.

estimates of the gradient and ensures fast and stable training. We will describe the alignment model and the inference procedure in detail in Section 4.3.

While the estimation procedure requires the use of relaxation, the learned parser is straightforward to use. Given our assumptions about the alignments, we can independently choose for each word w_k ($k = 0, \ldots, n-1$) the most probably concept according to $P_\theta(c|\mathbf{h}_k)$. If the highest scored option is $\emptyset$, no concept is introduced. The relations could then be predicted relying on $P_\theta(R|\mathbf{a}, \mathbf{w}, \mathbf{c})$. This would have led to generating inconsistent AMR graphs, so instead, we search for the highest scored valid graph (see Section 4.5). Note that the alignment model Q_ϕ is not used at test time and only necessary to train accurate concept and relation identification models.

4.2.1 Concept Identification Model

The concept identification model chooses a concept c (i.e. a labeled node) conditioned on the aligned word k or decides that no concept should be introduced (i.e. returns $\emptyset$). Though it can be modeled with a softmax classifier, it would not be effective in handling rare or unseen words. First, we split the decision into estimating the probability of concept category $\tau(c) \in \mathcal{T}$ (e.g., 'number' or 'frame') and estimating the probability of the specific concept within the chosen category. Second, based on a lemmatizer and training data[6] we prepare one candidate concept e_k for each word k in vocabulary (e.g., it would propose *want* if the word is *wants*). Similar to Luong et al. (2015), our model can then either copy the candidate e_k or rely on the softmax over potential concepts of category τ. Formally, the concept prediction model is defined as

$$P_\theta(c|\mathbf{h}_k, w_k) = P(\tau(c)|\mathbf{h}_k, w_k) \times \frac{[[e_k = c]] \times \exp(\mathbf{v}_{\text{copy}}^T \mathbf{h}_\mathbf{k}) + \exp(\mathbf{v}_c^T \mathbf{h}_\mathbf{k})}{Z(\mathbf{h}_\mathbf{k}, \theta)}, \tag{4.3}$$

where the first multiplicative term is a softmax classifier over categories (including $\emptyset$); $\mathbf{v}_{\text{copy}}, \mathbf{v}_c \in \mathbb{R}^d$ (for $c \in \mathcal{C}$) are model parameters; $[[\ldots]]$ denotes the indicator function and equals 1 if its argument is true and 0, otherwise; $Z(\mathbf{h}, \theta)$ is the partition function ensuring that the scores sum to 1.

4.2.2 Relation Identification Model

We use the following arc-factored relation identification model:

$$P_\theta(R|\mathbf{a}, \mathbf{w}, \mathbf{c}) = \prod_{i,j=1}^{m} P(r_{ij}|\mathbf{h}_{a_i}, \mathbf{c}_i, \mathbf{h}_{a_j}, \mathbf{c}_j) \tag{4.4}$$

[6] Appendix B.1.

Each term is modeled in exactly the same way:

1. for both endpoints, embedding of the concept c is concatenated with the RNN state $\mathbf{h}$;

2. they are linearly projected to a lower dimension separately through $M_h(\mathbf{h}_{a_i} \circ c_i) \in \mathbb{R}^{d_f}$ and $M_d(\mathbf{h}_{a_j} \circ c_j) \in \mathbb{R}^{d_f}$, where $\circ$ denotes concatenation;

3. a log-linear model with bilinear scores $M_h(\mathbf{h}_{a_i} \circ c_i)^T C_r M_d(\mathbf{h}_{a_j} \circ c_j)$, $C_r \in \mathbb{R}^{d_f \times d_f}$ is used to compute the probabilities.

In the above discussion, we assumed that BiLSTM encodes a sentence once, and the BiLSTM states are then used to predict concepts and relations. In semantic role labeling, the task closely related to the relation identification stage of AMR parsing, a slight modification of this approach was shown more effective (Zhou and Xu, 2015; Marcheggiani et al., 2017). In that previous work, the sentence was encoded by a BiLSTM once per each predicate, and the encoding was in turn used to identify arguments of that predicate. The only difference across the re-encoding passes was a binary flag used as input to the BiLSTM encoder at each word position. The flag was set to 1 for the word corresponding to the predicate and to 0 for all other words. In that way, BiLSTM was encoding the sentence specifically for predicting arguments of a given predicate. Inspired by this approach, when predicting label r_{ij} for $j \in \{1, \dots m\}$, we input binary flags $\mathbf{p}_1, \dots \mathbf{p}_n$ to the BiLSTM encoder which are set to 1 for the word indexed by a_i ($\mathbf{p}_{a_i} = 1$) and to 0 for other words ($\mathbf{p}_j = 0$, for $j \neq a_i$). This also means that BiLSTM encoders for predicting relations and concepts end up being distinct. We use this multi-pass approach in our experiments.[7]

4.3 Estimating Latent Alignment

In this section, we show how to estimate the latent alignment jointly with the parser.

[7] Using the vanilla one-pass model from Equation (4.4) results in 1.4% drop in Smatch score.

4.3.1 Variational Inference

As we discussed earlier, we adopt the deep variational inference framework. We optimize a lower bound on the marginal likelihood objective:

$$\begin{aligned}&\log \sum_{\mathbf{a}} P(\mathbf{a}) P_\theta(\mathbf{c}|\mathbf{a},\mathbf{w}) P_\theta(R|\mathbf{a},\mathbf{w},\mathbf{c}) \\ \geq &\sum_{\mathbf{a}} Q_\phi(\mathbf{a}|\mathbf{c},R,\mathbf{w}) \log P_\theta(\mathbf{c}|\mathbf{a},\mathbf{w}) P_\theta(R|\mathbf{a},\mathbf{w},\mathbf{c}) - \mathrm{KL}(Q_\phi(\mathbf{a}|\mathbf{c},R,\mathbf{w})||P(\mathbf{a})),\end{aligned} \tag{4.5}$$

where $Q_\phi(\mathbf{a}|\mathbf{c},R,\mathbf{w})$ is the variational posterior parameterized by a neural network (the encoder, a.k.a., the inference network), and KL is the Kullback-Leibler divergence. In the deep variational inference framework, the lower bound is maximized both with respect to model parameters (θ) and the parameters of the inference network (ϕ). The Monte-Carlo gradient estimation with respect to θ is standard, but non-trivial for ϕ. We construct a differentiable sampling process for $Q_\phi(\mathbf{a}|\mathbf{c},R,\mathbf{w})$ in Section 4.3.2. Note that such a sampling procedure implicitly defines the variational distribution.

Recall that the alignment model is only used at training, and hence it can rely both on input (states $\mathbf{h}_1,\ldots,\mathbf{h}_n$) and on the list of concepts $c_1,\ldots,c_m$. Formally, we add $(m-n)$ null concepts ($\emptyset$) to the list. Aligning a word to any $\emptyset$, would correspond to saying that the word is not aligned to any 'real' concept. Note that each one-to-one alignment (i.e., permutation) between n such concepts and n words implies a valid injective alignment of n words to m 'real' concepts. This reduction to permutations will come in handy when we turn to the Gumbel-Sinkhorn relaxation. Given this reduction, from now on, we will assume that $m = n$.

4.3.2 Gumbel-Sinkhorn

In Gumbel-Sinkhorn, we estimate scores over individual alignments. As with sentences, we use a BiLSTM model to encode concepts $\mathbf{c}$, where $\mathbf{g}_i \in \mathbb{R}^{d_g}$, $i \in \{1,\ldots,n\}$. λ_{ij} score each alignment link between node i and token j according to a bilinear form and a mask based on the copy function:

$$\lambda_{ij} = \mathbf{g}_i^T B \mathbf{h}_j + \lambda_{ij}^{\text{mask}}, \tag{4.6}$$

where $B \in \mathbb{R}^{d_g \times d}$ is a parameter matrix. If a node is copy-able from at least one token, the alignment mask prohibits alignments from other tokens by setting the corresponding components $\lambda_{ij}^{\text{mask}}$ to $-\infty$, otherwise it is 0. Now, we perturb the score and choosing

the highest scoring one in the convex hull relaxation of all permutations:

$$\hat{\mathbf{a}} = \arg\max_{\mathbf{a}\in\mathcal{P}} \langle \lambda + \varepsilon, \mathbf{a} \rangle - \tau \langle \mathbf{a}, \log \mathbf{a} \rangle, \tag{4.7}$$

where $\mathcal{P} = \{\mathbf{a} \in \{0,1\}^{n\times n} | \forall j, \sum_i \mathbf{a}_{ij} = 1; \forall i, \sum_j \mathbf{a}_{ij} = 1\}$ is the convex hull of all permutations of n elements, and ε_{ij} is a noise drawn independently from the fixed Gumbel distribution ($\mathcal{G}(0,1)$). The optimization is solved by Sinkhorn algorithm (as described in the Section 3.2.3) with t finite steps, and the solution is denoted as $S_t(\lambda, \varepsilon)$. Instead of returning index a_i for every concept i, $S_t(\lambda,\varepsilon)_i$ would return a (peaky) distribution over words. The peakiness is controlled by the temperature parameter τ of Gumbel-Sinkhorn which balances smoothness ('differentiability') vs. bias of the estimator. In all, Gumbel-Sinkhorn yields an approximate differentiable sample $\hat{\mathbf{a}} = S_t(\theta, \varepsilon)$.

Using the Gumbel-Sinkhorn construction unfortunately does not guarantee that $\sum_i \hat{\mathbf{a}}_{ij} = 1$.[8] To encourage this equality to hold, and equivalently to discourage overlapping alignments, we add another regularizer to the objective (4.9):

$$\Omega(\hat{\mathbf{a}}, \lambda) = \lambda \sum_j \max(\sum_i (\hat{\mathbf{a}}_{ij}) - 1, 0). \tag{4.8}$$

In addition, following Mena et al. (2018), the original KL term from Equation (4.5) is approximated by the KL term between two $n \times n$ matrices of i.i.d. Gumbel distributions with different temperatures and mean. The parameter τ_0 is the 'prior temperature'. The final objective (to be maximized) can now be approximated as

$$\underset{\varepsilon\sim\mathcal{G}(0,1)}{E} [\log P_\theta(\mathbf{c}|S_t(\lambda,\varepsilon),\mathbf{w}) + \log P_\theta(R|S_t(\lambda,\varepsilon),\mathbf{w},\mathbf{c}) - \Omega(\hat{\mathbf{a}},\lambda)] - \mathrm{KL}(\frac{\lambda+\varepsilon}{t} || \frac{\varepsilon}{\tau_0}) \tag{4.9}$$

Our final objective is fully differentiable with respect to all parameters (i.e. θ and ϕ). Still, we need to understand how to use the the relaxed alignment.

4.3.3 Relaxing Concept and Relation Identification

One remaining question is how to use the soft input $\hat{\mathbf{a}} = S_t(\lambda, \varepsilon)$ in the concept and relation identification models in Equation (4.9). In other words, we need to define how we compute $P_\theta(\mathbf{c}|S_t(\lambda,\varepsilon),\mathbf{w})$ and $P_\theta(R|S_t(\lambda,\varepsilon),\mathbf{w},\mathbf{c})$.

The standard technique would be to pass to the models expectations under the relaxed variables $\sum_{k=1}^{n} \hat{\mathbf{a}}_{ik}\mathbf{h}_k$, instead of the vectors $\mathbf{h}_{a_i}$ (Maddison et al., 2017; Jang et al., 2017). This is what we do for the relation identification model. We use this approach also to relax the one-hot encoding of the predicate position ($\mathbf{p}$, see Section 4.2.2).

[8] This was a comprise between number of Sinkhorn steps and the limited GPU memory at the time.

However, the concept prediction model $\log P_\theta(\mathbf{c}|S_t(\lambda,\varepsilon),\mathbf{w})$ relies on the pointing mechanism, i.e. directly exploits the words $\mathbf{w}$ rather than relies only on biLSTM states $\mathbf{h}_k$. So instead we treat $\hat{\mathbf{a}}_i$ as a prior in a hierarchical model:

$$\log P_\theta(\mathbf{c}_i|\hat{\mathbf{a}}_\mathbf{i},\mathbf{w}) \approx \log \sum_{k=1}^{n} \hat{\mathbf{a}}_{ik} P_\theta(\mathbf{c}_i|a_i=k,\mathbf{w}) \tag{4.10}$$

As we will show in our experiments, a softer version of the loss is even more effective:

$$\log P_\theta(\mathbf{c}_i|\hat{\mathbf{a}}_\mathbf{i},\mathbf{w}) \approx \log \sum_{k=1}^{n} (\hat{\mathbf{a}}_{ik} P_\theta(\mathbf{c}_i|a_i=k,\mathbf{w}))^\alpha, \tag{4.11}$$

where we set the parameter $\alpha = 0.5$. We believe that using this loss encourages the model to more actively explore the alignment space. Geometrically, the loss surface shaped as a ball in the 0.5-norm space would push the model away from the corners, thus encouraging exploration.[9]

4.4 Re-categorization

As we discussed in Section 2.2.1, there are many AMR semantic phenomena that have multiple concept nodes anchored to one token. This breaks the injective assumption we made. Following Werling et al. (2015); Foland and Martin (2017); Wang and Xue (2017); Peng et al. (2017), we propose a rule-based re-categorization system that converts the identification of several concept nodes into the identification of a subgraph.. The original AMR nodes can be categorized into five categories: frame (e.g., *opine-01*), basic concept (e.g., *thing*), string ("Johnson"), number (e.g., 5) and other constant (e.g., '-'). Our re-categorization system groups specific subgraphs of AMR into a single (re-categorized) node with a new compound category. During training, the system is used in the pre-processing stage, and all nodes in each subgraph share the same alignment to the RNN states for relation identification. During testing, we unpack our concepts before the relation identification stage, so the relations are predicted between original concepts.

Intuitively, the goal is to ensure concepts that are rarely lexically triggered (e.g., *thing* in Figure 4.3) get grouped together with lexically triggered nodes. Such 'primary' concepts get encoded in the category of the concept (the set of categories is τ,

[9] In hindsight, this particular relaxation might be our ad-hot strategy to close the gap between the training and testing time discrepancy. A more systematic approach should be using the straight-through estimator as we discussed in Section 3.

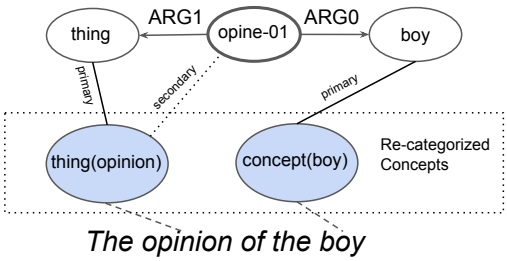


Figure 4.3: An example of re-categorized AMR. AMR graph at the top, re-categorized concepts in the middle, and the sentence is at the bottom.

primary	secondary	re-categorized
person	ARG0-of/ARG1-of	person([second])
thing	ARG0-of/ARG1-of/ARG2-of	thing([second])
most	degree-of	most([second])
-quantity	unit	primary([second])
-entity	value	primary([second])
date-entity	weekday/dayperiod/season	date-entity([second])
monetary-quantity	unit/ARG2-of/ARG1-of/quant	monetary-quantity([second])
temporal-quantity	unit/ARG3-of	temporal-quantity([second])

Table 4.1: Templates for re-categorization.

see also section 4.2.1). In Figure 4.3, the re-categorized concept *thing(opinion)* is produced from *thing* and *opine-01*. We use *concept* as the dummy category type. There are 8 templates in our system which extract re-categorizations for fixed phrases (e.g. *thing(opinion)*), and a deterministic system for grouping lexically flexible, but structurally stable sub-graphs (e.g., named entities, *have-rel-role-91* and *have-org-role-91* concepts).

For fixed phrases, re-categorization is handled with rules listed in Table 4.1. They are triggered if a given primary concept ('primary') appears adjacent to edges labeled with relations given in column 'secondary'.[10] The assigned category is shown in column 're-categorized'. The rules yield 32 categories when applied to the training set.

For more flexible subgraphs that we need for named entities and role special frames (i.e., *have-rel-role-91* and *have-org-role-91*), those are additional rules shown in Table 4.2. As we could have more than one node that is not lexically triggered. Similar to a 'secondary' column, we need an additional 'tertiary' column. These rules yield

[10]Edges are sorted according to priority. If one edge has been invoked, the rest won't be used.

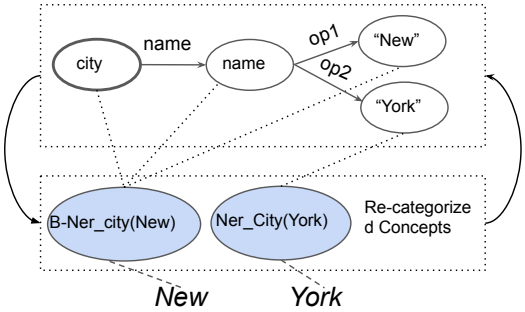


Figure 4.4: An example of re-categorized AMR. AMR graph at the top, re-categorized concepts in the middle, and the sentence is at the bottom.

109 additional categories (mostly due to different named entity types) when applied to the training set.

primary	secondary	tertiary	re-categorized
name	op1	name-of	B-Ner_third([second])
have-org-role-91	ARG2/ARG1	ARG0-of	have-org-role-91_person([second])
have-rel-role-91	ARG2/ARG1	ARG0-of	have-rel-role-91_person([second])

Table 4.2: Flexible Templates for re-categorization.

The re-categorized named entity has an extra 'third' in the category. This is because the type of named entity is an open vocabulary. Consequently, we use an extra classifier for identifying the named entity type. As for two special frames, the 'third' nodes are always *person*. Still, for named entities, we could have multiple tokens. The other tokens are also converted to re-categorized nodes. We show this in Figure 4.4. During testing, the distinction between 'B-Ner' and 'Ner' for the BIO decoding scheme, and we will have one named entity subgraph.

As one can see, this re-categorization system is intricate and covers many individual AMR phenomena we discussed in Section 2.2.1. Indeed, we hand-crafted this system through a detailed analysis of AMR phenomena. Naturally, such rule construction process needs to be repeated for different meaning representations and different languages (Anchiêta and Pardo, 2018; Migueles-Abraira et al., 2018; Song et al., 2020).[11] For example, Spanish AMR (Migueles-Abraira et al., 2018) contains new subgraphs dealing with pronouns that are often semantically underspecified (e.g., 'su' ('her/his/its/your')[12]). Identifying those new subgraphs would require new rules. In

[11]The extent to which non-English AMR sembanks differ from the English AMR sembank varies.

[12]A concept node 'ente' ('being') is introduced for pronouns, and ':sinespecificar' is a relation sig-

the next chapter, we avoid such rule construction process with a data-driven method.

4.5 Pre-and-Post Processing

We start with the tokenized dataset of Pourdamghani et al. (2014b). We take all dashed AMR concepts (e.g., *make-up* and *more-than*) and concatenate the corresponding spans (based on statistics from the training set and PropBank frame files). We also combine spans of words corresponding to a single number. For relation identification, we normalize relations to one canonical direction (e.g., arg0, time-of). For named entity recognition and lemmatization, we use Stanford CoreNLP toolkit (Manning et al., 2014). For pre-trained embedding, we used Glove (300 dimension embeddings) (Pennington et al., 2014).

For post-processing, we handle sense-disambiguation, wikification, and ensure legitimacy of the produced AMR graph. For sense disambiguation, we pick the most frequent sense for that particular concept ('-01', if unseen). For wikification, we again look-up in the training set and default to '-'. There is certainly room for improvement in both stages. Our probability model predicts edges conditional independently and thus cannot guarantee the connectivity of AMR graph. Also, there are additional constraints that are useful to impose. We enforce three constraints: (1) 'number', 'string' and '-' concepts can have only one neighbor; (2) each predicate concept can have at most one argument for each relation; (3) the graph should be connected. Constraint (1) is addressed by keeping only the highest scoring neighbor. To satisfy the last two constraints, we use a simple greedy procedure. First, for each edge, we pick-up the highest scored relation and edge (possibly $\emptyset$). If constraint (2) is violated, we keep the highest scored edge among the duplicates and drop the rest. If the graph is not connected (i.e., constraint (3) is violated), we greedily choose edges linking the connected components until the graph gets connected (MSCG in Flanigan et al. (2014)).

Finally, we need to select a root node. Similarly to relation identification, for each candidate concept c_i, we concatenate its embedding with the corresponding LSTM state ($\mathbf{h}_{a_i}$) and use these scores in a softmax classifier over all the concepts.

naling underspecification. '(e / ente :sinespecificar (s / su))' represents 'su'.

4.6 Experiments and Discussion

We primarily focus on the most recent LDC2016E25 (R2) dataset, which consists of 36521, 1368, and 1371 sentences in training, development, and testing sets, respectively. The earlier LDC2015E86 (R1) dataset has been used by much of the previous work. It contains 16833 training sentences and the same sentences for development and testing as R2.

We selected hyper-parameters based on the best performance on the development set. For all the ablation tests, the hyper-parameters are fixed. We used 2 different BiLSTM encoders of the same hyper-parameters to encode sentences for concept identification and alignment prediction, another BiLSTM to encode AMR concept sequence for alignment, and finally, 2 different BiLSTM of the same hyper-parameters to encode sentence for relation identification and root identification. There are 5 BiLSTM encoders in total. The exact hyper-parameters, as well as information about embeddings, are presented in the Appendix B.2.

We used Adam (Kingma and Ba, 2015) to optimize the loss (4.9) and to train the root classifier. Our best model is trained fully jointly, and we do early stopping on the development set scores. Training takes approximately 6 hours on a single GeForce GTX 1080 Ti with Intel Xeon CPU E5-2620 v4.

4.6.1 Results and Discussion

We start by comparing our parser to previous work (see Table 4.3).[13] Our model substantially outperforms all the previous models on both datasets. Specifically, it achieves 74.4% Smatch score on LDC2016E25 (R2), which is an improvement of 3.4% over character seq2seq model relying on silver data (van Noord and Bos, 2017). For LDC2015E86 (R1), we obtain 73.7% Smatch score, which is an improvement of 3.0% over the previous best model, multi-BiLSTM parser of Foland and Martin (2017).

In order to disentangle individual phenomena, we use the AMR-evaluation tools (Damonte et al., 2017) and compare to systems which reported these scores (Table 4.4). We obtain the highest scores on most subtasks. The exception is negation detection. However, this is not too surprising as many negations are encoded with morphology, and character models, unlike our word-level model, are able to capture predictive morphological features (e.g., detect prefixes such as “un-” or “im-”).

[13] We include more recent results in the next chapter.

Model	Data	Smatch
JAMR (Flanigan et al., 2016)	R1	67.0
AMREager (Damonte et al., 2017)	R1	64.0
CAMR (Wang et al., 2016)	R1	66.5
SEQ2SEQ + 20M (Konstas et al., 2017)	R1	62.1
Mul-BiLSTM (Foland and Martin, 2017)	R1	70.7
Ours	R1	**73.7**
Neural-Pointer (Buys and Blunsom, 2017)	R2	61.9
ChSeq (van Noord and Bos, 2017)	R2	64.0
ChSeq + 100K (van Noord and Bos, 2017)	R2	71.0
Ours	R2	**74.4** $\pm$ 0.16

Table 4.3: Smatch scores on the test set. R2 is LDC2016E25 dataset, and R1 is LDC2015E86 dataset. Statistics on R2 are over 8 runs.

Now, we turn to ablation tests (see Table 4.5). First, we would like to see if our latent alignment framework is beneficial. To test this, we create a baseline version of our system ('pre-align'), which relies on the JAMR aligner (Flanigan et al., 2014), rather than induces alignments as latent variables. Recall that in our model, we used training data and a lemmatizer to produce candidates for the concept prediction model (see Section 4.2.1, the copy function). To have a fair comparison, if a concept is not aligned after JAMR, we try to use our copy function to align it. If an alignment is not found, we make the alignment uniform across the unaligned words. In preliminary experiments, we considered alternatives versions (e.g., dropping concepts unaligned by JAMR or dropping concepts unaligned after both JAMR and the matching heuristic), but the chosen strategy was the most effective. These scores of pre-align are superior to the results from Foland and Martin (2017), which also relies on JAMR alignments and uses BiLSTM encoders. There are many potential reasons for this difference in performance. For example, their relation identification model is different (e.g., single-pass, no bi-affine modeling), they used much smaller networks than us, they use plain JAMR rather than a combination of JAMR and our copy function, they use a different re-categorization system. These results confirm that we started with a strong basic model and that our variational alignment framework provided further gains in performance.

Now we would like to confirm that joint training of alignments with both concepts

Models	A'	C'	J'	Ch'	Ours
	17	16	16	17	
Dataset	R1	R1	R1	R2	R2
Smatch	64	63	67	71	**74.4**±0.16
Unlabeled	69	69	69	74	**77.1**±0.10
No WSD	65	64	68	72	**75.5**±0.12
Reentrancy	41	41	42	**52**	**52.3**±0.43
Concepts	83	80	83	82	**85.9**±0.11
NER	83	75	79	79	**86.0**±0.46
Wiki	64	0	75	65	**75.7**±0.30
Negations	48	18	45	**62**	58.4±1.32
SRL	56	60	60	66	**69.8**±0.24

Table 4.4: F1 scores on individual phenomena. A'17 is AMREager, C'16 is CAMR, J'16 is JAMR, Ch'17 is ChSeq+100K. Ours are marked with standard deviation.

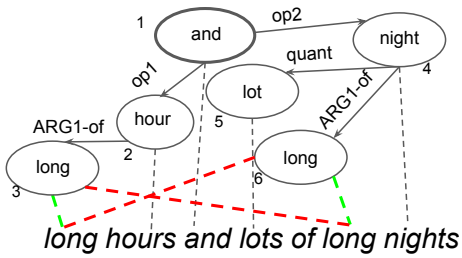


Figure 4.5: When modeling concepts alone, the posterior probability of the correct (green) and wrong (red) alignment links will be the same.

and relations is beneficial. In other words, we would like to see if alignments need to be induced in such a way as to benefit the relation identification task. For this ablation, we break the full joint training into two stages. We start by jointly training the alignment model and the concept identification model. When these are trained, we optimize the relation model but keep the concept identification model and alignment models fixed ('2 stages' in Table 4.6). When compared to our joint model ('full model'), we observe a substantial drop in Smatch score (-0.8%). In another version ('2 stages, tune align') we also use two stages, but we fine-tune the alignment model on the second stage. This approach appears slightly more accurate but still -0.5% below the full model. In both cases, the drop is more substantial for relations ('SRL'). In order to see why relations are potentially useful in learning alignments, consider Figure 4.5. The example contains duplicate concepts *long*. The concept prediction model factorizes

Metric	R1		R2	
	Pre-Align	Latent	Pre-Align	Latent
Smatch	72.8	73.7	73.5	**74.4**
Unlabeled	75.3	76.3	76.1	**77.1**
No WSD	73.8	74.7	74.6	**75.5**
Reentrancy	50.2	50.6	**52.6**	52.3
Concepts	85.4	85.5	85.5	**85.9**
NER	85.3	84.8	85.3	**86.0**
Wiki	66.8	75.6	67.8	**75.7**
Negations	56.0	57.2	56.6	**58.4**
SRL	68.8	68.9	**70.2**	69.8

Table 4.5: F1 scores of on subtasks. Scores on ablations are averaged over 2 runs. The left side results are from LDC2015E86 and right results are from LDC2016E25.

Ablation	Concepts	SRL	Smatch
2 stages	85.6	68.9	73.6
2 stages, tune align	85.6	69.2	73.9
Full model	**85.9**	**69.8**	**74.4**

Table 4.6: Ablation studies: effect of joint modeling (all on R2). Scores on ablations are averaged over 2 runs. The first two models load the same concept and alignment model before the second stage.

over concepts and does not care which way these duplicates are aligned: correctly (green edges) or not (red edges). Formally, the true posterior under the concept-only model in '2 stages' assigns exactly the same probability to both configurations, and the alignment model Q_ϕ will be forced to mimic it (even though it relies on an LSTM model of the graph). The spurious ambiguity will have a detrimental effect on the relation identification stage.

It is interesting to see the contribution of other modeling decisions we made when modeling and relaxing alignments. Instead of using Gumbel-Sinkhorn, which encourages mutually-repulsive alignments, we now use a factorized alignment model. Note that this model ('No Sinkhorn' in Table 4.7) still relies on (relaxed) discrete alignments (using Gumbel softmax) but does not constrain the alignments to be injective. A substantial drop in performance indicates that prior knowledge about the nature of alignments appears beneficial. Second, we remove the additional regularizer for Gumbel-

Ablation	Concepts	SRL	Smatch
No Sinkhorn	85.7	69.3	73.8
No Sinkhorn reg	85.6	69.5	74.2
No soft loss	85.2	69.1	73.7
Full model	**85.9**	**69.8**	**74.4**

Table 4.7: Ablation studies: alignment modeling and relaxation (all on R2). Scores on ablations are averaged over 2 runs.

Sinkhorn approximation (Equation (4.8)). The performance drop in Smatch score ('No Sinkhorn reg') is only moderate. Finally, we show that using the simple hierarchical relaxation (Equation (4.10)) rather than our softer version of the loss (Equation (4.11)) results in a substantial drop in performance ('No soft loss', -0.7% Smatch). We hypothesize that the softer relaxation favors the exploration of alignments and helps to discover better configurations.

4.7 Related Work

Alignment performance has been previously identified as a potential bottleneck affecting AMR parsing (Damonte et al., 2017; Foland and Martin, 2017). Some recent work has focused on building aligners specifically for training their parsers (Werling et al., 2015; Wang and Xue, 2017). However, those aligners are trained independently of concept and relation identification and only used at pre-processing.

Treating alignment as discrete variables has been successful in some sequence transduction tasks with neural models (Yu et al., 2017, 2016). Our work is similar in that we also train discrete alignments jointly, but the tasks, the inference framework, and the decoders are very different.

The discrete alignment modeling framework has been developed in the context of traditional (i.e., non-neural) statistical machine translation (Brown et al., 1993). Such translation models have also been successfully applied to semantic parsing tasks (e.g., (Andreas et al., 2013)), where they rivaled specialized semantic parsers from that period. However, they are considerably less accurate than current state-of-the-art parsers applied to the same datasets (e.g., (Dong and Lapata, 2016)).

For AMR parsing, another way to avoid using pre-trained aligners is to use seq2seq models (Konstas et al., 2017; van Noord and Bos, 2017). In particular, van Noord and Bos (2017) used character level seq2seq model and achieved the previous state-of-the-

art result. However, their model is very data demanding as they needed to train it on additional 100K sentences parsed by other parsers. This may be due to two reasons. First, seq2seq models are often not as strong on smaller datasets. Second, recurrent decoders may struggle with predicting the linearized AMRs, as many statistical dependencies are highly non-local.

Recently, the neural auto-regressive models (Zhang et al., 2019a; Cai and Lam, 2020) rely on soft attention instead of discrete alignment, and they have suppressed our parser in terms of performance without additional AMR data. We conjecture the improvements come from pre-trained contextualized sentence representations such as BERT (Devlin et al., 2019), which is trained on a huge amount of unlabeled data.

4.8 Summary

In this chapter, we introduced a neural AMR parser trained by jointly modeling alignments, concepts, and relations. We make such joint modeling computationally feasible by using deep variational inference and the Gumbel-Sinkhorn technique. The parser significantly improves the previous state of the art, and ablation tests show that joint modeling is indeed beneficial.

However, we still heavily rely on the re-categorization system that collapsing AMR subgraphs into nodes. The system is rule-based and needs to be re-designed for other meaning representations. In fact, it needs to be adjusted when new AMR semantic phenomena are introduced. In the next chapter, we show how to get rid of the rule-based recategorization stage, and learn graph-segmentation.

Chapter 5

AMR Parsing with Latent Alignment and Graph Segmentation

In this chapter, we train a graph-based parser without relying on an external aligner or a rule-based re-categorization system.

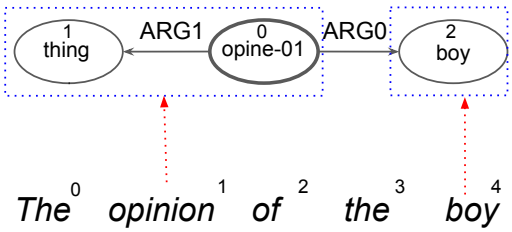


Figure 5.1: An example of AMR, the dashed red arrows mark latent alignment. Dashed blue boxes represent the latent graph segmentation.

An AMR graph can be regarded as consisting of multiple concept subgraphs, which can be individually aligned to sentence tokens (Flanigan et al., 2014). In Figure 5.1, each dashed box represents the boundary of a single semantic subgraph. Red arrows represent the alignment between subgraphs and tokens. For example, '(o / opine-01: ARG1 (t / thing))' refers to a combination of the predicate 'opine-01' and a filler of its semantic role ARG1. Intuitively, this subgraph needs to be aligned to the token 'opinion'. Similarly, '(b / boy)' should be aligned to the token 'boy'. Given such an alignment and segmentation, it is straightforward to construct a simple *graph-based* parser; for example, parsing can be framed as tagging input tokens with subgraphs (including empty subgraphs), followed by predicting relations between the subgraphs. The key obstacle to training an AMR parser is that the segmentation and alignment between AMR subgraphs and words are latent, i.e., not annotated in the data.

We propose to optimize a graph-based parser while treating alignment and graph segmentation as latent variables. A graph-based parser consists of two parts: concept identification and relation identification. The concept identification model generates the AMR nodes, and the relation identification component decides on the labeled edges. During training, both components rely on latent alignment and segmentation, which is being induced simultaneously. Importantly, at test time, the parser simply tags the input with the subgraphs and predicts the relations, so there is no test-time overhead from using the latent-structure apparatus. An extra benefit of this approach, in contrast to encoder-decoder AMR models (Konstas et al., 2017; van Noord and Bos, 2017; Cai and Lam, 2020) is its transparency, as one can readily see which input token triggered each subgraph.

To train most AMR parsers, due to one-to-many mapping between word and AMR concepts, one needs to segment the graph into subgraphs and align each such subgraph to a word in a sentence (Flanigan et al., 2014; Werling et al., 2015; Damonte et al., 2017; Ballesteros and Al-Onaizan, 2017; Peng et al., 2015; Artzi et al., 2015; Groschwitz et al., 2018); this is normally done at preprocessing, relying on hand-crafted rules. As more constructions are getting introduced to AMRs (Bonial et al., 2018) and AMR sembanks in languages other than English are being developed (Anchiêta and Pardo, 2018; Migueles-Abraira et al., 2018; Song et al., 2020), getting rid of the rules and learning graph segmentation from scratch becomes a compelling problem to tackle.

To achieve our goal, we frame the alignment and segmentation problems as choosing a *generation order* of concept nodes, as we explain in Section 5.1.2. As marginalizing over the structured latent variables is infeasible, we use the deep variational inference (VI) framework. To ensure end-to-end differentiable optimization, we adopt *stochastic softmax* (See Section 3.2.1) to sample the segmentation and alignment. Furthermore, to efficiently apply the stochastic softmax, we derive an inference algorithm for our problem; it can be regarded as an instance of the Bregman's method (Bregman, 1967). As a result, our model is end-to-end differentiable.

We experiment on the AMR 2.0 and 3.0 datasets. In particular, we present a *greedy segmentation* heuristic, inspired by Naseem et al. (2019), that produces a segmentation deterministically and provides a strong baseline to our segmentation induction method. We also compare against a hand-crafted rule-based segmentation system that is used in recent work.[1] On AMR 2.0 (LDC2016E25), we found that our VI system

[1]This is based on the re-categorization system we developed in Chapter 4. The rules are crafted for

obtained a competitive Smatch score of 76.1, while the greedy segmentation and the hand-crafted rule-based segmentation obtain 75.2 and 76.8, respectively. On AMR 3.0 (LDC2020T02), the VI system, the greedy segmentation, and the rule-based segmentation yield 75.5, 74.7, and 75.7, respectively. In other words, our method approaches the performance of the rule-based technique, even though it does not exploit the prior knowledge about AMR used to construct the rules. Our main contributions are:

- We frame the alignment and segmentation problems as inducing a generation order;
- We adopt the stochastic softmax (Paulus et al., 2020) to estimate the latent generation order and derive an efficient optimizer, an instance of Bregman's method (Bregman, 1967);
- We empirically show that our method outperforms a strong heuristic baseline and approaches the performance of a hand-crafted rule system.

The code can be accessed from https://github.com/ChunchuanLv/graph-parser.

5.1 Casting Alignment and Segmentation as Choosing a Generation Order

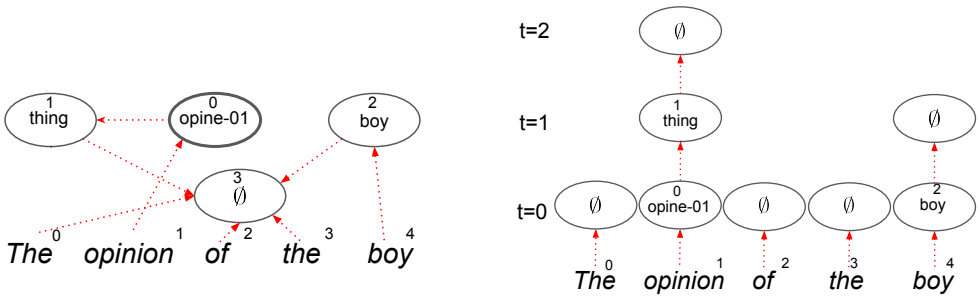
5.1.1 Preliminaries

We introduce the basic concepts and notation here. We refer to words in a sentence as $\mathbf{x} = (x_0, \ldots, x_{n-1})$, where n is the sentence length. The concepts (i.e. labeled nodes) are $\mathbf{v} = (v_0, v_1, \ldots, v_m)$, where m is the number of concepts. In particular, $v_m = \emptyset$ denotes a dummy terminal node (its purpose will be clear later); we refer to all nodes, except for the terminal node ($\emptyset$), as concept nodes.

A relation between 'predicate concept' i and 'argument concept' j is denoted by $\mathbf{E}_{ij} \in \mathcal{E}$; it is set to $\emptyset$ if j is not an argument of i. We will use $\mathbf{E}$ to denote all edges (i.e., relations) in the graph. In addition, we refer to the whole graph as $\mathbf{G} = (\mathbf{v}, \mathbf{E})$.

Our goal is to associate each input token with a (potentially empty) subset of the nodes of the graph, while making sure that we get a *partition* of the node set. In other words, each node in the original graph belongs to exactly one subset. In that way, we deal with both segmentation and alignment. Each subset uniquely corresponds to a

individual AMR constructions. Then, the subsequent work extends the rules (Zhang et al., 2019a; Cai and Lam, 2020). We adopt the code from Zhang et al. (2019a).



(a) Training (b) Test

Figure 5.2: Figure 5.2a shows AMR concept identification model generates nodes following latent generation order at training time. Figure 5.2b the AMR concept identification model generates nodes autoregressively, starting from each sentence token at test time.

vertex-induced subgraph (i.e., the subset of nodes together with any edges whose both endpoints are in this subset). For this reason, we will refer to the problem as graph decomposition[2] and to each subset as a subgraph. We will explain how we deal with edges in subsequent sections.

5.1.2 Generation Order

Instead of only selecting a subset of nodes for each token, we also select the order in which nodes in the subset are chosen. In Figure 5.2a, dashed red arrows point from every node to the subsequent node to be selected. For example, given the word 'opinion', the node 'opine-01' is chosen first, and then it is followed by another node 'thing'. After this node, we have an arrow pointing to the node ∅, signifying the termination. We refer to these red arrows as a *generation order*. To recover a graph decomposition from a generation order, we assign connected nodes (excluding the terminal node) to the same subgraph. Then, a subgraph will be aligned to the token that generated those nodes. In our example, 'opine-01' and 'thing' are connected, and, thus, they are both aligned to the word 'opinion'. The alignment is encoded by arrows between tokens and concept nodes, while the segmentation is represented by arrows between concept nodes.[3]

From a modeling perspective, the nodes will be generated with an autoregressive model. Thus, we do not have to represent and store all potential subgraphs in our model

[2] We slightly abuse the terminology as, in graph theory, graph decomposition usually refers to a partition of edges of the original graph.

[3] Alternatively, in the opposite direction, one could consider the generation order as the anchoring of nodes. The trick here is that we allow the nodes to be anchored to another node, in addition to words.

explicitly. It is easily possible to apply such an autoregressive concept identification model at test time (Figure 5.2b). From each token, a chain of nodes is generated until a $\emptyset$ node is predicted. It is more challenging to see how to induce the order and train the autoregressive model at the same time; we will discuss this in Sections 5.2 and 5.3.

While in Figure 5.2a the red arrows yield a valid generation order, the arrows need to obey certain constraints. Formally, we denote alignment by $\mathbf{A} \in \{0,1\}^{n\times(m+1)}$, where $\mathbf{A}_{ki} = 1$ means that for token k we start by generating node i. As the token can only point to one node, we have a constraint $\sum_i \mathbf{A}_{ki} = 1$. Similarly, we have segmentation $\mathbf{S} \in \{0,1\}^{m\times(m+1)}$ with a constraint $\sum_j \mathbf{S}_{ij} = 1$. Here, $\mathbf{S}_{ij} = 1$ encodes that node i is followed by node j. In Figure 5.2a, we have $\mathbf{A}_{03} = \mathbf{A}_{10} = \mathbf{A}_{23} = \mathbf{A}_{33} = \mathbf{A}_{42} = 1$ and $\mathbf{S}_{01} = \mathbf{S}_{13} = \mathbf{S}_{23} = 1$; the rest is 0. Now, we have the full generation order as their concatenation $\mathbf{O} = \mathbf{A} \circ \mathbf{S} \in \{0,1\}^{(n+m)\times(m+1)}$. As one node can only be generated once (except for $\emptyset$), we have a joint constraint: $\forall j \neq m, \sum_l \mathbf{O}_{lj} = 1$. Furthermore, the graph defined by $\mathbf{O}$ should be acyclic, as it represents the generative process. We denote the set of all valid generation orders as O. In the following sections, we will discuss how this generation order is used in the model and how to infer it as a latent variable while enforcing the above constraints.

5.2 Our Model

Formally, we aim at estimating $P_\theta(\mathbf{v}, \mathbf{E}|\mathbf{x})$, the likelihood of an AMR graph given the sentence. Our graph-based parser is composed of two parts: concept identification $P_\theta(\mathbf{v}|\mathbf{x}, \mathbf{O})$ and relation identification $P_\theta(\mathbf{E}|\mathbf{x}, \mathbf{O}, \mathbf{v})$. The concept identification model generates concept nodes, and the relation identification model assigns relations between them. Both require the latent generation order at the training time, denoted by $\mathbf{O}$. Overall, we have the following objective:

$$\log P_\theta(\mathbf{v}, \mathbf{E}|\mathbf{x}) = \log \sum_{\mathbf{O}} P_\theta(\mathbf{O}) P_\theta(\mathbf{v}|\mathbf{x}, \mathbf{O}) P_\theta(\mathbf{E}|\mathbf{x}, \mathbf{O}, \mathbf{v}) \ , \tag{5.1}$$

where $P_\theta(\mathbf{O})$ is a prior on the generation order (we discuss it in Section 5.3.2). To efficiently optimize this objective end-to-end, we will use tools from variational inference and also *stochastic softmax*, which will be explained in Section 5.3. One important requirement for applying the stochastic softmax is that both concept and relation identification models admit differentiable relaxation, i.e., the probability $P_\theta(\mathbf{v}|\mathbf{x}, \mathbf{O})$ and $P_\theta(\mathbf{E}|\mathbf{x}, \mathbf{O}, \mathbf{v})$ should be differentiable w.r.t. real-valued $\mathbf{O}$. While the model relaxation is relatively straightforward in the last chapter, the flexible generation order makes the

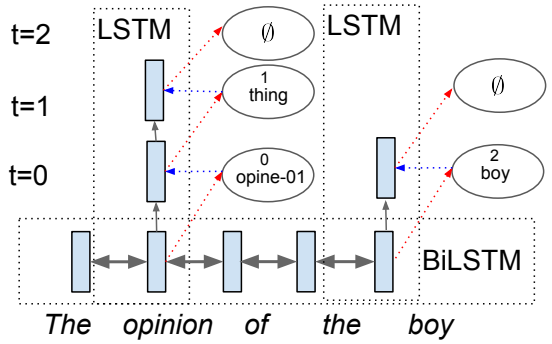


Figure 5.3: AMR concept identification model runs several independent LSTMs to generate nodes autoregressively at test time.

model relaxation quite involved. In the following subsections, we go through concept identification, relation identification, and their corresponding relaxations.

5.2.1 Concept Identification

As shown in Figure 5.3, our neural model first encodes the sentence with BiLSTM, producing token representations $\mathbf{h}_k^{\text{token}}$ ($k \in [0, \dots n-1]$),[4] then generates nodes autoregressively at each token with another LSTM. At the testing time, we perform greedy decoding to generate nodes from each token in parallel until either terminal node or T nodes are generated.

For training, we need to be able to run the models with any potential generation order and compute $P_\theta(\mathbf{v}|\mathbf{x}, \mathbf{O})$. If we take the order defined in Figure 5.2a, the node 1 ('thing') is predicted relying on the corresponding hidden representation; we refer to this representation as $\mathbf{h}_1^{\text{node}}$ where is 1 is the node index. With the discrete generation order defined by red arrows in Figure 5.2a, $\mathbf{h}_1^{\text{node}}$ is just the LSTM state of its parent (i.e. 'opine-01'). Importantly, our computation should be well-defined when the generation order $\mathbf{O}$ is soft (i.e. attention-like rather than pointer-like). In that case, $\mathbf{h}_1^{\text{node}}$ will be a weighted sum of LSTM representations of other nodes and input tokens, where the weights are defined by $\mathbf{O}$. Similarly, the termination symbol $\emptyset$ for the token 'opinion' is predicted from its hidden representation; we refer to this representation as $\mathbf{h}_1^{\text{tail}}$, where 1 is the position of 'opine' in the sentence. With the hard generation order of Figure 5.2a, $\mathbf{h}_1^{\text{tail}}$ is just the LSTM state computed after choosing the preceding node (i.e. 'thing'). In the relaxed case, it will again be a weighted sum with the weights

[4] Technically, we have two token encoders, one for concept identification and one for relation identification. For simplicity, we refer to the states of both as $\mathbf{h}^{\text{token}}$, but it is clear from the context which one is used.

defined by $\mathbf{O}$.

Formally, the probability of concept identification step can be decomposed into probability of generating m concepts nodes and n terminal nodes (one for each token):

$$P_\theta(\mathbf{v}|\mathbf{x},\mathbf{O}) = \prod_{i=0}^{m-1} P_\theta(\mathbf{v}_i|\mathbf{h}_i^{\text{node}}) \prod_{k=0}^{n-1} P_\theta(\emptyset|\mathbf{h}_k^{\text{tail}}) \tag{5.2}$$

Representation $\mathbf{h}_i^{\text{node}}$ is computed as the weighted sum of the LSTM states of preceding nodes as defined by $\mathbf{O}$ (recall that $\mathbf{O} = \mathbf{A} \circ \mathbf{S}$):

$$\mathbf{h}_i^{\text{node}} := \sum_{j=0}^{m-1} \mathbf{S}_{ji} \text{LSTM}(\mathbf{h}_j^{\text{node}}, \mathbf{v}_j) + \sum_{k=0}^{n-1} \mathbf{A}_{ki} \mathbf{h}_k^{\text{token}}. \tag{5.3}$$

Note that the preceding node can be either a concept node (then the output of the LSTM, consuming the preceding node, is used) or a word (then we use its contextualized encoding). Note that this expression is 'recursive' – each node's representation $\mathbf{h}_i^{\text{node}}$ is computed based on representations of all the nodes $\mathbf{h}_j^{\text{node}}$; $i, j \in 1, \ldots m-1$. Iterating the assignment defined by Expression (5.3) for a valid discrete generation order (i.e., a DAG, like the one given in Figure 5.2a), will converge to a stationary point. Crucially, the stationary point will be equal to the result of applying the autoregressive model (as in test time, see Figure 5.3). It will be reached after T steps, where T is the number of nodes in the largest subgraph.[5] This 'message passing' process is fully differentiable and, importantly, well-defined for a relaxed (i.e., continuous) generation order. The equivalence between the train-time message passing and the test-time autoregressive computation with discrete $\mathbf{O}$ should ensure that the gap between training and testing is minimal if a near-discrete sample is provided during training.

The representations $\mathbf{h}_k^{\text{tail}}$, needed for the terms $P_\theta(\emptyset|\mathbf{h}_k^{\text{tail}})$ in Equation (5.2), are computed as:

$$\mathbf{h}_k^{\text{tail}} = \sum_{j=0}^{m-1} \mathbf{B}_{jk} \text{LSTM}(\mathbf{h}_j^{\text{node}}, \mathbf{v}_j) + (1 - \sum_{j=0}^{m-1} \mathbf{B}_{jk}) \mathbf{h}_k^{\text{token}}, \tag{5.4}$$

where $\mathbf{B}_{jk} = 1$ denotes that the concept node j is the last concept node before generating $\emptyset$ for the token k, else $\mathbf{B}_{jk} = 0$. (e.g. in Figure 5.2a, we have $\mathbf{B}_{11} = \mathbf{B}_{42} = 1$, and others are 0).

5.2.1.1 Last Node Indicator B

We obtain $\mathbf{B}$ by having:

$$\mathbf{B} = \mathbf{A}[\mathbf{S}_{:,:m} + \text{Diag}(\mathbf{S}_{:,m})]^T; \tag{5.5}$$

[5] We use $T = 4$, as we do not expect subgraphs with more than 4 nodes.

where $\mathbf{S}_{:,:m}$ takes the submatrix of $\mathbf{S}$, excluding the last column, and $\text{Diag}(\mathbf{S}_{:,m}$ is the diagonal matrix whose diagonal entries are the last column of $\mathbf{S}$. Intuitively, $[\mathbf{S}_{:,:m} + \text{Diag}(\mathbf{S}_{:,m})]$ can be thought as a Markov transition matrix that pass down the alignment along the generation order, but keep the alignment mass if the node will generate $\emptyset$. We truncate the transition at $T = 4$, as we do not expect a subgraph containing more than 4 nodes.

Again, in the discrete case, the result will be exactly equivalent to what is obtained by running the corresponding autoregressive model, but the computation is well-defined and differentiable in the continuous case. Intuitively, if we are to think of relaxed $\mathbf{O}$ as transition probabilities in a Markov process, then B_{jk} can be thought of as the probability of having node j as the last concept node in the chain when starting from token k.

5.2.1.2 Individual Concept Identification

Now, we specify $P_\theta(\mathbf{v}_i|\mathbf{h}_i^{\text{node}})$ and $P_\theta(\mathbf{v}_m|\mathbf{h}_i^{\text{tail}})$ with copy mechanism. Importantly, the copying mechanism needs to be relaxed during training. Formally, we have a small set of candidate nodes $\mathcal{V}(\mathbf{x}_i)$ for each token $\mathbf{x}_i$, and a shared set of candidate nodes $\mathcal{V}^{\text{share}}$, which contains $\mathbf{v}_{\text{copy}}$. This, however, depends on the aligned token. During training, as the alignment is not given, we consider the union of candidate nodes from all possible tokens $\mathcal{V}(\mathbf{v}_i) = \cup_{j:\mathbf{v}_i \in \mathcal{V}(\mathbf{x}_j)} \mathcal{V}(\mathbf{x}_j)$. Abusing the notation slightly, we denote the embedding of node i by $\mathbf{v}_i$. In all, at training time, for node $\mathbf{v}_i$, we have

$$\mathbf{h}_i^c = \text{NN}^{\text{node}}(\mathbf{h}_i^{\text{node}}; \theta) \tag{5.6}$$

$$\begin{aligned} P_\theta(\mathbf{v}_i|\mathbf{h}_i^{\text{node}}) = & \frac{[[\mathbf{v}_i \in \mathcal{V}^{\text{share}}]] \exp(\langle \mathbf{v}_i, \mathbf{h}_i^c \rangle)}{\sum_{\mathbf{v} \in \mathbf{v}} \exp(\langle \mathbf{v}, \mathbf{h}_i^c \rangle)} \\ & + \frac{[[\mathbf{v}_i \in \mathcal{V}(\mathbf{v}_i)]] \exp(\langle \mathbf{v}_{\text{copy}}, \mathbf{h}_i^c \rangle)}{\sum_{\mathbf{v} \in \mathbf{v}} \exp(\langle \mathbf{v}, \mathbf{h}_i^c \rangle)} \times \frac{\exp(\text{S}(\mathbf{v}_i, \mathbf{h}_i^c))}{\sum_{\mathbf{v} \in \mathcal{V}(\mathbf{v}_i)} \exp(\text{S}(\mathbf{v}, \mathbf{h}_i^c)} \end{aligned} \tag{5.7}$$

where NN is a standard one-layer feedforward neural network, and $[[\ldots]]$ denotes the indicator function. $\text{S}(\mathbf{v}, \mathbf{h}_i^c)$ scores candidate nodes given the hidden state. To utilize pre-trained word embedding (Pennington et al., 2014), the representation of $\mathbf{v}$ is decomposed into primitive category embedding $\mathcal{C}(\mathbf{v})$[6] and surface lemma embedding. The score function is then a biaffine scoring based on embeddings and hidden

[6] AMR nodes have primitive category, including string, number, frame, concept and special node (e.g. polarity).

states($\mathcal{L}(\mathbf{v})$ $\mathrm{S}(\mathbf{v}, \mathbf{h}_i^c) = \mathrm{Biaffine}(\mathcal{C}(\mathbf{v}) \circ \mathcal{L}(\mathbf{v}), \mathbf{h}_i^c; \theta)$. For the terminal nodes, we have:

$$\mathbf{h}_i^t = \mathrm{NN}^{\mathrm{node}}(\mathbf{h}_i^{\mathrm{tail}}; \theta) \tag{5.8}$$

$$P_\theta(\mathbf{v}_m | \mathbf{h}_i^{\mathrm{tail}}) = \frac{\exp(\langle \mathbf{v}_m, \mathbf{h}_i^t \rangle)}{\sum_{\mathbf{v} \in \mathbf{v}} \exp(\langle \mathbf{v}, \mathbf{h}_i^t \rangle)} \tag{5.9}$$

5.2.2 Relation Identification

Following last chapter, we use an arc-factored model for relation identification:

$$P_\theta(\mathbf{E} | \mathbf{x}, \mathbf{O}, \mathbf{v}) = \prod_{i,j=1}^{m} P_\theta(\mathbf{E}_{ij} | \mathbf{h}_i^{\mathrm{edge}}, \mathbf{h}_j^{\mathrm{edge}}) \tag{5.10}$$

where $P_\theta(\mathbf{E}_{ij} | \mathbf{h}_i^{\mathrm{edge}}, \mathbf{h}_j^{\mathrm{edge}})$ is the softmax of the biaffine function of node representations $\mathbf{h}_i^{\mathrm{edge}}$ and $\mathbf{h}_j^{\mathrm{edge}}$. The node representations are defined as

$$\mathbf{h}_i^{\mathrm{edge}} = \mathrm{NN}^{\mathrm{edge}}(\mathbf{h}_i^{\mathrm{node}} \circ \sum_{k=0}^{n-1} \mathbf{A}_{ki}^{\infty} \mathbf{h}_k^{\mathrm{token}}), \tag{5.11}$$

where $\circ$ denotes concatenation, $\mathbf{h}_i^{\mathrm{node}}$ is defined in section 5.2.1, and $\mathbf{A}_{ki}^{\infty}$ indicates whether i is in a subgraph aligned to token k or not. Not that this is different from $\mathbf{A}_{ki}$ which encodes that the node i is the first node in the subgraph (e.g., in Figure 5.2a, $\mathbf{A}_{11} = 0$ but $\mathbf{A}_{11}^{\infty} = 1$). In the continuous case, as used during training, $\mathbf{A}_{ki}^{\infty}$ can be thought of as the alignment probability that can be computed from $\mathbf{O}$, and it is differentiable with respect to $\mathbf{O}$.

5.2.2.1 Alignment $\mathbf{A}^{\infty}$

To obtain $\mathbf{A}^{\infty}$, we observe $\mathbf{A}^{\infty}$ should obey the following self-consistency equation:

$$\mathbf{A}^{\infty} = \mathbf{A}^{\infty} \mathbf{S}_{:,:m} + \mathbf{A} \tag{5.12}$$

This means that node j is generated from token k iff node i is is generated from token k and node i generates node j or node j is directly generated from token k. This $\mathbf{A}^{\infty}$ can be computed by initializing $\mathbf{A}^{(0)} = \mathbf{A}$, and repeating Equation 5.12 as assignment $\mathbf{A}^{(t+1)} = \mathbf{A}^{(t)} \mathbf{S}_{:,:m} + \mathbf{A}$ for $T = 4$ times. Intuitively, the $\mathbf{A}^{(t)}$ alignment is passed down along the generation order, while keep getting alignment mass from the first node alignment. As a result, all nodes get assigned alignment. As an alternative motivation, the above algorithmic assignment works as a truncated power series expansion of self-consistency equation solution $\mathbf{A}^{\infty} = [I - \mathbf{S}_{:,:m}]^{-1} \mathbf{A}$.

5.3 Estimating Latent Generation Order

We show how to estimate the latent generation order jointly with the parser.

5.3.1 Variational Inference

In Equation (5.1), the marginalization over $\mathbf{O}$ is intractable due to the use of neural parameterization in $P_\theta(\mathbf{v}|\mathbf{x},\mathbf{O})$ and $P_\theta(\mathbf{E}|\mathbf{x},\mathbf{O},\mathbf{v})$. Instead, we resort to the deep VI framework. We optimizes a lower bound on the marginal likelihood objective:

$$\begin{aligned}&\log \sum_{\mathbf{O}} P_\theta(\mathbf{O}) P_\theta(\mathbf{v}|\mathbf{x},\mathbf{O}) P_\theta(\mathbf{E}|\mathbf{x},\mathbf{O},\mathbf{v}) \\ \geq &\mathop{\mathrm{E}}_{\mathbf{O}\sim Q_\phi(\mathbf{O}|\mathbf{G},\mathbf{x})} \log P_\theta(\mathbf{v}|\mathbf{x},\mathbf{O}) P_\theta(\mathbf{E}|\mathbf{x},\mathbf{O},\mathbf{v}) - \mathrm{KL}(Q_\phi(\mathbf{O}|\mathbf{G},\mathbf{x})||P_\theta(\mathbf{O}))\ , \end{aligned} \tag{5.13}$$

where KL is the Kullback-Leibler divergence, and $Q_\phi(\mathbf{O}|\mathbf{G},\mathbf{x})$ is the variational distribution parameterized with a neural network (the encoder, aka, the inference network). The lower bound is maximized with respect to both the original parameters θ and the variational parameters ϕ. The distribution $Q_\phi(\mathbf{O}|\mathbf{G},\mathbf{x})$ can be thought of as an approximation to the intractable posterior distribution $P_\theta(\mathbf{O}|\mathbf{G},\mathbf{x})$.

5.3.2 Stochastic Softmax

In order to estimate the gradient with respect to the encoder parameters ϕ, we use the stochastic softmax that we introduced in Section 3.2.1.

Instead of sampling $\mathbf{O}$ directly, we sample a random variable ε from a fixed distribution $\mathcal{G}$. Then, we apply a deterministic parameterized function $\mathbf{O}_\phi$ to get $\mathbf{O} = \mathbf{O}_\phi(\varepsilon,\mathbf{G},\mathbf{x})$.[7] More concretely, we independently compute logits $\mathbf{W} \in \mathbb{R}^{(n+m)\times(m+1)}$ for the all the potential edges in the generation order, and perturb them:

$$\mathbf{W} = \mathrm{F}_\phi(\mathbf{G},\mathbf{x}) \tag{5.14}$$

$$\widetilde{\mathbf{W}} = \mathbf{W} + \varepsilon, \quad \text{where} \quad \varepsilon_{ij} \sim \mathcal{G}(0,1) \tag{5.15}$$

where F_ϕ is a neural module that we will define in Section 5.3.2.2, $\mathcal{G}(0,1)$ is the standard Gumbel distribution, and $\varepsilon \in \mathbb{R}^{(n+m)\times(m+1)}$. Then, those perturbed logits $\widetilde{\mathbf{W}}$ are

[7] As we discussed in Section 3.2.3, a score-function gradient estimator can not be applied for sampling permutation. For generation order, in the case of $n = 1$, the constraints reduced to a permutation constraint. Therefore, neither a score-function gradient estimator can be applied here.

fed into a constrained convex optimization problem:

$$\mathbf{O}(\widetilde{\mathbf{W}},\tau) := \arg\max_{\mathbf{O}\geq 0} \langle \widetilde{\mathbf{W}}, \mathbf{O}\rangle - \tau\langle \mathbf{O}, \log \mathbf{O}\rangle$$
$$\text{s.t.} \forall j < m \sum_{i=0}^{n+m-1} \mathbf{O}_{ij} = 1; \forall i \sum_{j=0}^{m} \mathbf{O}_{ij} = 1 \tag{5.16}$$

this is the linear programming (LP) relaxation of constraints discussed in Section 5.1.2, where we allowed continuous-valued $\mathbf{O}$. Importantly, this LP relaxation is 'tight', and ensures that $\mathbf{O}(\widetilde{\mathbf{W}},0)$ is a valid generation order.[8] This allows straight-through (ST) estimator (see Section 3.2.2.1) to forward pass a valid generation order. We will use ST estimator in our model.[9] There is an extra acyclicity constraint, which is enforced by masking on $\mathbf{W}$, see Section 5.3.2.2.

Now, as we will show in the next section, the solution to this optimization $\mathbf{O}(\widetilde{\mathbf{W}},\tau)$ can be obtained with a differentiable computation, thus, we write:

$$\mathbf{O}_\phi(\varepsilon, \mathbf{G}, \mathbf{x}) = \mathbf{O}(\widetilde{\mathbf{W}},\tau) \tag{5.17}$$

The entropy regularizor, weighted by $\tau > 0$ ('the temperature'), ensures differentiability with respect to $\mathbf{W}$ and, thus, with respect to ϕ.

To handle the KL term in Equation (5.13), we define the prior probability $P_\theta(\mathbf{O})$ implicitly by having $\mathbf{W} = 0$ in the stochastic softmax framework. Even then, $\mathrm{KL}(Q_\phi(\mathbf{O}|\mathbf{G},\mathbf{x})||P_\theta(\mathbf{O}))$ cannot be easily computed. Following Mena et al. (2018), we upper bound it by replacing it with $\mathrm{KL}(\mathcal{G}(\mathbf{W},1)||\mathcal{G}(0,1))$, which is available in closed form.

5.3.2.1 Bregman's Method

To optimize objective (5.16) we iterate over the following steps of optimization:

$$\mathbf{O}^{(0)} = \exp\frac{\widetilde{\mathbf{W}}}{\tau} \tag{5.18}$$
$$\forall j < m,\ \mathbf{O}_{:,j}^{(t+\frac{1}{2})} = \mathcal{T}(\mathbf{O}_{:,j}^{(t)}) \tag{5.19}$$
$$\mathbf{O}_{:,m}^{(t+\frac{1}{2})} = \mathbf{O}_{:,m}^{(t)} \tag{5.20}$$
$$\forall i,\ \mathbf{O}_{i,:}^{(t+1)} = \mathcal{T}(\mathbf{O}_{i,:}^{(t+\frac{1}{2})}) \tag{5.21}$$

[8] We provide the proof in Appendix C.4.

[9] A quick reminder, in ST estimator, we use $\mathbf{O}(\widetilde{\mathbf{W}},0)$ in the forward pass, and set $\nabla_{\widetilde{\mathbf{W}}}\mathbf{O}(\widetilde{\mathbf{W}},0) := \nabla_{\widetilde{\mathbf{W}}}\mathbf{O}(\widetilde{\mathbf{W}},\tau)$. We solve this LP by setting $\tau = 1e-5$ in our Bregman's method. Paulus et al. (2020) established that the low-temperature solution of the convex optimization problem to the LP solution in general cases.

where $\{i,:\}$ index ith row, $\{:,j\}$ index jth column and $\mathcal{T} = \frac{\mathbf{x}}{\sum_i \mathbf{x}_i}$ normalize the vectors. Intuitively, the alignment scores are initially computed from the logits $\widetilde{\mathbf{W}}$, without taking constraints into account, and then alternating optimization is used to 'fit' subsets of the constraints.[10]

Proposition 1. $\lim_{t\to\infty} \mathbf{O}^{(t)} = \mathbf{O}(\widetilde{\mathbf{W}}, \tau)$ *where* $\mathbf{O}(\widetilde{\mathbf{W}}, \tau)$ *is defined in Equation (5.16).*

See Appendix C.3 for a proof based on the Bregman method (Bregman, 1967). In practice, we take $T = 50$, and have $\mathbf{O}_\phi(\varepsilon, \mathbf{G}, \mathbf{x}) = \mathbf{O}^{(T)}$. Importantly, this algorithm is highly parallelizable and amendable to batch implementation on GPU. We compute the gradient with unrolled optimization.

5.3.2.2 Neural Parameterization

We introduce the neural modules used for estimating logits $\mathbf{W} = \mathrm{F}_\phi(\mathbf{G}, \mathbf{x})$ and also the masking mechanism that both ensures acyclicity and enables the use of the copy mechanism. We have $\mathbf{W} = \mathbf{W}^{\text{raw}} + \mathbf{W}^{\text{mask}}$. First, we define the unmasked logits, $\mathbf{W}^{\text{raw}} = \mathbf{A}^{\text{raw}} \circ \mathbf{S}^{\text{raw}}$:

$$\mathbf{h}^{\text{g}} = \text{RelGCN}(\mathbf{G}; \theta) \in \mathbb{R}^{m \times d}$$
$$\mathbf{A}^{\text{raw}} = \text{BiAffine}^{\text{align}}(\mathbf{h}^{\text{token}}, \mathbf{h}^{\text{g}} \circ \mathbf{h}^{\text{end}}; \phi)$$
$$\mathbf{S}^{\text{raw}} = \text{BiAffine}^{\text{segment}}(\mathbf{h}^{\text{g}}, \mathbf{h}^{\text{g}} \circ \mathbf{h}^{\text{end}}; \phi)$$

where RelGCN is a relational graph convolutional network (Schlichtkrull et al., 2018) that takes an AMR graph $\mathbf{G}$ and produces embeddings of its nodes informed by their neighbourhood in $\mathbf{G}$. $\mathbf{h}^{\text{end}} \in \mathbb{R}^{1\times d}$ is the trainable embedding of the terminal node, and $\mathbf{h}^{\text{token}} \in \mathbb{R}^{n\times d}$ is the BiLSTM encoding of a sentence from Section 5.2.1.[11]

The masking also consists of two parts, the alignment mask and the segmentation mask, $\mathbf{W}^{\text{mask}} = \mathbf{A}^{\text{mask}} \circ \mathbf{S}^{\text{mask}}$. If a node is copy-able from at least one token, the alignment mask prohibits alignments from other tokens by setting the corresponding components $\mathbf{A}^{\text{mask}}_{ij}$ to $-\infty$.

Acyclicity is ensured by setting $\mathbf{S}^{\text{mask}}$ so that generation order with circles will get negative infinity in Equation (5.16). While there may be better and more general ways to encode acyclicity (Martins et al., 2009), we perform a depth-first search (DFS) from the root node[12] and permit an edge from node i and j only if i precedes j (not

[10] Our formulation is very similar to the Gumbel-Sinkhorn (Mena et al., 2018), which models bijective alignment. The difference made our formulation useful for modeling non-square injective alignment.

[11] Technically, this is the average of the two token encoders mentioned in footnote 4.

[12] Arbitrarily, we use lexicographic ordering of edge labels in DFS.

necessarily immediately) in the traversal. In other words, $\mathbf{S}^{\text{mask}}_{ij}$ is set to $-\infty$ for edges (i, j) violating this constraint. The rest of components in $\mathbf{S}^{\text{mask}}$ are set to 0. Note that this masking approach does not require changes in the optimization method.

5.4 Decoding

While we relied on the latent variable machinery to train the parser, we do not use it at test time. In fact, the encoder $Q_\phi(\mathbf{O}|\mathbf{G}, \mathbf{x})$ is discarded after training. At test time, the first step is to predict sets of concept nodes for every token using the concept identification model $P_\theta(\mathbf{v}|\mathbf{x}, \mathbf{O})$ (as shown in Figure 5.3). Note that the token-specific autoregressive models can be run in parallel across tokens. The second step is predicting relations between all the nodes, relying on the relation induction model $P_\theta(\mathbf{E}|\mathbf{x}, \mathbf{O}, \mathbf{v})$. In addition, we need another root identifier that chooses the root and a decoding algorithm to obtain a connected graph. We specify the root identification as:

$$P_\theta(i|\mathbf{x}, \mathbf{O}, \mathbf{v}) = \frac{\exp(\langle \mathbf{h}^{\text{root}}, \mathbf{h}^e_i \rangle)}{\sum_{j=0}^{m-1} \exp(\langle \mathbf{h}^{\text{root}}, \mathbf{h}^e_j \rangle)} \tag{5.22}$$

where $\mathbf{h}^{\text{root}}$ is a trainable vector. Inspired by Zhang et al. (2019a), who utilizes the fact that AMR graph is very closely related to the dependency tree, we first decode the AMR graph as a maximum spanning tree with the log probability of most likely arc-label as edge weights. The reentrancy edges are added afterward, if their probability is larger than 0.5. We add at most 5 reentrancy edges, based on the empirical founding of Szubert et al. (2020) that shows most AMR graphs have less than 5 reentrancy edges.

5.5 Fixed Segmentations

We'd like to compare our generation-order induction framework to some fixed segmentations, i.e., producing the segmentation on a preprocessing step. We fix the segmentation, but still induce the alignment through our latent-generation-order framework. This is achieved by having the fixed $\mathbf{S}$, then set $\mathbf{S}^{\text{mask}}_{ij} = \infty(\mathbf{S}_{ij} - 0.5)$ if $j \neq m$. In particular, we have a hand-crafted rule-based segmentation and a greedy segmentation as our fixed segmentations.

Rule-based Segmentation We introduce a hand-crafted rule-based segmentation, which relies on rules designed to handle specific AMR constructions. In particular, we consider a hand-crafted segmentation system that we derive from the re-categorization

system[13] from the last chapter. A recategorized concept represents a subgraph. e.g., e.g. 'thing(opinion)' represents a subgraph containing 'thing' and 'opine-01'. Still, the local autoregressive generation of nodes requires an ordering of nodes within the subgraph. This is chosen according to the DFS traversal that we performed for ensuring acyclicity. e.g., 'thing' points to 'opine-01' in '(t / thing :ARG1-of (o / opine-01 :ARG0 (b / boy)))'. Arguably, this can be thought of as an upper bound for how well an induction method can do.

Greedy Segmentation We provide a greedy strategy for segmentation that serves as a deterministic baseline. This greedy segmentation can be used in the same way as the rule-based segmentation by setting $\mathbf{S}^{\text{mask}}$.

Intuitively, many nodes are aligned to tokens with the copy mechanism. We could force the unaligned nodes to join their neighbors. This is very similar to the forced alignment of unaligned nodes used in the transition parser of Naseem et al. (2019). We traversal the AMR graph the same way as we do when we produce the masking (Section 5.3.2.2). During the traversal, we greedily combine subgraphs until one of the constraints is violated: (1) the combined subgraph will have more than 4 nodes; (2) the combined subgraph will have more than 2 copy-able nodes. We present the algorithm as pseudo code in Appendix C.1. Importantly, this greedy method does not require any expert knowledge about AMR, so this should serve as a baseline.

5.6 Pre-and-Post processing

We use CoreNLP (Manning et al., 2014) for tokenization and lemmatization. The copy-able dictionary is built with the rules based on string matching between lemmas and concept node string as in the previous chapter with minor modifications.[14]

For post-processing, wiki tags are added after the named entity being produced in the graph via a look-up table built from the training set or provided by CoreNLP. We also collapse nodes that represent the same pronouns as a heuristics for co-reference resolution.

[13] More specifically, its re-implementation by Zhang et al. (2019a).
[14] The modification is a consequence of our reliance on code from Zhang et al. (2019a).

Metric	Concept	SRL	Smatch
Lyu and Titov (2018)	85.9	69.8	74.4
Zhang et al. (2019b)	86	71	77.0
Naseem et al. (2019)	86	72	75.5
Lindemann et al. (2020)	-	-	76.8
Cai and Lam (2020)	88.1	74.2	**80.2**
Xu et al. (2020)	87.4	**78.9**	**80.2**
Lee et al. (2020)	88.1	78.2	**80.2**
greedy	87.5 $\pm$ 0.1	71.3 $\pm$0.1	75.2 $\pm$ 0.1
rule	**88.7** $\pm$ **0.2**	73.6$\pm$ 0.2	76.8 $\pm$ 0.4
full	88.3 $\pm$ 0.3	73.0 $\pm$ 0.2	76.1 $\pm$ 0.2

Table 5.1: Scores with standard deviation on the AMR 2.0 test set. Integer number denotes 2 significant digits. Results are averaged over 4 runs.

5.7 Experiments and Discussions

We experiment on LDC2016E25 (AMR2.0) and LDC2020T02 (AMR3.0). The evaluation is based on Smatch (Cai and Knight, 2013), and the evaluation tool of Damonte et al. (2017). We compare our generation-order induction framework to fixed segmentations and other parsers. We vary the segmentation methods while keeping the rest of the model identical to our full model (i.e., the same autoregressive model and the learned alignment). We provide ablation studies for our induction framework. Extra details, including hyper-parameters, are in Appendix C.

5.7.1 Results and Discussion

We compare our models with other recent AMR parsers in Table 5.1. Overall, our model performs competitively, but lags behind those very recent parsers from Cai and Lam (2020); Lee et al. (2020); Xu et al. (2020).[15] In addition, because we are using RoBERTA (Liu et al., 2019c), our model is better than that of Lyu and Titov (2018) (the last chapter). Importantly, both our VI model and the rule-based segmentation achieve high concept identification scores (Damonte et al., 2017). This suggests that the bottleneck of our graph-based parser is on the relation identification stage, which

[15]Results from Lee et al. (2020) replace Roberta-large with Roberta-base in Fernandez Astudillo et al. (2020). With semi-supervised learning, Lee et al. (2020) achieved 81.3 Smatch score.

Metric	Concept	SRL	Smatch
greedy	87.0	71.5	74.8
rule	**88.0**	72.6	**75.8**
full	87.8	**72.9**	75.6

Table 5.2: Scores on the AMR 3.0 test set, averaged over 2 runs.

Metric	Concept	SRL	Smatch
all prior	81.7	62.6	61.9
alignment prior	86.0	69.1	70.5
segmentation prior	87.6	71.1	74.4
full	**88.3**	**73.0**	**76.1**

Table 5.3: Scores with different latent segmentation on the AMR 2.0 test set, averaged over 2 runs

we largely borrowed from Lyu and Titov (2018) and is rather basic. For example, independent scoring of the edges may be too restrictive. Also, unlike Fernandez Astudillo et al. (2020); Lee et al. (2020), we use potentially weaker BiLSTM encoders instead of the Transformer.

Results on AMR 3.0 are summarized in Table 5.2.[16] Our VI segmentation beats the greedy baseline and approaches the rule-based system. The performance gap between the rule-based system and VI is smaller on AMR 3.0, presumably because the rules were developed for AMR 2.0.

To reconfirm that it is important to learn the segmentation and alignment, rather than to sample it randomly, we perform further ablations. In our parameterization, discussed in Section 5.3.2.2, it is possible to set $\mathbf{A}^{\text{raw}} = 0$ and $\mathbf{S}^{\text{raw}} = 0$, which corresponds to sampling from the prior in training (i.e., quasi-uniformly while respecting the constraints defined by masking) rather than learning this importance sampling distribution with the encoder. There are 4 potential options: (1) 'only prior', $\mathbf{A}^{\text{raw}} = 0$ and $\mathbf{S}^{\text{raw}} = 0$; (2) 'alignment prior', $\mathbf{A}^{\text{raw}} = 0$ (while $\mathbf{S}^{\text{raw}}$ is learned) (3) 'segmentation prior', $\mathbf{S}^{\text{raw}} = 0$ (while $\mathbf{A}^{\text{raw}}$ is learned); (4) our full model where both are learned, i.e. they constitute an output of a trained encoder. The results are summarized in Table 5.3. As expected, the full model performs the best, demonstrating that it is important to learn both alignments and segmentation. Interestingly, both 'segmentation

[16] There is no published result on AMR 3.0 yet.

Metric	Concept	SRL	Smatch
no free bits	83.5	66.3	66.1
soft	84.9	68.1	70.3
rounding	87.7	71.8	74.5
straight-through	**88.3**	**73.0**	**76.1**

Table 5.4: Scores with different latent segmentation on the AMR 2.0 test set. Scores are averaged over 2 runs

prior' and 'alignment prior' obtain reasonable performance, but the 'all prior' model fails badly. One possible explanation is that given one of them being learned, the variations remaining in the other can be controlled so that the overall sampling variance will be small.

Our full model uses the ST gradient estimator and the Free Bits trick[17] with $\lambda = 10$ (Kingma et al., 2016). We perform analysis of different variations of the stochastic softmax: (1) the soft stochastic softmax is the original one with the entropic regularizer (see Section 5.3.2); (2) the rounded stochastic softmax, which selects the highest scored next node from each token and concept nodes based on the soft stochastic softmax;[18] (3) our full model with the ST estimator. All those models use Free Bits ($\lambda = 10$), while for 'no free bits $\lambda = 0$. As we can see in Table 5.4, there is a substantial gap between using structured ST and the two other versions. This illustrates the need for exposing the parsing model to discrete structures in training. Also, the Free Bits trick appears crucial as it prevents the (partial) posterior collapse in our model. We inspected the logits after training and observed that, without free-bits, the learned $\mathbf{W}$ are in the range of $[-0.01, +0.01]$. Indeed, they are quite small.

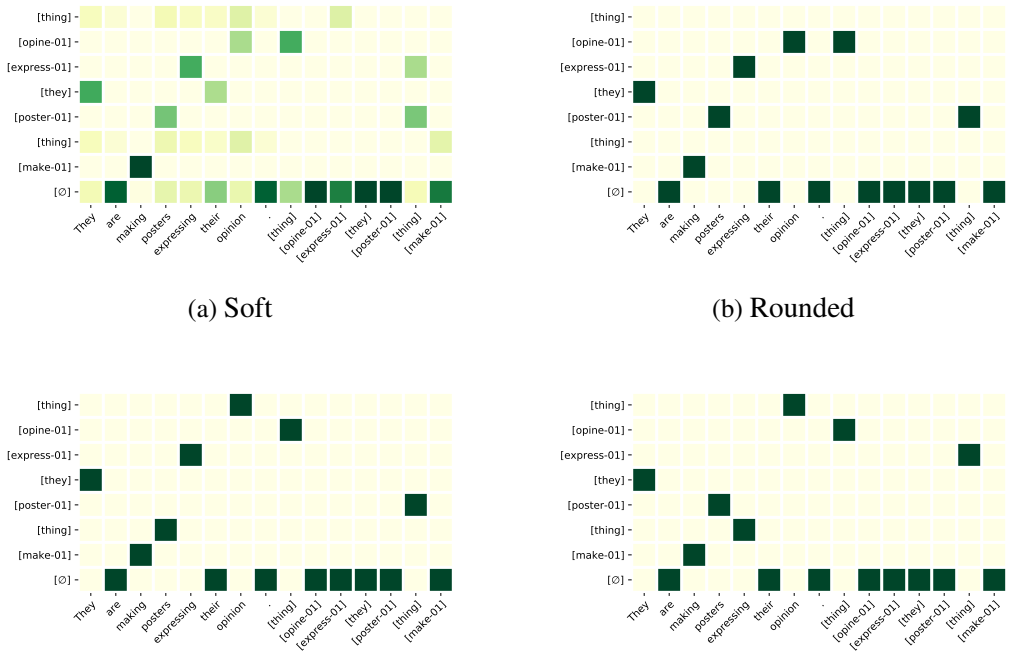
5.7.2 Visualization

In Figure 5.4, we present one example of the induced generation order for our three variations of stochastic softmax, and one with rule-based segmentation.[19] The nodes are represented in []. Their gold AMR is:

[17] The Free Bits trick is used to prevent 'the posterior collapse' (Kingma et al., 2016). Concretely, we use $\max(\lambda, \mathrm{KL}(\mathcal{G}(\mathbf{W}, 1) || \mathcal{G}(0, 1)))$ for the KL divergence regularizer to avoid penalizing small logits.

[18] Such rounding does not provide any guarantee of a sample being a valid generation order, but serves as a baseline. In general, a threshold function (at 0.5) can be applied if the constraints have no structure.

[19] Incidentally, the greedy segmentation produces the same segmentation as rule-based in this example.



(a) Soft

(b) Rounded

(c) Hard

(d) Rule

Figure 5.4: Examples of Stochastic Softmax Latent Generation Order. 5.4a is from soft stochastic softmax, 5.4b is from rounded stochastic softmax, 5.4c is from hard (strighat-through) stochastic softmax and 5.4d is from rule-based segmentation.

```
(m / make-01
    :ARG0 (t / they)
    :ARG1 (t2 / thing
        :ARG2-of (p / poster-01)
        :ARG0-of (e / express-01
            :ARG1 (t3 / thing
                :ARG1-of (o / opine-01
                    :ARG0 t)))))
```

As we can see, the standard stochastic softmax indeed produces soft latent structure that might result in a large training/testing gap. Furthermore, the rounding strategy does not satisfy the constraint that every concept node can only be generated from one token or another concept node (e.g.., [poster-01] is generated twice, and [thing] is never generated.). Meanwhile, the straight-through stochastic softmax produce a valid generation order. It is worth to note that our learned generation order differs from the rule-based one. When producing the rule-based segmentation, '(t2 / thing :ARG0-of (e / express-01)' took precedence over '(t2 / thing :ARG2-of (p / poster-01)' due to the order over traversal edges. The learned model, however, figured out that the 'poster' is

the thing.

5.8 Related Work

Most strong parsers use graph segmentation. They typically rely on hand-crafted rules, with rule templates developed by studying training set statistics and ensuring the necessary level of coverage (Flanigan et al., 2014; Werling et al., 2015; Foland and Martin, 2017).

Alternatively, grammar-based parsing starts with lexical entries that correspond to subgraphs. The grammar induction is similar to the induction of alignment and graph segmentation.[20] Artzi et al. (2015) optimized the parser parameters and induced the grammar jointly by treating derivations in Combinatory Categorial Grammar (CCG) as latent variables (Zettlemoyer and Collins, 2005; Kwiatkowksi et al., 2010; Kwiatkowski et al., 2011). Groschwitz et al. (2017, 2018); Lindemann et al. (2020) use a decomposition automaton (Koller and Kuhlmann, 2011) to segment AMR graphs. Peng et al. (2015) apply Markov Chain Monte Carlo to sample synchronous hyperedge replacement grammar rules for derivations of AMR graph (similar to CCG, lexical rules are graph fragments). Later, Gildea et al. (2019) extracted hyperedge replacement grammar rules from tree decompositions of the AMR graphs. All those approaches rely on existing grammar formalisms. We will discuss further the end-to-end differentiable training of grammar-based parsing in Section 7.2.

Importantly, the state-of-the-art neural autoregressive model still benefits from an explicit graph segmentation (Graph Recategorization) (Cai and Lam, 2020). In their transition parsers, Fernandez Astudillo et al. (2020); Lee et al. (2020) use the term 'graph recategorization' to refer to a system that involves an external NER tagger, but they also have a system to 'pack' and 'unpack' nodes.

More generally, outside of AMR parsing, differentiable relaxations of latent structure representations have received attention in NLP (Kim et al., 2017; Liu and Lapata, 2018), including previous applications of the perturb-and-MAP framework (Corro and Titov, 2019). From the general goal perspective – inducing a segmentation of a linguistic structure – our work is related to tree-substitution grammar induction (Sima'an et al., 1994; Cohn et al., 2010), the DOP paradigm (Bod et al., 2003), 'unsupervised semantic parsing' (Poon and Domingos, 2009; Titov and Klementiev, 2011), sequence segmentation (Wang et al., 2017), though the methods used in that previous work are

[20] Indeed, we can extract the alignment and graph segmentation from an induced derivation.

very different from ours.

5.9 Summary

To get rid of the hand-crafted segmentation systems used in previous AMR parsers, we cast the alignment and segmentation as generation-order induction. We propose to treat this generation order as a latent variable in a deep variational inference framework. With the stochastic softmax, our model is end-to-end differentiable, Empirically, our method outperforms a technique relying on a simple segmentation heuristic and approaches the performance of a method using rules designed to handle specific AMR constructions. In ablation studies, we found straight-through estimator and free-bits trick are essential for the success of our method.

Importantly, while the latent variable modeling machinery is used in training, the parser is very simple at test time. It tags the input words with AMR concept nodes with autoregressive models and then predicts relations between the nodes independent from each other.

As we achieved high performance on concept identification, one way to further improve the graph-based parser is to replace the fully factorized relation identification component. In the next chapter, we go beyond such factorized prediction.

Chapter 6

Semantic Role Labeling with Iterative Structure Refinement

The graph-based parsers that we built for AMR parsing in the earlier chapters have one fundamental limitation in terms of their expressiveness. The probabilistic models are factorized, where the prediction of edges are independent from each other. [1]

In this chapter, we propose *iterative structure refinement* to capture the interdependencies within the output structure. We chose (dependency-based) semantic role labeling (SRL) 2.1 to test our method. We switch from AMR parsing for three reasons: first, as shown from the previous chapter, our AMR parser performs worse than other parsers in the SRL metric (i.e., less accurate in predicting relations). We would like to focus on this phenomenon;[2] second, SRL is a simple task and is hence more suitable as a controlled experiment setting for our new method; third, the SRL benchmarks are more extensive as they include out-of-domain settings and many languages.

Prior to the deep learning models, earlier works on semantic role labeling assign scores over the joint configuration of semantic roles of different arguments (Toutanova et al., 2008; Watanabe et al., 2010). With the deep learning models, end-to-end parameter estimation also becomes more involved and fragile (Belanger and McCallum, 2016; Belanger et al., 2017). Second-order scoring (e.g., incorporating information about sibling or grandparents) has been first investigating in the context of syntactic dependency parsing (McDonald and Pereira, 2006). Very recently, with unrolled iterative approximate inference, end-to-end neural models adopt similar second-order scoring for semantic parsing tasks (Wang et al., 2019; Li et al., 2020).

[1] Of course, the model is not completely factorized, as one cannot have edges without nodes, and the node generation in Chapter 5 is locally autoregressive.

[2] We discuss how to apply refinement to AMR in Section 6.7.

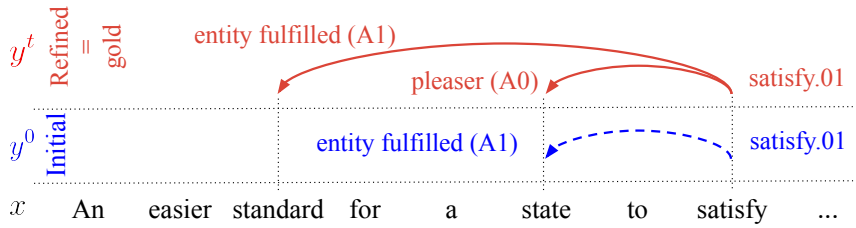


Figure 6.1: An example of structured refinement, the sentence fragment is from CoNLL-2009: the initial prediction by the factorized model in blue, the refined one (identical to the gold standard) in red.

Instead of modeling a restricted joint scoring function and adopting a specific iterative inference algorithm, we model interactions between argument labeling decisions through iterative refinement. Starting with an output produced by a strong factorized model, we iteratively refine it using a refinement network. Conceptually, since neural networks are universal function approximators (Hornik, 1991), such refinement steps can imitate an iterative decoding process that maximizes a joint score over the entire structure. Still, to avoid overfitting, instead of modeling arbitrary interactions among roles and words, we encode prior knowledge about the SRL problem by designing a restricted network architecture capturing interactions.

Iterative refinement strategies have been successful in machine translation (Lee et al., 2018; Novak et al., 2017; Xia et al., 2017; Hassan et al., 2018). A refinement step takes a translated sentence and the source sentence as input and produces a new translation. However, their effectiveness in other NLP tasks is yet to be demonstrated.[3] We conjecture that this discrepancy is due to differences in data availability. Given larger amounts of training data typically used in machine translation, their base models and refinement networks overfit to a lesser extent. Overfitting in (1) the base model and (2) the refinement network are both problematic. The first implies that either there are no mistakes in the base models in the training set or their distribution is very different from that in the test regime, so the training material for the inference networks ends up being misleading. The second naturally means that refinement will fail at test time. We address both these issues by designing restricted inference networks and adding a specific form of noise when training them.

Consider the example in Figure 6.1. The argument *'state'* appears in the highly ambiguous syntactic position '[.] *to satisfy*'. All three core semantic roles of the predicate SATISFY.01 can in principle appear here: patient (A1:ENTITY_FULFILLED,

[3] See extra discussion in Section 6.6.

as in '*a sweet tooth to satisfy*'), instrument (A2:METHOD, as in '*a little dessert to satisfy your sweet tooth*') and agent (A0: PLEASER, as in our actual example). The basic factorized model got it wrong, assigning A1 to the argument *'state'*. However, taking into account other arguments, the model can correct the label. The configuration 'A1 *to satisfy*' is more likely when an agent (A0) is present in the sentence. The lack of an agent boosts the score for the correct configuration 'A0 *to satisfy*'.

Our *iterative refinement* approach encodes the above intuition. In iterative refinement (Lee et al., 2018), a refinement network repeatedly takes previous output as input and produces its refined version. Formally, we have

$$y^{t+1} = \text{Refine}(x, y^t) \tag{6.1}$$

Naturally, such refinement strategy also requires an initial prediction y^0, which is produced by a ('base') factorized model. Our structured refinement network is simple but encodes non-local dependencies. Specifically, it takes into account the information about the semantic role distributions on the previous iteration aggregated over the entire sentences but not the information what the other arguments are. It is a coarse compressed representation of the prediction, yet it represents long-distance information not readily available within the factorized base model. While this is not the only possible design, we believe that the empirical gains from using this simple refinement network, demonstrate the viability of our general framework of iterative refinement with restricted inference networks.

Experimentally, we consider the entire CoNLL-2009 shared task (Hajič et al., 2009), which includes 7 languages and out-of-domain settings. We start with a strong factorized baseline model. Then, using our structure refinement network, we improve on this baseline on all 7 CoNLL-2009 languages. Compare to the previous best results, we obtained better performance on 5 out of 7 languages. We also observe improvements on out-of-domain test sets, confirming the robustness of our approach. We perform experiments demonstrating the importance of adding noise, and ablation studies showing the necessity of incorporating output interactions. Furthermore, we provide analysis on constraint violations (e.g., A0 can appear to most once) and errors on the English test set.

Our main contributions in this chapter are:

- We introduce a restricted inference network that iteratively refines predicates and argument labels for semantic role labeling;

- We empirically demonstrate that our improvements are consistent across languages. Importantly, the improvements are robust to domain change, unlike other similar methods.

The work in this chapter was published at EMNLP/IJCNLP 2019 (Lyu et al., 2019). The code can be accessed from https://github.com/ChunchuanLv/Iterative_Inference.

6.1 Preliminaries

In dependency SRL, for each sentence of length n, we have a sequence of words w, dependency labels dep,part-of-speech tags pos, each being a discrete sequence of length n. To simplify notation, we consider one predicate at a time. We denote the number of roles by r, it includes the 'null' role, signifying that the corresponding word is not an argument of the predicate. Formally, $\mathrm{P} \in \Delta_{m-1}$ is the probability distribution over m predicate senses, and Δ_{m-1} represents the corresponding probability simplex. We also have predicate sense embeddings $\Pi \in \mathbb{R}^{m \times d_\pi}$, and index j, throughout the discussion, refers to the position of the target predicate in the sentence. $\mathrm{R} \in \Delta_{r-1}^{n}$ is a matrix of size $n \times r$ such that each row sums to 1, corresponding to a probability distribution over roles. In particular $\mathrm{R}_{i,0}$ is the probability of i-th word not being an argument of the predicate.

We index role label and sense predictions from different refinement iterations ('time steps') with t, i.e. P^t and R^t. The index t ranges from 0 to T, and P^0 and R^0 denotes the predictions from the factorized baseline model.

6.2 Factorized Model

Similarly to recent approaches to SRL and semantic graph parsing (He et al., 2017; Li et al., 2019b; Dozat and Manning, 2018), our factorized baseline model starts with concatenated embeddings x. Then, we encode the sentence with a BiLSTM, further extract features with an MLP (multilayer perceptron) and apply a bi-affine classifier to the resulting features to label the words with roles. We also use a predicate-dependent linear layer for sense disambiguation.

More formally, we start with getting a sentence representation by concatenating embeddings. We have $x^{\mathrm{w}} \in \mathbb{R}^{n \times d_w}$, $x^{\mathrm{dep}} \in \mathbb{R}^{n \times d_\delta}$, $x^{\mathrm{pos}} \in \mathbb{R}^{n \times d_p}$ for words, dependency

labels and part-of-speech tags, respectively. We concatenate them to form a sentence representation:

$$x = x^{\mathrm{w}} \circ x^{\mathrm{dep}} \circ x^{\mathrm{pos}} \in \mathbb{R}^{n \times d_x} \tag{6.2}$$

We further encode the sentence with a BiLSTM:

$$\mathrm{h} = \mathrm{BiLSTM}(x) \in \mathbb{R}^{n \times d_h} \tag{6.3}$$

From these context-aware word representations, we produce features for argument identification and role labeling that will be used by a bi-affine classifier. Note that, for every potential predicate-argument dependency (i.e. a candidate edge), we need to produce representations of both endpoints: the argument and the predicate 'sides'. For the argument side, h^{ρ_0} will be used to compute the logits for argument identification and h^{ρ_1} will be used for deciding on its role:

$$\mathrm{h}^{\rho_0} = \mathrm{MLP}(\mathrm{h}) \in \mathbb{R}^{n \times d_{\rho_0}} \tag{6.4}$$

$$\mathrm{h}^{\rho_1} = \mathrm{MLP}(\mathrm{h}) \in \mathbb{R}^{n \times d_{\rho_1}} \tag{6.5}$$

Similarly, for the predicate side, we also extract two representations h^{ρ}_0 and h^{ρ_1} (recall that the predicate is at position j):

$$\mathrm{h}^{\rho_0} = \mathrm{MLP}(\mathrm{h}_j) \in \mathbb{R}^{d_{\rho_0}} \tag{6.6}$$

$$\mathrm{h}^{\rho_1} = \mathrm{MLP}(\mathrm{h}_j) \in \mathbb{R}^{d_{\rho_1}} \tag{6.7}$$

We then obtain logits I^{ρ_0} corresponding to the decision to label arguments as *null*, and logits I^{ρ_1} for other roles. So, we have:

$$\mathrm{I}^{\rho_0} = \mathrm{BiAffine}(\mathrm{h}^{\rho_0}, \mathrm{h}^{\rho_0}) \in \mathbb{R}^{n} \tag{6.8}$$

$$\mathrm{I}^{\rho_1} = \mathrm{BiAffine}(\mathrm{h}^{\rho_1}, \mathrm{h}^{\rho_1}) \in \mathbb{R}^{n \times (r-1)} \tag{6.9}$$

Unlike Dozat and Manning (2018), where argument identification and role labeling are trained with two losses,[4] we feed them together into a single softmax layer to compute the semantic-role distribution R^0:

$$\mathrm{I}^{\alpha} = \mathrm{I}^{\rho_0} \circ \mathrm{I}^{\rho_1} \in \mathbb{R}^{n \times r} \tag{6.10}$$

$$\mathrm{R}^0 = \mathrm{Softmax}(\mathrm{I}^{\alpha}) \in \Delta^{n}_{r-1} \tag{6.11}$$

[4] The separate processing of I^{ρ_0} and I^{ρ_1} rather than using a single MLP for all roles, including *null*, results in extra representation power allocated for the argument identification subtask.

Now, for sense disambiguation, we need to extract yet another predicate representation h^{π}:

$$\mathrm{h}^{\pi} = \mathrm{MLP}(\mathrm{h}_j) \in \mathbb{R}^{d_\pi} \tag{6.12}$$

In the formalism we use (PropBank), senses are predicate-specific, so we use predicate-specific sense embedding matrices Π. The matrix Π acts as a linear layer before softmax:

$$\mathrm{I}^{\pi} = \Pi \cdot \mathrm{h}^{\pi} \in \mathbb{R}^{m} \tag{6.13}$$

$$\mathrm{P}^{0} = \mathrm{Softmax}(\mathrm{I}^{\pi}) \in \Delta_{m-1} \tag{6.14}$$

This ends the description of our baseline model, which we also use to get initial predictions for iterative refinement.

6.3 Structured Refinement Network

In this section, we introduce the structured refinement network for dependency SRL. When doing refinement, it has access to the roles distribution $\mathrm{R}^t \in \Delta_{r-1}^{n}$ and the sense distribution $\mathrm{P}^t \in \Delta_{m-1}$ computed at the previous iteration (i.e. time t). In addition, it exploits the sentence representation $x \in \mathbb{R}^{n \times d_x}$. Our refinement network is limited and structured, in the sense that it only has access to a compressed version of the previous prediction, and the network itself is a simple MLP.

Similarly to our baseline model, we extract feature vectors g from input x and further separately encode the argument representation g^{α} and the predicate token representation g^{π}:

$$\mathrm{g} = \mathrm{BiLSTM}(x) \in \mathbb{R}^{n \times d_h} \tag{6.15}$$

$$\mathrm{g}^{\alpha} = \mathrm{MLP}(\mathrm{g}) \in \mathbb{R}^{n \times d_g} \tag{6.16}$$

$$\mathrm{g}^{\pi} = \mathrm{MLP}(\mathrm{g}_j) \in \mathbb{R}^{d_g} \tag{6.17}$$

To simplify the notation, we omit indexing them by t, except for R^t and P^t. We use two refinement networks, one for roles and another one for predicate senses.

6.3.1 Role Refinement Network

First, we describe our structured refinement network for role labeling. We use i to index arguments. We obtain a compressed representation o_i used for refining R_i^t by

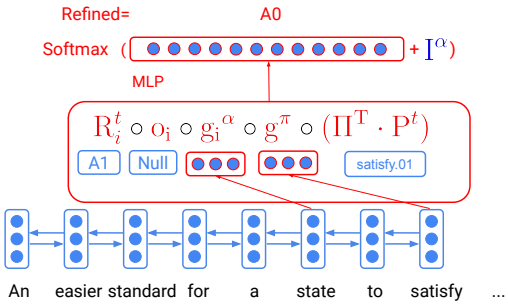


Figure 6.2: Illustration of Role Refinement Network

summing up the probability mass for all roles, excluding the null role:

$$\mathrm{o}_{i,u} = \sum_{k \neq i} \mathrm{R}^t_{k,u} \in \mathbb{R} \tag{6.18}$$

$$\mathrm{o}_i = [\mathrm{o}_{i,u}] \in \mathbb{R}^{r-1} \tag{6.19}$$

Intuitively, o_i is the aggregation of all other roles being labeled by the current predicate. We concatenate o_i with feature vectors of the current argument g^α, predicate g^π, the relaxed predicate sense embedding $\Pi^\intercal \cdot \mathrm{P}^t$ and the role probability itself (R_i) to form the input to a two-layer network:

$$\mathrm{z}^\alpha_i = \mathrm{R}^t_i \circ \mathrm{o}_i \circ \mathrm{g}^\alpha_i \circ \mathrm{g}^\pi \circ (\Pi^\intercal \cdot \mathrm{P}^t) \tag{6.20}$$

$$\mathrm{z}^\alpha_i \in \mathbb{R}^{2r-1+2d_g+d_\pi}$$

$$\mathrm{M}^\alpha_i = W^\alpha \cdot \sigma(W_\alpha \cdot \mathrm{z}^\alpha_i) \in \mathbb{R}^r \tag{6.21}$$

$$\mathrm{M}^\alpha = [\mathrm{M}^\alpha_{i,u}] \in \mathbb{R}^{n \times r}, \tag{6.22}$$

where σ is the logistic sigmoid function, $W_\alpha \in \mathbb{R}^{d_r \times (2r-1+2d_g+d_\pi)}, W^\alpha \in \mathbb{R}^{r \times d_r}$ are learned linear mappings. We obtain our refined logits M^α_i for the i-th argument; M^α refers to the stacked matrix of logits for all arguments. To obtain the refined role distribution, we add up M^α and I^α that we got from the baseline model, and follow that by a softmax layer:

$$\mathrm{R}^{t+1} = \mathrm{Softmax}(\mathrm{M}^\alpha + \mathrm{I}^\alpha) \in \Delta^n_{r-1} \tag{6.23}$$

We summarize the role refinement network in Figure 6.2.

6.3.2 Sense Refinement Network

To build a representation for sense disambiguation, we simply compute the probability mass for each role (excluding the null role) to obtain r^π, and concatenate this with g^π

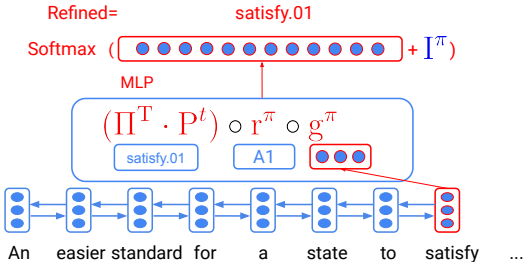


Figure 6.3: Illustration of Sense Refinement Network

and $\Pi^\intercal \cdot \mathrm{P}^t$:

$$\mathrm{r}^\pi = \sum_k \mathrm{R}^t_{k,1:} \in \mathbb{R}^{r-1} \tag{6.24}$$

$$\mathrm{z}^\pi = (\Pi^\intercal \cdot \mathrm{P}^t) \circ \mathrm{r}^\pi \circ \mathrm{g}^\pi \in \mathbb{R}^{r-1+d_g+d_\pi} \tag{6.25}$$

Differently from the role refinement network, sense prediction is predicate-specific. Therefore, we first map z^π to $\mathbb{R}^{d_\pi}$, and then take the inner product with the predicate-specific sense embeddings Π to get the refined logits:

$$\mathrm{M}^\pi = \Pi \cdot W^\pi \cdot \sigma(W_\pi \cdot \mathrm{z}^\pi) \in \mathbb{R}^m \tag{6.26}$$

Similarly to role refinement, σ is the logistic function, $W_\pi \in \mathbb{R}^{d_r \times (r-1+d_g+d_\pi)}, W^\pi \in \mathbb{R}^{m \times d_r}$ are learned linear mappings. Again, we combine the logits M^π and I^π before the softmax layer:

$$\mathrm{P}^{t+1} = \mathrm{Softmax}(\mathrm{M}^\pi + \mathrm{I}^\pi) \in \Delta_{m-1} \tag{6.27}$$

We summarize the sense refinement network in Figure 6.3.

6.3.3 Weight Tying

We use the weight-tying regularization technique popular with the denoising autoencoders (Vincent et al., 2008; Kamyshanska and Memisevic, 2015). Weight-tying ties the input and output embeddings to reduce the number of parameters. Weight tying has also been applied in language modeling (Press and Wolf, 2017; Inan et al., 2017) and machine translation (Sennrich et al., 2017). We believe that the technique may be even more effective here as the amount of labeled data for SRL is lower than in many usual applications of weight typing. We tie W_α with a subset of W^α rows: specifically with the rows acting on R^t_i in the computation of M^α_i (see equations 6.20 and 6.21).

Similarly, we tie W_π with the part of W^π corresponding to $\Pi^\intercal \cdot \mathrm{P}^t$ (see equations 6.25 and 6.26). Formally,

$$W^\alpha = W_\alpha^\intercal[:r] \tag{6.28}$$

$$W^\pi = W_\pi^\intercal[:d_\pi] \tag{6.29}$$

where $W[:k]$ takes the first k rows of matrix W.

6.3.4 Self Refinement

We describe a simpler version of the refinement network which we will use in experiments to test whether the improvements with the structured refinement network over the factorized baseline are genuinely coming from modeling interaction between arguments rather than from simply combining multiple classifiers. This simpler refinement network does not account for any interactions between arguments. Instead of equations 6.20 and 6.25, we have:

$$\mathrm{z}_i^\alpha = \mathrm{R}_i^t \circ \mathrm{g}_i^\alpha \circ \mathrm{g}^\pi \in \mathbb{R}^{r+2d_g+d_\pi} \tag{6.30}$$

$$\mathrm{z}^\pi = (\Pi^\intercal \cdot \mathrm{P}^t) \circ \mathrm{g}^\pi \in \mathbb{R}^{d_g+d_\pi} \tag{6.31}$$

Everything else is kept the same as in the full model, expect that the size of W^α and W^π needs to be adjusted. We refer to this ablated network as the *self-refinement network*.

6.4 Training for Iterative Structure Refinement

In this section, we describe our training procedure.

6.4.1 Two-Stage Training

We have two models: the baseline model, producing the initial predictions, and the iterative refinement network, correcting them. While it is possible to train them jointly, we find joint training slow to converge. Instead, we train the factorized baseline model first and then optimize the refinement networks while keeping the baseline model fixed.

6.4.2 Stochastic Training

Our baseline model overfits to the training set, and, if simply training on its output, our refinement network would learn to copy the base predictions. Instead, we perturb the baseline prediction during training. Naturally, we can add dropout (Srivastava

et al., 2014) and recurrent dropout (Gal and Ghahramani, 2016) to our neural networks. However, for the smaller data set we use, we find this not sufficient. In particular, we use Gumbel-Softmax instead of Softmax. Gumbel-Softmax$(\mathrm{I}) = \mathrm{Softmax}(\mathrm{I} + \lambda_g \varepsilon)$, where the random variable ε is drawn from the standard Gumbel distribution (Maddison et al., 2017; Jang et al., 2017), and λ_g is a hyper-parameter controlling decoding stochasticity.[5]

6.4.3 Loss for Iterative Refinement

Let us denote gold-standard labels for roles and predicates as R^* and P^*. We use two separate losses $\mathcal{L}_{\text{base}}(\mathrm{R}^*, \mathrm{P}^*, x)$ and $\mathcal{L}_{\text{refine}}(\mathrm{R}^*, \mathrm{P}^*, x)$ for our two-stage training. We define losses for predictions from each refinement iteration and sum them up:

$$\mathcal{L}_{\text{base}}(\mathrm{R}^*, \mathrm{P}^*, x) = \mathcal{L}(\mathrm{R}^*, \mathrm{R}^0) + \mathcal{L}(\mathrm{P}^*, \mathrm{P}^0) \tag{6.32}$$

$$\mathcal{L}_{\text{refine}}(\mathrm{R}^*, \mathrm{P}^*, x) = \sum_{t=1}^{T} \mathcal{L}(\mathrm{R}^*, \mathrm{R}^t) + \mathcal{L}(\mathrm{P}^*, \mathrm{P}^t) \tag{6.33}$$

We adopt the Softmax-Margin loss (Gimpel and Smith, 2010; Blondel et al., 2020) for individual $\mathcal{L}$. Effectively, we subtract 1 from the logit of the gold label, and apply the cross entropy loss.

6.5 Experiments

Datasets We conduct experiments on CoNLL-2009 (Hajič et al., 2009) data set for all languages, including Catalan (Ca), Chinese (Zh), Czech (Cz), English (En), German (De), Japanese (Jp) and Spanish (Es). We use the predicted part-of-speech tags, dependency labels,[6] and pre-identified predicate, provided with the dataset. The statistics of datasets are shown in Table 6.1.

Hyperparameters We use ELMo (Peters et al., 2018b) for English, and FastText embeddings (Bojanowski et al., 2017; Grave et al., 2018) for all other languages. We use the predicted part-of-speech tags, dependency labels, and pre-identified predicate, provided with the dataset. We train and run the refinement networks for two iterations. All other hyper-parameters are the same for all languages, except BiLSTMs for English is larger than others. See Appendix D for more details.

[5] A more canonical way of controlling stochasticity is to use the temperature but we prefer not to scale the gradient.

[6] In retrospect, using the predicated dependency label is a mistake, as the dependency tree parser (Nivre et al., 2007) is not very reliable.

	#sent	#pred	#pred/#sent
Ca	13200	37444	2.84
Cz	38727	414133	10.69
De	36020	17400	0.48
En	39279	179014	4.56
Ja	4393	25712	5.85
Es	14329	43828	3.06
Zh	22277	102827	4.62

Table 6.1: Number of sentences and predicates in training set of different languages.

Training Details Training the refinement network takes roughly 2 times more time than the baseline models, as it requires running BiLSTMs. The extra computation for the structured refinement network is minimal. For English, training the iterative refinement model for 1 epoch takes about 6 minutes on one 1080ti GPU. Adam is used as the optimizer (Kingma and Ba, 2015), with the learning rate of 3e-4. We use early stopping on the development set. We run 600 epochs for all baseline models, and 300 epochs for the refinement networks. Batch sizes are chosen from 32, 64, or 128 to maximize GPU memory usage. Our implementation is based on PyTorch and AllenNLP (Paszke et al., 2019; Gardner et al., 2018).

6.5.1 Results and Discussions

Test Results Results for all CoNLL-2009 languages on the standard (in-domain) datasets are presented in Table 6.2. We compare our best model to the best previous single model for the corresponding language (excluding ensemble ones). Most research has focused on English, but we include results of recent models which were evaluated on at least 3 languages. When compared to the previous models, both our models are very competitive, with the exception of German. On the German dataset, Mulcaire et al. (2018) also report a relatively weak result, when compared to Roth and Lapata (2016). The German dataset is the smallest one in terms of the number of predicates. Syntactic information used by Roth and Lapata (2016) may be very beneficial in this setting and may be the reason for this discrepancy. Our structured refinement approach improves over the best previous results on 5 out of 7 languages. Note that hyper-parameters of the refinement network are not tuned for individual languages, suggesting that the proposed method is robust and may be easy to apply to new languages and/or new base

Model	Ca	Cz	De	En	Ja	Es	Zh	Avg.
Watanabe et al. (2010)	79.6	86.0	79.6	85.0	78.7	79.3	77.2	
Roth and Lapata (2016)	-	-	**80.1**	86.7	-	80.2	79.4	-
Marcheggiani et al. (2017)	-	86.0	-	87.7	-	80.3	81.2	-
Mulcaire et al. (2018)*	79.5	85.1	70.0	87.2	76.0	77.3	81.9	79.6
Previous best single model	80.3	86.0	**80.1**	90.4	78.7	80.5	**84.3**	82.9
Baseline model	80.7	87.3	75.1	90.7	82.0	79.9	83.3	82.7
Structured refinement	**80.9**	**87.6**	75.9	**91.0**	**82.5**	**80.5**	83.3	**83.1**

Table 6.2: Labeled F1 score (including senses) for all languages on the CoNLL-2009 in-domain test set. For previous best result, Catalan is from Zhao et al. (2009), Japanese is from Watanabe et al. (2010), Czech is from Henderson et al. (2013), German and Spanish are from Roth and Lapata (2016), English is from Li et al. (2019b) and Chinese is from Cai et al. (2018). We report the best testing results from Mulcaire et al. (2018).

models. The only case where the refinement network was not effective is Chinese.

English	Test	Ood
Li et al. (2019b)	90.4	81.5
Baseline	90.7	82.0
Structured Refinement	**91.0**	**82.2**
German	Test	Ood
Zhao et al. (2009)	**76.2**	**67.8**
Baseline	75.1	65.3
Structured Refinement	75.9	65.7
Czech	Test	Ood
Marcheggiani et al. (2017)	86.0	**87.2**
Baseline	87.3	85.8
Structured Refinement	**87.6**	86.0

Table 6.3: Labeled F1 scores (including senses) on English, German, Czech in-domain and out-of-domain test sets; we chose the previous models achieving the best scores on the out-of-domain test sets.

Out-of-Domain Results on the out-of-domain testing sets are presented in Table 6.3.[7]

[7] Roth and Lapata (2016) has better in-domain testing score, but did not report the out-of-domain score.

Model	Ca	Cz	De	En	Ja	Es	Zh	Avg.
Baseline	81.7	88.4	74.0	89.6	83.0	80.5	85.3	83.2
Full	**82.1**	88.6	75.0	89.8	**83.6**	**81.2**	**85.5**	**83.7**
1 iteration	82.1	88.6	**75.1**	89.9	83.5	81.03	85.5	83.4
un-tied	82.0	88.6	75.04	89.8	83.5	80.9	85.5	83.6
no Gumbel	82.1	**88.7**	74.6	**90.1**	83.3	80.6	85.4	83.5

Table 6.4: Labeled F1 score (including senses) for all languages on development set for different configurations.

We observe improvements from using refinement in all the cases. This shows that our refinement approach is robust against domain shift.

Ablations We report development set results in different settings in Table 6.4. Our full model performs 2 refinement iterations, uses weight tying, and the Gumbel noise.[8] We select the best configuration for each language to report the test set performance in Table 6.2 and Table 6.3. As expected, weight tying is beneficial for lower-resource languages such as Catalan, Japanese and Spanish (see Table 6.1 for dataset characteristics). The Gumbel noise helps for all languages except for Czech and English, the two largest datasets. In particular, we observe almost no improvement on the Spanish dataset without using the Gumbel noise. We note relatively consistent but small gains from using 2 refinement iterations. The magnitude of the gains may be an artifact of us having the loss terms $\mathcal{L}(\mathrm{R}^*, \mathrm{R}^t)$ and $\mathcal{L}(\mathrm{P}^*, \mathrm{P}^t)$, encouraging not only the final (second), but also the first, iteration to produce accurate predictions. A potential alternative explanation is that our refinement network is restricted to simple interactions, resulting in the fixed point reachable in one step.

Model	U	C	R
Gold	55	0	88
Baseline	301	2	114
Structured Refinement	142	2	111

Table 6.5: Unique core roles violations (U), continuation roles violations (C) and reference roles violations (R) on English in-domain test set.

Constraints Violation We consider violation of unique core roles (U), continuation

[8] We set $\lambda_g^{\alpha} = 5$ for role and $\lambda_g^{\pi} = 50$ for sense, so that initial predictions contain around 20% errors.

roles (C) and reference roles (R) constraints from Johansson and Nugues (2008); Punyakanok et al. (2008); FitzGerald et al. (2015) in Table 6.5. U is violated if a core role (A0 - A5, AA) appears more than once; C is violated when the C-X role is not preceded by the X role (for some X); R is violated if R-X role does not appear. Our approach results in a large reduction in the uniqueness constraint violations. Our model slightly reduces the number of R violations, while He et al. (2017) reported that deterministically enforcing constraints is not helpful (albeit in span-based SRL). However learning those constraints in a soft way might be beneficial.

Model	Ca	Cz	De	En	Ja	Es	Zh	Avg.
Baseline model	80.7	87.3	75.1	90.7	82.0	79.9	**83.3**	82.7
Self refinement	80.7	87.3	74.8	90.7	82.3	80.1	**83.3**	82.7
Structured refinement	**80.9**	**87.6**	**75.9**	**91.0**	**82.5**	**80.5**	**83.3**	**83.1**

Table 6.6: Labeled F1 score (including senses) for all languages on the CoNLL-2009 in-domain test set.

Argument Interaction vs. No Argument Interaction We compare the structured refinement network and the self-refinement network in Table 6.6. Both networks share the same hyper-parameters. The structured refinement network consistently outperforms the self-refinement counterpart. This suggests that the refinement model benefits from accessing information about other arguments when doing refinement. In other words, modeling argument interaction appears genuinely useful.

Model	RP	RR	Sense
Baseline	88.1	88.3	96.2
Structured Refinement	88.7	88.5	96.3

Table 6.7: Labeled roles precision (RP), recall (RR) and sense disambiguation accuracy (Sense) on English in-domain test set.

Improvement Decomposition We report labeled role precision, recall and sense disambiguation accuracy in Table 6.7. Our structured refinement approach consistently improves over the baseline model in all metrics. While we cannot assert the improvements on all metrics are significant, this suggests that it learns some non-trivial interactions instead of merely learning to balance precision and recall.

Error Correction Analysis We show the errors that the structured refinement network corrects in Figure 6.4. In the baseline confusion matrix, we see the errors are

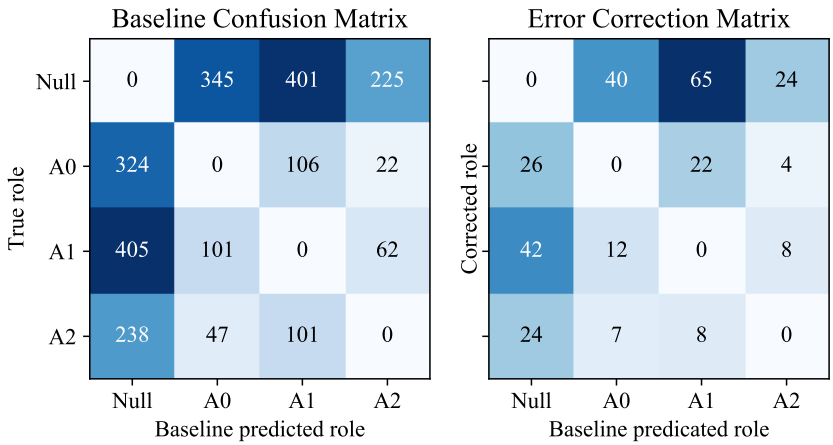


Figure 6.4: Confusion matrix for the baseline model, and a correction matrix where the errors were corrected by the refinement network. Only Null, A0, A1, A2 are presented here.

fairly balanced for all the roles we consider here. In the error correction matrix, the corrections are also fairly evenly distributed. Yet, this is not completely uniform. There is a tendency towards filtering out arguments rather than generating new ones.

6.5.2 Subsequent Results

The work in this chapter is done in 2019. It is worth to look at a subsequent work on semantic role labeling that also models high-order statistics. Li et al. (2020) models the second-order statistics on selected features, including sibling, co-parents. Sibling means argument pair of the same predicate, and co-parents denote predicates sharing the same argument. Unlike us, they consider all the predicate-argument structure in a sentence jointly.

We compare their results with ours in Table 6.8. Notably, with the use of BERT, Li et al. (2020) (+H) achieve much better results than all the previous results, including ours. Importantly, Li et al. (2020)+H, the models with high-order statistics, outperform their respective baselines that use BERT. Such improvement is consistent with our finding that modeling high-order interaction could be useful for semantic role labeling. However, in both their model and our model, the improvements are modest.

We compare results from Li et al. (2020) with ours on the out-of-domain testing sets

Model	Ca	Cz	De	En	Ja	Es	Zh	Avg.
Previous best single model	80.3	86.0	80.1	90.4	78.7	80.5	84.3	82.9
Baseline model	80.7	87.3	75.1	90.7	82.0	79.9	83.3	82.7
Structured refinement	80.9	87.6	75.9	91.0	82.5	80.5	83.3	83.1
Li et al. (2020)	86.4	91.5	85.2	91.2	85.6	86.6	88.2	87.8
Li et al. (2020)+H	**86.9**	**91.9**	**85.5**	**91.8**	**85.9**	**87.0**	**88.7**	**88.2**

Table 6.8: Labeled F1 score (including senses) for all languages on the CoNLL-2009 in-domain test set. Li et al. (2020) is a baseline model with BERT, and Li et al. (2020)+H models high-order statistics.

English	Test	Ood
Baseline	90.7	82.0
Structured Refinement	91.0	82.2
Li et al. (2020)	91.2	84.8
Li et al. (2020)+	**91.8**	**85.1**
German	Test	Ood
Baseline	75.1	65.3
Structured Refinement	75.9	65.7
Li et al. (2020)	85.2	70.9
Li et al. (2020)*	**85.5**	**71.9**
Li et al. (2020)+	84.6	69.7
Czech	Test	Ood
Baseline	87.3	85.8
Structured Refinement	87.6	86.0
Li et al. (2020)	91.5	91.6
Li et al. (2020)*	**91.6**	**91.7**
Li et al. (2020)+	91.6	91.5

Table 6.9: Labeled F1 scores (including senses) on English, German, Czech in-domain and out-of-domain test sets; we chose the previous models achieving the best scores on the out-of-domain test sets. Li et al. (2020) is a baseline model with BERT, Li et al. (2020) +H models high-order statistics on all features, Li et al. (2020)* models high-order statistics on selected features that achieved the best score.

in Table 6.9. While we observe improvements from using refinement in all the cases, the results from Li et al. (2020) are more mixed. In particular, the base model from Li et al. (2020) has higher scores than the full model on both German and Czech. It seems that using high-order features might still make the model vulnerable to overfitting. While there are other feature choices that can yield improvements, it is not clear how to choice the features a priori.

6.6 Related Work

Before the rise of deep learning methods, the most accurate SRL methods relied on modeling high-order interactions in the output space (e.g., between arguments or arguments and predicates) (Toutanova et al., 2008; Watanabe et al., 2010). Earlier neural methods can model such output interactions through a transition system, and achieve competitive performance (Henderson et al., 2013). However, most previous state-of-the-art SRL systems use powerful sentence encoders (e.g., layers of LSTMs (Li et al., 2019b; He et al., 2017) or multi-head self-attention (Strubell et al., 2018)) and factorize over small fragments of the predicted structures. Specifically, most modern models process individual arguments and perform predicate disambiguation independently. The trend towards more factorizable models is not unique to dependency-based SRL but common for most structured prediction tasks in NLP (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017, 2018). One major exception is language generation tasks, especially machine translation and language modeling, where larger amounts of text are typically used in training. Another exception is the very recent advance in neural auto-regressive models for AMR parsing (Zhang et al., 2019a; Cai and Lam, 2020), as we discussed in previous chapters.

Learning to refine predictions from neural structured prediction models has recently received significant attention. Our approach bears similarity to methods used in machine translation (Lee et al., 2018; Novak et al., 2017; Xia et al., 2017). All these methods refine a translated sentence produced by a seq2seq model with another seq2seq model. Among them, the deliberation networks by Xia et al. (2017) rely on BiLSTMs and improve initial predictions from an competitive baseline and obtain state-of-art-results on English-to-French translation. Later, it has been shown that the deliberation networks can improve translation when used within the Transformer framework (Hassan et al., 2018).

Certain approaches, not necessarily directly optimized for refinement, can never-

theless be regarded as iterative refinement methods. Structured prediction energy networks (SPENs) are trained to assign global energy scores to output structures, and the gradient descent is used during inference to minimize the global energy (Belanger and McCallum, 2016). As the gradient descent involves iterative optimization, its steps can be viewed as iterative refinement. In particular, Belanger et al. (2017) build a SPEN for SRL, but for the span-based formalism, not the dependency one we consider in this work. While they improve over their baseline model, their baseline model used multi-layer perceptron to encode local factors, thus the encoder power is limited. Moreover their refined model performs worse in the out-of-domain setting than their baseline model, indicating overfitting (Belanger et al., 2017).

In the follow-up work, Tu and Gimpel (2018, 2019) introduce inference networks to replace gradient descent. Their inference networks directly refine the output. Improvements over competitive baselines are reported on part-of-speech tagging, named entity recognition and CCG super-tagging (Tu and Gimpel, 2019). However, their inference networks are distilling knowledge from a tractable linear-chain conditional random field (CRF) model. Thus, these methods do not provide direct performance gains. More importantly, the interactions captured in these models are likely local, as they learn to mimic Markov CRFs.

Denoising autoencoders (Vincent et al., 2008) can also be used to refine structure. Indeed, image segmentation can be improved through iterative inference with denoising autoencoders (Romero et al., 2017; Drozdzal et al., 2018). Their framework is very similar to ours, albeit we are working in a discrete domain. One other difference is that by using a convolutional architecture in the refinement network, they are still modeling only local interactions. At a more conceptual level, Bengio et al. (2013b) argued that a denoising autoencoder should not be too robust to the input variations as to ignore the input. This indicates that we should not expect refinement networks to correct all the errors, even in theory, and hence, the refinement networks do not need to be particularly powerful.

Second-order statistical modeling has been applied to syntactic dependency parsing (McDonald and Pereira, 2006) prior to the rise of deep learning models. Very recently, Fonseca and Martins (2020) estimated a global score on dependency tree, and optimize with a hinge loss. Wang et al. (2019) used second-order statistical model for semantic dependency parsing (Oepen et al., 2015), and obtain improvements over strong baseline using BiLSTM. They attempted loopy belief propagation and mean field variational inference for inference, and train the model end to end. Li et al.

(2020) that we discussed in the previous section actually applied this technique to semantic role labeling, and achieved the state-of-the-art results. Compared to Belanger and McCallum (2016); Belanger et al. (2017), the restriction from fully flexible energy function to the second-order scoring provides certain inductive bias. Compared to ours, the inference can also be regarded as iterative structure refinement. Yet, we do not provide a global score and directly try to model the refinement. In principle, our formalization should give us more liberty in terms of designing the refinement network.

6.7 Summary

To overcome the limitation of the independence assumption in graph-based parser, we propose the structured refinement network for dependency semantic role labeling. The structured refinement network corrects predictions made by a strong factorized baseline model while modeling interactions in the predicated structure. The resulting model outperforms the previous state-of-the-art results on 5 out of 7 languages in the CoNLL-2009, and substantially outperforms the factorized model on all of these languages. Analysis shows that instead of simple re-balancing recall and precision, the structured refinement network improves both metrics.

With recent results from others Wang et al. (2019); Fonseca and Martins (2020); Li et al. (2020), we could be more hopeful for further research on modeling high-order interaction in semantic parsing. In particular, applying the refinement technique to AMR parsing is feasible. Refining the semantic relation labels given the concept nodes is a trivial extension of the current work. More interestingly, we would like to refine our node identification model. Considering the locally auto-regressive node generation model from the last chapter, we can fix the maximum number of generated nodes. In such cases, we could use terminal nodes as placeholders the same way we use null roles in this chapter. Yet, it is perhaps more beneficial to investigate a more effective method for modeling the refinement, which we will discuss in the last chapter.

Chapter 7

Conclusion and Future Directions

7.1 Conclusions

This thesis investigated the graph-based approach for broad-coverage semantic parsing. Graph-based parsing factorizes the prediction of a graph into the predictions of its parts. The predictions are conditioned on the anchoring of nodes to the words. Graph-based parsers are sequence taggers at test time. Therefore, it is easy to use and results in a simple and interpretable parsing process. However, training a graph-based parser for formalisms such as AMR involves a preprocessing step that anchors the graph in the sentence. Instead of approaching this issue in a pipeline manner, we jointly train the graph-based parser while inducing such an anchoring, relying on variational inference and stochastic softmax. Still, the simplicity of graph-based parsing comes at the cost of limited expressive power. We overcome this limitation by iteratively refining the output, conditioning on the previous prediction.

In Chapter 4, relying on a hand-crafted re-categorization system that collapses AMR subgraphs into re-categorized nodes, we introduced a graph-based AMR parser trained by jointly modeling alignments, concepts, and relations. The re-categorization system segments the AMR graph so that the alignment between subgraphs (i.e., the re-categorized node) to words is injective. Treating the alignment as a latent variable, we sample the latent alignment between subgraphs and words from the Gumbel-Sinkhorn distribution. Consequently, the training is end-to-end differentiable. The resulting parser significantly improves the previous state of the art, and ablation tests show that joint modeling results in a higher score than using pre-fixed alignments.

In Chapter 5, we eliminated the need for the hand-crafted segmentation system used in the previous chapter. We cast the alignment and segmentation as generation-

order induction and jointly model the generation order, concepts, and relations in a variational inference framework. Similarly, the model is end-to-end differentiable with the stochastic softmax. To successfully apply stochastic softmax to our problem, we derived an efficient optimizer as an instance of Bregman's method. Empirically, our method outperforms a strong heuristic baseline and approaches the performance of the hand-crafted rule system. Notably, the concept identification performance approaches that of the current state of the art systems.

In Chapter 6, we addressed the limitation posed by the independence assumption in graph-based parsing. We proposed a structured refinement network for dependency semantic role labeling. The structured refinement network corrects predictions made by a strong factorized baseline model by modeling interactions in the predicted structure. The resulting model outperforms the previous state-of-the-art results on five out of seven languages and outperforms the factorized models in both in-domain and out-of-domain settings. Importantly, instead of simply re-balancing recall and precision, our refinement improves both metrics.

Overall, graph-based parsing can be completely data-driven without domain knowledge, and its expressiveness can be further improved with new techniques.

7.2 Future Directions

We discuss a few future specific research directions.

End-to-end Graph-based Semantic Parsing While we have been arguing that our end-to-end training method for AMR parsing is general, applying them to other meaning representations is left for future work. In particular, our alignment and segmentation induction method makes very few assumptions about the nature of the graphs. So beyond broad-coverage meaning representation, it may be effective in other tasks that can be framed as graph prediction (e.g., executable semantic parsing (Liang, 2016) or scene graph prediction (Xu et al., 2017)).

However, our methods still heavily rely on the copy mechanism to provide an inductive bias for inducing the latent structures (through masking). Therefore, we still have one step of preprocessing. Furthermore, as the modern contextualized sentence encoders (e.g., BERT) break the word into subwords, such a word-level copy mechanism might be inappropriate. It would be appealing to replace the copy mechanism with a seq-to-seq neural module on the character level that generates the concept

lemma.

Beyond the scope of semantic parsing in NLP, scene graph generation (Xu et al., 2017) in computer vision involves predicting scene graphs from the boxed region of images. In particular, if the boxed region is not given, scene graph generation could involve alignment (Zareian et al., 2020). Furthermore, scene graph generation can be improved with subgraphs that are implicit (Li et al., 2018).

Reentrancy in Graph-based AMR Parsing Focusing on AMR parsing, there is still one major phenomenon that our graph-based model does not handle properly: reentrancy. AMR captures co-reference through reentrancy. However, reentrancies do not necessarily come from co-reference. Reentrancies could come from implicit roles. Implicit roles cause reentrancies when the trigger word only appears once in the sentence.[1] For example, "The boy wants to be believed by the girl" (implicit role) has the same AMR as "The boy wants the girl to believe him" (co-reference):

```
(w / want-01
    :ARG0 (b / boy)
    :ARG1 (b2 / believe-01
                :ARG0 (g / girl)
                :ARG1 b))
```

Now, as the alignment is never given, the AMR graph alone does not tell us the cause of reentrancy.[2] This ambiguity could cause trouble for graph-based parsing. While in the case of the implicit role, the graph-based parsing handles the phenomena quite easily. However, suppose reentrancy is caused by co-reference. One ideal solution is that we combine the two 'boy' concept nodes after they are identified. Naturally, we would like to treat this ambiguity as a latent variable when reentrancy is found. For example, we relax the number of aligned tokens for nodes with reentrancy and treat the alignment as the latent variable.

End-to-end Differentiable Grammar-based Parsing In addition to graph-based parsing, grammar-based parsing also relies on discrete latent structures. The grammar-based parsing uses lexical rules to produce compositional graph fragments from tokens and then combines them compositionally given a grammar. Therefore, training a grammar-based parser needs a decomposition of the graph (Groschwitz, 2019). While

[1] More precisely, the trigger word appears less than the number of incoming edges

[2] Szubert et al. (2020) did a classification of causes based on linguistic phenomena (e.g., co-reference, control).

grammar constraints the combination rules of graph fragments and consequently constraints the decomposition of a graph into fragments and their combinations, such a decomposition is underdetermined. Modeling latent structure in grammar-based parsing might be considerably more difficult than that in graph-based parsing due to constraints on combinations of fragments. Still, it is desirable to have end-to-end training for the neural grammar-based parsing, as a learning-based method can help avoid error propagation caused by heuristics.

Prior to the rise of neural parsers, the Combinatory Categorial Grammar (CCG) achieved good performance on AMR parsing (Artzi et al., 2015). Following earlier works on CCG grammar induction (Zettlemoyer and Collins, 2005; Kwiatkowksi et al., 2010; Kwiatkowski et al., 2011),[3] Artzi et al. (2015) marginalized over latent derivations in CCG with dynamic programming.[4] On a conceptual level, the neural parameterization still yields a log-linear model on the derivation (Lewis et al., 2016). Hence, it appears that marginalization over latent derivations can still be carried out by dynamic programming. The implementation can be facilitated with Torch-Struct (Rush, 2020) that enables auto-differentiation through dynamic programming.

More recently, Apply-Modify (AM) algebra has been the most successful grammar for AMR parsing (Groschwitz et al., 2018) and other graph-based meaning representations (Lindemann et al., 2020). As pointed out by Lindemann et al. (2019), it is worth exploring latent variable models for the graph decomposition in AM algebra to avoid manual construction of decomposition heuristics for each meaning representation. As AM algebra permits flexible compositions, marginalization is likely to be intractable. Either differentiable sampling or REINFORCE (Williams, 1992) is needed for training.

(Not) Designing Refinement Network We argued in Chapter 6 that a purposefully designed refinement network could be advantageous because it incorporates inductive bias from expert knowledge/intuition. However, it is still appealing if the design of a refinement network can be standardized. We observe that our refinement adds 'refining' logits to the original logits produced by a baseline. This is structurally similar to residual networks (He et al., 2016) for image classification, which can be naturally applied to graph structure (Xu et al., 2018). As the probabilistic graphical models (Wainwright

[3] Due to the functional nature of CCG lexicons, CCG grammar permits an infinite number of possible derivations. Consequently, restrictions have to be put (Kwiatkowksi et al., 2010; Kwiatkowski et al., 2011). Such restrictions can manifest themselves as rules that generate candidate lexicons (Zettlemoyer and Collins, 2005).

[4] In the follow-up work, Misra and Artzi (2016) suggested that the neural parameterization causes trouble for the latent variable approach. We suspect that the perceived difficulties come from the implementation challenge, rather than theoretical issues.

and Jordan, 2008) can encode expert knowledge about the dependency between output variables, it would be interesting to apply a residual neural network over a graph (as in probabilistic graphical models) defined by expert knowledge. The resulting refinement would be similar to the traditional message passing inference, but it does not explicitly specify the underlying distribution.

Semi-supervised Learning So far, the improvements from refinement have been modest. As we have been arguing, one limiting factor might be the amount of available data. Therefore, combing semi-supervised learning with modeling high order interactions might be useful. The variational inference approach for semi-supervised learning is to treat the output structure as a latent variable. To parameterize expressive priors on structured continuous variables, *flows* apply invertible transformations to the continuous variables sampled from a factorized prior (Rezende and Mohamed, 2015). Such flows have been used to parameterize the probability of discrete structures. This allows the optimization a factorized probability distribution over discrete structures (e.g., graphs) with supervision (Liu et al., 2019a; Shi et al., 2020). With stochastic softmax, we can sample differentiable structured discrete variables from the logit space parameterized by flows. Hence, an expressive prior over structured discrete variables can be optimized without direct supervision. Furthermore, as a special case of the flow models, residual flows (Chen et al., 2019) have invertible transformations based on residual neural networks like architecture. The intriguing question is whether there can be a connection between the flows that generate a structure and the flows that refine a structure.

Bibliography

Abend, Omri and Ari Rappoport. 2013. 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.

Abzianidze, Lasha, 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.

Agrawal, Akshay, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J. Zico Kolter. 2019. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, volume 32, pages 9562–9574. Curran Associates, Inc.

Anchiêta, Rafael and Thiago Pardo. 2018. Towards AMR-BR: A SemBank for Brazilian Portuguese language. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Andreas, Jacob, Andreas Vlachos, and Stephen Clark. 2013. Semantic parsing as machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 47–52, Sofia, Bulgaria. Association for Computational Linguistics.

Artzi, Yoav, 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.

Azin, Zahra and Gülşen Eryiğit. 2019. Towards Turkish Abstract Meaning Representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 43–47, Florence, Italy. Association for Computational Linguistics.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

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

Ballesteros, Miguel 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.

Banarescu, Laura, 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.

Banarescu, Laura, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2017. Abstract meaning representation (AMR) 1.2.4 specification.

Banarescu, Laura, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2019. amr-guidelines.

Barratt, S. 2018. On the differentiability of the solution to convex optimization problems. *arXiv: Optimization and Control*.

Basile, Valerio 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.

Basile, Valerio, 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).

Belanger, David and Andrew McCallum. 2016. Structured prediction energy networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 983–992. JMLR.org.

Belanger, David, Bishan Yang, and Andrew McCallum. 2017. End-to-end learning for structured prediction energy networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 429–439. JMLR.org.

Bengio, Yoshua, Nicholas Léonard, and Aaron C. Courville. 2013a. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432.

Bengio, Yoshua, Li Yao, Guillaume Alain, and Pascal Vincent. 2013b. Generalized denoising auto-encoders as generative models. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, pages 899–907, USA. Curran Associates Inc.

Beschke, Sebastian and Wolfgang Menzel. 2018. Graph algebraic Combinatory Categorial Grammar. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 54–64, New Orleans, Louisiana. Association for Computational Linguistics.

Birkhoff, Garrett. 1946. Tres observaciones sobre el algebra lineal [three observations on linear algebra]. *Univ. Nac. Tucuman. Revista A.*, 5.

Blloshmi, Rexhina, Rocco Tripodi, and Roberto Navigli. 2020. XL-AMR: Enabling cross-lingual AMR parsing with transfer learning techniques. In *Proceedings of the*

2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2487–2500, Online. Association for Computational Linguistics.

Blondel, Mathieu, AndrÃ© F.T. Martins, and Vlad Niculae. 2020. Learning with fenchel-young losses. *Journal of Machine Learning Research*, 21(35):1–69.

Bod, Rens, Remko Scha, Khalil Sima'an, et al. 2003. *Data-oriented parsing*. University of Chicago Press.

Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Bonial, Claire, Bianca Badarau, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Tim O'Gorman, Martha Palmer, and Nathan Schneider. 2018. Abstract Meaning Representation of constructions: The more we include, the better the representation. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Bonial, Claire, Lucia Donatelli, Mitchell Abrams, Stephanie M. Lukin, Stephen Tratz, Matthew Marge, Ron Artstein, David Traum, and Clare Voss. 2020. Dialogue-AMR: Abstract Meaning Representation for dialogue. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 684–695, Marseille, France. European Language Resources Association.

Bonial, Claire, Lucia Donatelli, Stephanie M. Lukin, Stephen Tratz, Ron Artstein, David Traum, and Clare Voss. 2019. Augmenting Abstract Meaning Representation for human-robot dialogue. In *Proceedings of the First International Workshop on Designing Meaning Representations*, pages 199–210, Florence, Italy. Association for Computational Linguistics.

Bornschein, Jörg and Yoshua Bengio. 2015. Reweighted wake-sleep. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Braune, Fabienne, Daniel Bauer, and Kevin Knight. 2014. Mapping between english strings and reentrant semantic graphs. In *Proceedings of the Ninth International*

Conference on Language Resources and Evaluation, LREC 2014, Reykjavik, Iceland, May 26-31, 2014, pages 4493–4498. European Language Resources Association (ELRA).

Bregman, L.M. 1967. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3):200 – 217.

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

Buys, Jan and Phil Blunsom. 2017. Oxford at semeval-2017 task 9: Neural amr parsing with pointer-augmented attention. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 914–919. Association for Computational Linguistics.

Cai, Deng and Wai Lam. 2020. 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.

Cai, Jiaxun, Shexia He, Zuchao Li, and Hai Zhao. 2018. A full end-to-end semantic role labeler, syntactic-agnostic over syntactic-aware? In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2753–2765, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Cai, Shu 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.

Cao, Jie, Yi Zhang, Adel Youssef, and Vivek Srikumar. 2019. Amazon at MRP 2019: Parsing meaning representations with lexical and phrasal anchoring. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*, pages 138–148, Hong Kong. Association for Computational Linguistics.

Carreras, Xavier and Lluís Màrquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of the Eighth Conference on Com-*

putational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004, pages 89–97, Boston, Massachusetts, USA. Association for Computational Linguistics.

Carreras, Xavier and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 152–164, Ann Arbor, Michigan. Association for Computational Linguistics.

Chen, Tian Qi, Jens Behrmann, David Duvenaud, and Jörn-Henrik Jacobsen. 2019. Residual flows for invertible generative modeling. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 9913–9923.

Cheng, Jianpeng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2019. Learning an executable neural semantic parser. *Computational Linguistics*, 45(1):59–94.

Choe, Hyonsu, Jiyoon Han, Hyejin Park, Tae Hwan Oh, and Hansaem Kim. 2020. Building Korean Abstract Meaning Representation corpus. In *Proceedings of the Second International Workshop on Designing Meaning Representations*, pages 21–29, Barcelona Spain (online). Association for Computational Linguistics.

Clevert, Djork-Arné, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and accurate deep network learning by exponential linear units (elus). In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Cohn, Trevor, Phil Blunsom, and Sharon Goldwater. 2010. Inducing tree-substitution grammars. *Journal of Machine Learning Research*, 11:3053–3096.

Collins, Michael. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 1–8. Association for Computational Linguistics.

Conforti, Michele, Gerard Cornuejols, and Giacomo Zambelli. 2014. *Integer Programming*. Springer Publishing Company, Incorporated.

Corro, Caio and Ivan Titov. 2019. Differentiable perturb-and-parse: Semi-supervised parsing with a structured variational autoencoder. In *International Conference on Learning Representations*.

Cremer, Chris, Xuechen Li, and David Duvenaud. 2018. Inference suboptimality in variational autoencoders. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1078–1086, Stockholmsmässan, Stockholm Sweden. PMLR.

Damonte, Marco 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.

Damonte, Marco, 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.

Dayan, Peter, Geoffrey E. Hinton, Radford M. Neal, and Richard S. Zemel. 1995. The helmholtz machine. *Neural Comput.*, 7(5):889–904.

Devlin, Jacob, 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.

Diamond, Steven and Stephen Boyd. 2016. Cvxpy: A python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5.

Dong, Li 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.

Dowty, David. 1991. Thematic proto-roles and argument selection. *Language*, 67(3):547–619.

Dozat, Timothy and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Dozat, Timothy and Christopher D. Manning. 2018. Simpler but more accurate semantic dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 484–490, Melbourne, Australia. Association for Computational Linguistics.

Drozdzal, Michal, Gabriel Chartrand, Eugene Vorontsov, Mahsa Shakeri, Lisa Di Jorio, An Tang, Adriana Romero, Yoshua Bengio, Chris Pal, and Samuel Kadoury. 2018. Learning normalized inputs for iterative estimation in medical image segmentation. *Medical Image Analysis*, 44:1–13.

Fernandez Astudillo, Ramón, Miguel Ballesteros, Tahira Naseem, Austin Blodgett, and Radu Florian. 2020. Transition-based parsing with stack-transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1001–1007, Online. Association for Computational Linguistics.

FitzGerald, Nicholas, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 960–970, Lisbon, Portugal. Association for Computational Linguistics.

Flanigan, Jeffrey, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. 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. Association for Computational Linguistics.

Flanigan, Jeffrey, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A Discriminative Graph-Based Parser for the Abstract Meaning Representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Baltimore, Maryland. Association for Computational Linguistics.

Foland, William 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.

Fonseca, Erick and André F. T. Martins. 2020. Revisiting higher-order dependency parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8795–8800, Online. Association for Computational Linguistics.

Gal, Yarin and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 1027–1035, USA. Curran Associates Inc.

Gardner, Matt, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

Gershman, Samuel J. and Noah D. Goodman. 2014. Amortized inference in probabilistic reasoning. *Cognitive Science*, 36.

Gildea, Daniel and Daniel Jurafsky. 2000. Automatic labeling of semantic roles. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 512–520, Hong Kong. Association for Computational Linguistics.

Gildea, Daniel, Giorgio Satta, and Xiaochang Peng. 2019. Ordered tree decomposition for HRG rule extraction. *Computational Linguistics*, 45(2):339–379.

Gimpel, Kevin and Noah A. Smith. 2010. Softmax-margin CRFs: Training log-linear models with cost functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 733–736, Los Angeles, California. Association for Computational Linguistics.

Glynn, Peter W. 1990. Likelihood ratio gradient estimation for stochastic systems. *Commun. ACM*, 33(10):75–84.

Goodman, James, Andreas Vlachos, and Jason Naradowsky. 2016. Noise reduction and targeted exploration in imitation learning for Abstract Meaning Representation parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–11, Berlin, Germany. Association for Computational Linguistics.

Grave, Edouard, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the 11th Language Resources and Evaluation Conference*, Miyazaki, Japan. European Language Resource Association.

Groschwitz, Jonas. 2019. *Methods for Taking Semantic Graphs Apart and Putting Them Back Together Again*. Ph.D. thesis, Saarland University and Macquarie University.

Groschwitz, Jonas, Meaghan Fowlie, Mark Johnson, and Alexander Koller. 2017. A constrained graph algebra for semantic parsing with AMRs. In *IWCS 2017 - 12th International Conference on Computational Semantics - Long papers*.

Groschwitz, Jonas, Matthias Lindemann, Meaghan Fowlie, Mark Johnson, and Alexander Koller. 2018. AMR dependency parsing with a typed semantic algebra. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1831–1841, Melbourne, Australia. Association for Computational Linguistics.

Gumbel, E.J. 1954. *Statistical Theory of Extreme Values and Some Practical Applications: A Series of Lectures*. Applied mathematics series. U.S. Government Printing Office.

Guo, Zhijiang 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.

Gupta, Sonal, Rushin Shah, Mrinal Mohit, Anuj Kumar, and Mike Lewis. 2018. Semantic parsing for task oriented dialog using hierarchical representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2787–2792, Brussels, Belgium. Association for Computational Linguistics.

Hajič, Jan, 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).

Hajič, Jan, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, CoNLL '09, pages 1–18, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hardy, Hardy and Andreas Vlachos. 2018. Guided neural language generation for abstractive summarization using Abstract Meaning Representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 768–773, Brussels, Belgium. Association for Computational Linguistics.

Hassan, Hany, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. 2018. Achieving human parity on automatic chinese to english news translation. *CoRR*, abs/1803.05567.

Hazan, Tamir, Subhransu Maji, and Tommi Jaakkola. 2013. On sampling from the gibbs distribution with random maximum a-posteriori perturbations. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

He, K., X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

He, Luheng, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual*

Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 473–483. Association for Computational Linguistics.

He, Luheng, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 643–653, Lisbon, Portugal. Association for Computational Linguistics.

Henderson, James, Paola Merlo, Ivan Titov, and Gabriele Musillo. 2013. Multilingual joint parsing of syntactic and semantic dependencies with a latent variable model. *Computational Linguistics*, 39(4):949–998.

Hinton, G. E., J. L. McClelland, and D. E. Rumelhart. 1986. Distributed representations. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*, page 77–109. MIT Press, Cambridge, MA, USA.

Hoffman, Matthew D., David M. Blei, Chong Wang, and John Paisley. 2013. Stochastic variational inference. *Journal of Machine Learning Research*, 14(4):1303–1347.

Hornik, Kurt. 1991. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257.

Huang, Liang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, pages 586–594, Columbus, Ohio. Association for Computational Linguistics.

Inan, Hakan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Issa, Fuad, Marco Damonte, Shay B. Cohen, Xiaohui Yan, and Yi Chang. 2018. Abstract Meaning Representation for paraphrase detection. 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 442–452, New Orleans, Louisiana. Association for Computational Linguistics.

Ivanova, Angelina, Stephan Oepen, Lilja Øvrelid, and Dan Flickinger. 2012. 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.

Jang, Eric, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Jin, Wengong, Regina Barzilay, and Tommi S. Jaakkola. 2018. Junction tree variational autoencoder for molecular graph generation. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2328–2337. PMLR.

Johansson, Richard and Pierre Nugues. 2008. Dependency-based semantic role labeling of PropBank. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 69–78, Honolulu, Hawaii. Association for Computational Linguistics.

Jones, Bevan, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight. 2012. Semantics-based machine translation with hyperedge replacement grammars. In *Proceedings of COLING 2012*, pages 1359–1376, Mumbai, India. The COLING 2012 Organizing Committee.

Kamp, Hans. 1993. *From discourse to logic : introduction to model theoretic semantics of natural language, formal logic and discourse representation theory*, student edition.. edition. Studies in linguistics and philosophy ; v. 42. Kluwer Academic, Dordrecht ; London.

Kamyshanska, H. and R. Memisevic. 2015. The potential energy of an autoencoder. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(6):1261–1273.

Kapanipathi, Pavan, Ibrahim Abdelaziz, Srinivas Ravishankar, Salim Roukos, Alexander G. Gray, Ramón Fernandez Astudillo, Maria Chang, Cristina Cornelio, Saswati Dana, Achille Fokoue, Dinesh Garg, Alfio Gliozzo, Sairam Gurajada, Hima Karanam, Naweed Khan, Dinesh Khandelwal, Young-Suk Lee, Yunyao Li, Francois P. S. Luus, Ndivhuwo Makondo, Nandana Mihindukulasooriya, Tahira Naseem, Sumit Neelam, Lucian Popa, Revanth Gangi Reddy, Ryan Riegel, Gaetano

Rossiello, Udit Sharma, G. P. Shrivatsa Bhargav, and Mo Yu. 2020. Question answering over knowledge bases by leveraging semantic parsing and neuro-symbolic reasoning. *CoRR*, abs/2012.01707.

Kasper, Robert T. 1989. A flexible interface for linking applications to Penman's sentence generator. In *Speech and Natural Language: Proceedings of a Workshop Held at Philadelphia, Pennsylvania, February 21-23, 1989*.

Kim, Yoon, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured attention networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Kim, Yoon, Sam Wiseman, and Alexander M. Rush. 2018. A tutorial on deep latent variable models of natural language. *ArXiv*, abs/1812.06834.

Kingma, Diederik P. and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kingma, Diederik P. and Max Welling. 2014. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Kingma, Durk P, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. 2016. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, volume 29, pages 4743–4751. Curran Associates, Inc.

Kiperwasser, Eliyahu and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Kipf, Thomas N. and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Kishan, K., Rui Li, F. Cui, Qi Yu, and Anne R. Haake. 2019. Gne: a deep learning framework for gene network inference by aggregating biological information. *BMC Systems Biology*, 13.

Kleijnen, Jack P.C. and Reuven Y. Rubinstein. 1996. Optimization and sensitivity analysis of computer simulation models by the score function method. *European Journal of Operational Research*, 88(3):413 – 427.

Klein, Ayal, Jonathan Mamou, Valentina Pyatkin, Daniela Stepanov, Hangfeng He, Dan Roth, Luke Zettlemoyer, and Ido Dagan. 2020. QANom: Question-answer driven SRL for nominalizations. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3069–3083, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Kollar, Thomas, Danielle Berry, Lauren Stuart, Karolina Owczarzak, Tagyoung Chung, Lambert Mathias, Michael Kayser, Bradford Snow, and Spyros Matsoukas. 2018. The Alexa meaning representation language. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 177–184, New Orleans - Louisiana. Association for Computational Linguistics.

Koller, Alexander and Marco Kuhlmann. 2011. A generalized view on parsing and translation. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 2–13, Dublin, Ireland. Association for Computational Linguistics.

Konstas, Ioannis, 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.

Korte, Bernhard and Jens Vygen. 2018. *Combinatorial Optimization: Theory and Algorithms*, 6th ed. 2018 edition, volume 21 of *Algorithms and Combinatorics*. Springer Berlin / Heidelberg, Berlin, Heidelberg.

Kwiatkowksi, Tom, 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, Cambridge, MA. Association for Computational Linguistics.

Kwiatkowski, Tom, 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.

Lee, Jason, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182, Brussels, Belgium. Association for Computational Linguistics.

Lee, Young-Suk, Ramón Fernandez Astudillo, Tahira Naseem, Revanth Gangi Reddy, Radu Florian, and Salim Roukos. 2020. Pushing the limits of AMR parsing with self-learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3208–3214, Online. Association for Computational Linguistics.

Lewis, Mike, Kenton Lee, and Luke Zettlemoyer. 2016. LSTM CCG parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 221–231, San Diego, California. Association for Computational Linguistics.

Li, Bin, Yuan Wen, Li Song, Weiguang Qu, and Nianwen Xue. 2019a. Building a Chinese AMR bank with concept and relation alignments. In *Linguistic Issues in Language Technology, Volume 18, 2019 - Exploiting Parsed Corpora: Applications in Research, Pedagogy, and Processing*. CSLI Publications.

Li, Yikang, Wanli Ouyang, Bolei Zhou, Jianping Shi, Chao Zhang, and Xiaogang Wang. 2018. Factorizable net: An efficient subgraph-based framework for scene graph generation. In *Computer Vision – ECCV 2018*, volume 11205 of *Lecture Notes in Computer Science*, pages 346–363. Springer International Publishing, Cham.

Li, Zuchao, Shexia He, Hai Zhao, Yiqing Zhang, Zhuosheng Zhang, Xi Zhou, and Xiang Zhou. 2019b. Dependency or span, end-to-end uniform semantic role labeling. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):6730–6737.

Li, Zuchao, Hai Zhao, Rui Wang, and Kevin Parnow. 2020. High-order semantic role labeling. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1134–1151, Online. Association for Computational Linguistics.

Liang, Percy. 2016. Learning executable semantic parsers for natural language understanding. *Commun. ACM*, 59(9):68–76.

Liang, Percy, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 761–768, Sydney, Australia. Association for Computational Linguistics.

Liang, Percy, 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.

Liao, Kexin, Logan Lebanoff, and Fei Liu. 2018. Abstract Meaning Representation for multi-document summarization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1178–1190, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Lindemann, Matthias, Jonas Groschwitz, and Alexander Koller. 2019. Compositional semantic parsing across graphbanks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4576–4585, Florence, Italy. Association for Computational Linguistics.

Lindemann, Matthias, Jonas Groschwitz, and Alexander Koller. 2020. Fast semantic parsing with well-typedness guarantees. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3929–3951, Online. Association for Computational Linguistics.

Liu, Ding and Daniel Gildea. 2010. Semantic role features for machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 716–724, Beijing, China. Coling 2010 Organizing Committee.

Liu, Fei, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. 2015. Toward abstractive summarization using semantic representations. In *Proceedings*

of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1077–1086, Denver, Colorado. Association for Computational Linguistics.

Liu, Jenny, Aviral Kumar, Jimmy Ba, Jamie Kiros, and Kevin Swersky. 2019a. Graph normalizing flows. In *Advances in Neural Information Processing Systems*, volume 32, pages 13578–13588. Curran Associates, Inc.

Liu, Jiangming, Shay B. Cohen, and Mirella Lapata. 2018. 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.

Liu, Jiangming, Shay B. Cohen, and Mirella Lapata. 2019b. 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.

Liu, Yang and Mirella Lapata. 2018. Learning structured text representations. *Transactions of the Association for Computational Linguistics*, 6:63–75.

Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019c. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Loper, Edward and Steven Bird. 2002. NLTK: The Natural Language Toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA. Association for Computational Linguistics.

Luong, Thang, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. 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 11–19, Beijing, China. Association for Computational Linguistics.

Lyu, Chunchuan, Shay B. Cohen, and Ivan Titov. 2019. Semantic role labeling with iterative structure refinement. 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 1071–1082, Hong Kong, China. Association for Computational Linguistics.

Lyu, Chunchuan, Shay B. Cohen, and Ivan Titov. 2020. A differentiable relaxation of graph segmentation and alignment for amr parsing. *ArXiv*, abs/2010.12676.

Lyu, Chunchuan 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.

Maddison, Chris J., Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Marcheggiani, Diego, Jasmijn Bastings, and Ivan Titov. 2018. Exploiting semantics in neural machine translation with graph convolutional networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 486–492, New Orleans, Louisiana. Association for Computational Linguistics.

Marcheggiani, Diego, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 411–420, Vancouver, Canada. Association for Computational Linguistics.

Martins, André, Noah Smith, and Eric Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference*

on Natural Language Processing of the AFNLP, pages 342–350, Suntec, Singapore. Association for Computational Linguistics.

McDonald, Ryan and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy. Association for Computational Linguistics.

Mena, Gonzalo, David Belanger, Scott Linderman, and Jasper Snoek. 2018. Learning latent permutations with gumbel-sinkhorn networks. In *International Conference on Learning Representations*.

Meyers, Adam, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The NomBank project: An interim report. In *Proceedings of the Workshop Frontiers in Corpus Annotation at HLT-NAACL 2004*, pages 24–31, Boston, Massachusetts, USA. Association for Computational Linguistics.

Migueles-Abraira, Noelia, Rodrigo Agerri, and Arantza Diaz de Ilarraza. 2018. Annotating Abstract Meaning Representations for Spanish. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26, pages 3111–3119. Curran Associates, Inc.

Misra, Dipendra Kumar and Yoav Artzi. 2016. Neural shift-reduce CCG semantic parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1775–1786, Austin, Texas. Association for Computational Linguistics.

Mitra, Arindam and Chitta Baral. 2016. Addressing a question answering challenge by combining statistical methods with inductive rule learning and reasoning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2779–2785. AAAI Press.

Mnih, Andriy and Karol Gregor. 2014. Neural variational inference and learning in belief networks. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1791–1799, Bejing, China. PMLR.

Mohamed, Shakir, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. 2020. Monte carlo gradient estimation in machine learning. *Journal of Machine Learning Research*, 21(132):1–62.

Mulcaire, Phoebe, Swabha Swayamdipta, and Noah A. Smith. 2018. Polyglot semantic role labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 667–672, Melbourne, Australia. Association for Computational Linguistics.

Naseem, Tahira, Abhishek Shah, Hui Wan, Radu Florian, Salim Roukos, and Miguel Ballesteros. 2019. Rewarding Smatch: Transition-based AMR parsing with reinforcement learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4586–4592, Florence, Italy. Association for Computational Linguistics.

Neal, Radford M. and Geoffrey E. Hinton. 1999. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, page 355–368, Cambridge, MA, USA. MIT Press.

Nivre, J., J. Hall, J. Nilsson, A. Chanev, G. Eryigit, D. Kübler, S. Marinov, and E.C. Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135. Pagination: 41.

van Noord, Rik and Johan Bos. 2017. Neural Semantic Parsing by Character-based Translation: Experiments with Abstract Meaning Representations. *Computational Linguistics in the Netherlands Journal*, 7:93–108.

Novak, Roman, Michael Auli, and David Grangier. 2017. Iterative refinement for machine translation. *CoRR*, abs/1610.06602.

Oepen, Stephan, Omri Abend, Lasha Abzianidze, Johan Bos, Jan Hajic, Daniel Hershcovich, Bin Li, Tim O'Gorman, Nianwen Xue, and Daniel Zeman. 2020. MRP 2020: The second shared task on cross-framework and cross-lingual meaning representation parsing. In *Proceedings of the CoNLL 2020 Shared Task: Cross-Framework*

Meaning Representation Parsing, pages 1–22, Online. Association for Computational Linguistics.

Oepen, Stephan, Omri Abend, Jan Hajic, Daniel Hershcovich, Marco Kuhlmann, Tim O'Gorman, Nianwen Xue, Jayeol Chun, Milan Straka, and Zdenka Uresova. 2019. MRP 2019: Cross-framework meaning representation parsing. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*, pages 1–27, Hong Kong. Association for Computational Linguistics.

Oepen, Stephan, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, and Zdeňka Urešová. 2015. SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 915–926, Denver, Colorado. Association for Computational Linguistics.

Oepen, Stephan and Jan Tore Lønning. 2006. 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).

O'Gorman, Tim, Michael Regan, Kira Griffitt, Ulf Hermjakob, Kevin Knight, and Martha Palmer. 2018. AMR beyond the sentence: the multi-sentence AMR corpus. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3693–3702, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Palmer, Alexis and Caroline Sporleder. 2010. Evaluating FrameNet-style semantic parsing: the role of coverage gaps in FrameNet. In *Coling 2010: Posters*, pages 928–936, Beijing, China. Coling 2010 Organizing Committee.

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

Palmer, Martha, Daniel Gildea, and Nianwen Xue. 2010. *Semantic Role Labeling*. Morgan & Claypool, Reading, Massachusetts.

Pan, Xiaoman, Taylor Cassidy, Ulf Hermjakob, Heng Ji, and Kevin Knight. 2015. Unsupervised entity linking with Abstract Meaning Representation. In *Proceedings of*

the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1130–1139, Denver, Colorado. Association for Computational Linguistics.

Papandreou, George and Alan L Yuille. 2011. Perturb-and-map random fields: Using discrete optimization to learn and sample from energy models. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 193–200. IEEE.

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Paulus, Max B., Dami Choi, Daniel Tarlow, Andreas Krause, and Chris J. Maddison. 2020. Gradient estimation with stochastic softmax tricks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Peng, Xiaochang, Linfeng Song, and Daniel Gildea. 2015. A synchronous hyperedge replacement grammar based approach for AMR parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 32–41, Beijing, China. Association for Computational Linguistics.

Peng, Xiaochang, Chuan Wang, Daniel Gildea, and Nianwen Xue. 2017. Addressing the Data Sparsity Issue in Neural AMR Parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 366–375. Association for Computational Linguistics.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Peters, Matthew, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. 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.

Peters, Matthew, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018b. 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.

Poon, Hoifung and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1–10, Singapore. Association for Computational Linguistics.

Pourdamghani, Nima, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014a. Aligning English strings with Abstract Meaning Representation graphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 425–429, Doha, Qatar. Association for Computational Linguistics.

Pourdamghani, Nima, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014b. Aligning English strings with Abstract Meaning Representation graphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 425–429, Doha, Qatar. Association for Computational Linguistics.

Pradhan, Sameer S., Eduard Hovy, Mitch Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2007. Ontonotes: A unified relational semantic representation. In *Proceedings of the International Conference on Semantic Computing*, ICSC '07, page 517–526, USA. IEEE Computer Society.

Press, Ofir and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain. Association for Computational Linguistics.

Punyakanok, Vasin, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *American Journal of Computational Linguistics*, 34(2):257–287.

Rezende, Danilo and Shakir Mohamed. 2015. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France. PMLR.

Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286, Bejing, China. PMLR.

Ritchie, Daniel, Paul Horsfall, and Noah D. Goodman. 2016. Deep amortized inference for probabilistic programs. *ArXiv*, abs/1610.05735.

Roark, Brian, Murat Saraclar, Michael Collins, and Mark Johnson. 2004. Discriminative language modeling with conditional random fields and the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL '04, page 47–es, USA. Association for Computational Linguistics.

Romero, Adriana, Michal Drozdzal, Akram Erraqabi, Simon Jégou, and Yoshua Bengio. 2017. Image segmentation by iterative inference from conditional score estimation. *CoRR*, abs/1705.07450.

Roth, Michael and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1192–1202, Berlin, Germany. Association for Computational Linguistics.

Ruppenhofer, Josef, Michael Ellsworth, Miriam R.L. Petruck, Christopher R. Johnson, and Jan Scheffczyk. 2006. *FrameNet II: Extended Theory and Practice*. International Computer Science Institute, Berkeley, California. Distributed with the FrameNet data.

Rush, Alexander. 2020. Torch-struct: Deep structured prediction library. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 335–342, Online. Association for Computational Linguistics.

Rush, Alexander M., David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural*

Language Processing, pages 1–11, Cambridge, MA. Association for Computational Linguistics.

Schlichtkrull, Michael Sejr, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, volume 10843 of *Lecture Notes in Computer Science*, pages 593–607. Springer.

Sennrich, Rico, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nădejde. 2017. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68, Valencia, Spain. Association for Computational Linguistics.

Shi, Chence, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. 2020. Graphaf: a flow-based autoregressive model for molecular graph generation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Shu, Rui, Hung H Bui, Shengjia Zhao, Mykel J Kochenderfer, and Stefano Ermon. 2018. Amortized inference regularization. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Sima'an, Khalil, Rens Bod, Steven Krauwer, and Remko Scha. 1994. Efficient disambiguation by means of stochastic tree substitution grammars. In *International Conference on New Methods in Language Processing, Proceedings of the Conference*, pages 50–58.

Sinkhorn, Richard. 1964. A relationship between arbitrary positive matrices and doubly stochastic matrices. *Ann. Math. Statist.*, 35(2):876–879.

Sobrevilla Cabezudo, Marco Antonio and Thiago Pardo. 2019. Towards a general Abstract Meaning Representation corpus for Brazilian Portuguese. In *Proceedings of the 13th Linguistic Annotation Workshop*, pages 236–244, Florence, Italy. Association for Computational Linguistics.

Song, Li, Yuling Dai, Yihuan Liu, Bin Li, and Weiguang Qu. 2020. Construct a sense-frame aligned predicate lexicon for Chinese AMR corpus. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 2962–2969, Marseille, France. European Language Resources Association.

Song, Linfeng, Daniel Gildea, Yue Zhang, Zhiguo Wang, and Jinsong Su. 2019. Semantic neural machine translation using AMR. *Transactions of the Association for Computational Linguistics*, 7:19–31.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Srivastava, Rupesh Kumar, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2377–2385.

Stanovsky, Gabriel, Julian Michael, Luke Zettlemoyer, and Ido Dagan. 2018. Supervised open information extraction. 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 885–895, New Orleans, Louisiana. Association for Computational Linguistics.

Stark, Chris, B. Breitkreutz, T. Reguly, L. Boucher, A. Breitkreutz, and M. Tyers. 2006. Biogrid: a general repository for interaction datasets. *Nucleic Acids Research*, 34:D535 – D539.

Strassel, Stephanie, Caitlin Christianson, John McCary, William Staderman, and Joseph Olive. 2011. Data acquisition and linguistic resources. In *Handbook of Natural Language Processing and Machine Translation: DARPA Global Autonomous Language Exploitation*, pages 1–131. Springer New York, New York, NY.

Strubell, Emma, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038, Brussels, Belgium. Association for Computational Linguistics.

Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL 2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177, Manchester, England. Coling 2008 Organizing Committee.

Sutskever, Ilya, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27, pages 3104–3112. Curran Associates, Inc.

Szubert, Ida, 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.

Takase, Sho, 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*, pages 1054–1059, Austin, Texas. Association for Computational Linguistics.

Titov, Ivan and Alexandre Klementiev. 2011. A Bayesian model for unsupervised semantic parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1445–1455, Portland, Oregon, USA. Association for Computational Linguistics.

Titsias, Michalis and Miguel Lázaro-Gredilla. 2014. Doubly stochastic variational bayes for non-conjugate inference. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1971–1979, Bejing, China. PMLR.

Tomczak, Jakub M. 2016. On some properties of the low-dimensional gumbel perturbations in the perturb-and-map model. *Statistics & Probability Letters*, 115:8–15.

Toutanova, Kristina, Aria Haghighi, and Christopher D. Manning. 2008. A global joint model for semantic role labeling. *American Journal of Computational Linguistics*, 34(2):161–191.

Tu, Lifu and Kevin Gimpel. 2018. Learning approximate inference networks for structured prediction. In *6th International Conference on Learning Representations,*

ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings.

Tu, Lifu and Kevin Gimpel. 2019. Benchmarking approximate inference methods for neural structured prediction. 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 3313–3324, Minneapolis, Minnesota. Association for Computational Linguistics.

Valiant, L.G. 1979. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189 – 201.

Vincent, Pascal, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, pages 1096–1103.

Wainwright, Martin J. and Michael I. Jordan. 2008. Graphical models, exponential families, and variational inference. *Found. Trends Mach. Learn.*, 1(1–2):1–305.

Wang, Chong, Yining Wang, Po-Sen Huang, Abdelrahman Mohamed, Dengyong Zhou, and Li Deng. 2017. Sequence modeling via segmentations. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3674–3683, International Convention Centre, Sydney, Australia. PMLR.

Wang, Chuan, Sameer Pradhan, Xiaoman Pan, Heng Ji, and Nianwen Xue. 2016. CAMR at SemEval-2016 Task 8: An Extended Transition-based AMR Parser. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1173–1178, San Diego, California. Association for Computational Linguistics.

Wang, Chuan and Nianwen Xue. 2017. Getting the most out of AMR parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1257–1268, Copenhagen, Denmark. Association for Computational Linguistics.

Wang, Chuan, Nianwen Xue, and Sameer Pradhan. 2015a. Boosting transition-based AMR parsing with refined actions and auxiliary analyzers. In *Proceedings of the*

53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pages 857–862, Beijing, China. Association for Computational Linguistics.

Wang, Chuan, Nianwen Xue, and Sameer Pradhan. 2015b. A transition-based algorithm for AMR parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 366–375, Denver, Colorado. Association for Computational Linguistics.

Wang, Xinyu, Jingxian Huang, and Kewei Tu. 2019. Second-order semantic dependency parsing with end-to-end neural networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4618, Florence, Italy. Association for Computational Linguistics.

Watanabe, Yotaro, Masayuki Asahara, and Yuji Matsumoto. 2010. A structured model for joint learning of argument roles and predicate senses. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 98–102, Uppsala, Sweden. Association for Computational Linguistics.

Werling, Keenon, Gabor Angeli, and Christopher D. Manning. 2015. Robust subgraph generation improves Abstract Meaning Representation parsing. 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 982–991, Beijing, China. Association for Computational Linguistics.

Williams, Ronald J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229.

Wolf, Thomas, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Wu, Dekai and Pascale Fung. 2009. Semantic roles for SMT: A hybrid two-pass model. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 13–16, Boulder, Colorado. Association for Computational Linguistics.

Wu, Zhaofeng, Hao Peng, and N. A. Smith. 2020. Infusing finetuning with semantic dependencies. *ArXiv*, abs/2012.05395.

Xia, Yingce, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2017. Deliberation networks: Sequence generation beyond one-pass decoding. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 1782–1792.

Xu, D., Y. Zhu, C. B. Choy, and L. Fei-Fei. 2017. Scene graph generation by iterative message passing. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3097–3106.

Xu, Dongqin, Junhui Li, Muhua Zhu, Min Zhang, and Guodong Zhou. 2020. Improving AMR parsing with sequence-to-sequence pre-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2501–2511, Online. Association for Computational Linguistics.

Xu, Keyulu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5453–5462, Stockholmsmässan, Stockholm Sweden. PMLR.

Xue, Nianwen, Ondřej Bojar, Jan Hajič, Martha Palmer, Zdeňka Urešová, and Xiuhong Zhang. 2014. Not an interlingua, but close: Comparison of English AMRs to Chinese and Czech. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1765–1772, Reykjavik, Iceland. European Language Resources Association (ELRA).

Yu, Lei, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Tomás Kociský. 2017. The neural noisy channel. In *5th International Conference on Learning Represen-*

tations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net.

Yu, Lei, Jan Buys, and Phil Blunsom. 2016. Online Segment to Segment Neural Transduction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1307–1316. Association for Computational Linguistics.

Zareian, Alireza, Svebor Karaman, and Shih-Fu Chang. 2020. Weakly supervised visual semantic parsing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zettlemoyer, Luke S. and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, UAI'05, page 658–666, Arlington, Virginia, USA. AUAI Press.

Zhang, Sheng, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019a. 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.

Zhang, Sheng, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019b. Broad-coverage semantic parsing as transduction. 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 3786–3798, Hong Kong, China. Association for Computational Linguistics.

Zhang, Yue, Wei Jiang, Qingrong Xia, Junjie Cao, Rui Wang, Zhenghua Li, and Min Zhang. 2019c. SUDA-Alibaba at MRP 2019: Graph-based models with BERT. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*, pages 149–157, Hong Kong. Association for Computational Linguistics.

Zhao, Hai, Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009. Multilingual dependency learning: Exploiting rich features for tagging syntactic and semantic dependencies. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 61–66, Boulder, Colorado. Association for Computational Linguistics.

Zhou, Jie and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural 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 1127–1137, Beijing, China. Association for Computational Linguistics.

Appendix A

Bregman's Method

While stochastic softmax formulate the sampling as a standard convex optimization problem, solving them fast is essential. Bregman's method can be highly parallel, which is suitable for modern deep learning applications. We will use Bregman's method to derive an optimization method for sampling our latent structure for AMR parsing in Chapter 5. Here, we describe Bregman's method and show how to derive Sinkhorn's algorithm from it.

Bregman's method solves convex optimization with linear equality,[1] the setting is as follows:

$$\begin{aligned} \mathbf{z}^* = \min_{\mathbf{z}\in\Omega} \; & F(\mathbf{z}) \\ \text{s.t. } & \mathbf{A}\mathbf{z} = \mathbf{b}, \end{aligned} \tag{A.1}$$

where F is strongly convex and continuously differentiable, and Ω is a convex set (for our purpose, it's usually $\mathbb{R}^n_+$). We also have m linear constraints $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{b} \in \mathbb{R}^m$. Two important ingredients are Bregmans' divergence $D_F(\mathbf{z},\mathbf{y}) = F(\mathbf{z}) - F(\mathbf{y}) - \langle \nabla F(\mathbf{y}), \mathbf{z}-\mathbf{y}\rangle$, and Bregman's projection: $P_{\omega,F}(\mathbf{y}) = \arg\min_{\mathbf{z}\in\omega} D_F(\mathbf{z},\mathbf{y})$, where ω represents a constraint. Now, the Bregman's method works as: Intuitively speaking, Bregman's method performs alternating projections with respect to each constraint iteratively. After each projection, the score F is lowered by the construction of Bregman's projection. Such alternating projections eventually converge, and with careful initialization solves the optimization problem:

Theorem 2 ((Bregman, 1967)). *$\mathbf{z}^{(t)}$ in Algorithm 1 has a limit, and $\lim_{t\to\infty} \mathbf{z}^{(t)} = \mathbf{z}^*$ in Equation A.1.*

[1] A slightly more sophisticated version exists to handle inequality (Bregman, 1967).

pick $\mathbf{z}^{(0)} \in \{\mathbf{z} \in \Omega | \nabla F(\mathbf{z}) = \mathbf{u}\mathbf{A}, \mathbf{u} \in \mathbb{R}^m\}$;
for $t \leftarrow 1$ **to** ∞ **do**
  $\mathbf{z}_0^{(t)} \leftarrow \mathbf{z}^{(t-1)}$;
  for $i \leftarrow 1$ **to** m **do**
    $\mathbf{z}_i^{(t)} \leftarrow P_{\mathbf{A}_i\mathbf{z}=\mathbf{b}_i,F}(\mathbf{z}_{i-1}^{(t)})$;
  end
  $\mathbf{z}^{(t)} \leftarrow \mathbf{z}_m^t$;
end

Algorithm 1: Bregman's method for solving convex optimization over linear constraints

A.1 Proof of Theorem 1

First, we show that the correctness of Sinkhorn's algorithm

Proposition 2. *The Sinkhorn's algorithm as defined by Equations 3.15 and 3.16 implements the Bregman's method for solving the optimization problem 3.14.*

Proof. Now, we build Bregman's method for the optimization problem defined by Equation 3.14. We have $\Omega = \mathbb{R}_+^{n\times n}$. For simplicity, we focus on the linear algebraic structure, but do not strictly follow the standard matrix notation. We treat matrix as a vector, but access its items through two indexes.[2] Concretely, we have $\mathbf{z} \in \mathbb{R}_+^{n\times n}$ as our variable, which we treate as a vector in Bregman's method, and $F(\mathbf{z}) = -\langle \lambda + \varepsilon, \mathbf{z}\rangle + \tau\langle \mathbf{z}, \log \mathbf{z} - 1\rangle$[3]. For initialization, we have $\nabla F(\mathbf{z}) = -\lambda + \varepsilon + \tau \log \mathbf{z}$. Take $\mathbf{u} = \mathbf{0}$, we have $\mathbf{z}^{(0)} = \exp(\frac{\lambda+\varepsilon}{\tau}) \iff \log \mathbf{z}^{(0)} = \frac{\lambda+\varepsilon}{\tau}$. This corresponds to initialization step as in our equation 3.15. Then, we iterate through constraints to perform Bregman's projection. First, the column normalization constraints $\forall j, \sum_{i=0}^n \mathbf{z}_{ij} = 1$. So $\forall j$, we need to compute $P_{\sum_i \mathbf{z}_{ij}=1,F}(\mathbf{z}^{(t-1)})$. A very important property is that our $F(\mathbf{z}) = \sum_{ij} f_{ij}(\mathbf{z}_{ij})$, where $f_{ij}(\mathbf{z}_{ij}) = -\lambda_{ij} + \varepsilon_{ij}\mathbf{z}_{ij} + \tau\mathbf{z}_{ij}(\log \mathbf{z}_{ij} - 1)$. Moreover, due to the strong convexity of F, $D_F(\mathbf{z}, \mathbf{y}) = 0 \iff \mathbf{z} = \mathbf{y}$. It's not hard to see that for variables that is not involved in the constraints, they are kept the same before and after

[2] Otherwise, we will have a matrix $\mathbf{A} \in \{0,1\}^{(2n)\times(n\times n)}$, and $\mathbf{b} = \mathbf{1} \in \mathbb{R}^{2n}$.

[3] The regularizor differ from the original one by a constant n, due to the constraints. So, the optimization problem is equivalent.

the projection. Now, let's focus on column j, we have:

$$
\begin{aligned}
&\arg\min_{\mathbf{z}:\sum_i \mathbf{z}_i=1} F(\mathbf{z}) - F(\mathbf{z}_{:,j}^{(t-1)}) - \langle \nabla F(\mathbf{z}_{:,j}^{(t-1)}), \mathbf{z} - \mathbf{z}_{:,j}^{(t-1)} \rangle \\
=&\arg\min_{\mathbf{z}:\sum_i \mathbf{z}_i=1} -\langle \lambda + \varepsilon_{:,j}, \mathbf{z} \rangle + \tau \langle \mathbf{z}, \log \mathbf{z} - 1 \rangle \\
&- \langle \nabla F(\mathbf{z}_{:,j}^{(t-1)}), \mathbf{z} \rangle \\
=&\arg\min_{\mathbf{z}:\sum_i \mathbf{z}_i=1} -\langle \lambda + \varepsilon_{:,j}, \mathbf{z} \rangle + \tau \langle \mathbf{z}, \log \mathbf{z} - 1 \rangle \\
&- \langle -\lambda + \varepsilon_{:,j} + \tau \log \mathbf{z}_{:,j}^{(t-1)}, \mathbf{z} \rangle \\
=&\arg\min_{\mathbf{z}:\sum_i \mathbf{z}_i=1} \tau \langle \mathbf{z}, \log \mathbf{z} - 1 \rangle + \langle \tau \log \mathbf{z}_{:,j}^{(t-1)}, \mathbf{z} \rangle \\
=&\arg\min_{\mathbf{z}:\sum_i \mathbf{z}_i=1} \langle \mathbf{z}, \log \mathbf{z} - 1 \rangle + \langle \log \mathbf{z}_{:,j}^{(t-1)}, \mathbf{z} \rangle \\
=&\mathrm{Softmax}(\log \mathbf{z}_{:,j}^{(t-1)})
\end{aligned}
$$

As iterating over those mutually non-overlapping constraints, the non-focused variables are always kept the same. It is hence equivalent to compute them in parallel, which is expressed in our column normalization operator $\mathcal{T}_c$. Similarly, we can derive row normalization operator $\mathcal{T}_r$. The $\mathcal{T}_c$ and $\mathcal{T}_r$ are sequentially executed because they have overlapping variables in their constraints. Yet, their relative order does not matter. For all the above, Sinkhorn's algorithm is an implementation of Bregman's method. □

Now the Theorem 1 is simply a corollary.

Proof of Theorem 1. By Proposition 2, our algorithm is an implementation of Bregman's method, therefore Theorem 1 follows from Theorem 2. □

Appendix B

Appendix of AMR Parsing with Latent Alignment

B.1 Matching Algorithm for Copying Concepts

We present the algorithm for building copying function (i.e., dictionary from words to concepts).

Input : $\{\mathbf{w}^l, \mathbf{c}^l\}_{l=1}^{L}$
Output: D copy dictionary
Counter $\leftarrow \emptyset$
for $l = 1$ *to* L **do**
  for *all pairs* c_i^l *and* w_j^l **do**
    if *match*(c_i^l, w_j^l) **then**
      Increment Counter[w_j^l][c_i^l]
    end
  end
end
D $\leftarrow$ default Stanford lemmatizer
for $w \leftarrow$ *Counter* **do**
  D[w] $\leftarrow$ argmax$_c$ Counter[w][c]
end
return D

Algorithm 2: Copy function construction

Only frequent concepts c (frequency at least 5) can be generated without the copying mechanism (i.e., have their own vector $\mathbf{v}_c$ associated with them). Both frequent and

infrequent ones are processed with coping, using candidates produced by the algorithm below and the matching rule in Table B.1.

Rules	Matching Criteria
Verbalization Match	exact match frame in ”verbalization-list-v1.06.txt”
PropBank Match	exact match frame in PropBank frame files
Suffix Removal Match	word with suffix (“-ed”, “-ly”,“-ing”)
Edit-distance Match	edit distance smaller than 50% of the length

Table B.1: Matching rules for Algorithm 1

B.2 Hyper-parameters

Optimizer Parameters	Values
Batch size for single stage	64
Maximum Epochs	30
Batch size for first stage	512
Batch size for second stage	64
Maximum Epochs for both stages	30
Learning Rate	1e-4
Adam betas	(0.9, 0.999)
Adam eps	1e-8
Weight decay	1e-5

Table B.2: Optimization parameters for full joint training and two stages training.

Model components	Hyper-parameters
Glove Embeddings	300
Lemma Embeddings	200
POS Embeddings	32
NER Embeddings	16
Category Embeddings	32
Concept/Alignment	1 layer 548 input
Sentence BiLSTM	256 hidden (each direction)
AMR Categories $\mathcal{T}$	32
AMR Lemmas $\mathcal{C}$	506
AMR NER types	109
Alignment	1 layer 232 input
AMR BiLSTM	100 hidden (each direction)
B bilinear align	200×512
Relation map dimensionality d_g	200
Relation/Root	2 layers 549 input (predicate position)
Sentence BiLSTM	256 hidden (each direction)
d_f relation vector	200
v_c, v_{copy} lemma vector	512
v_{root} root vector	200
Sinkhorn temperature	1
Sinkhorn prior temperature	5
Sinkhorn steps l for full joint training	10
Sinkhorn steps l for two stages training	5
λ	10
Dropout	.2

Table B.3: Model hyper-parameters

Appendix C

Appendix of AMR Parsing with Latent Alignment and Segmentation

C.1 Greedy Segmentation

Input: graph $\mathbf{G}$, node index i

Result: segmentation $\mathbf{S}$, n, z, k

$\mathbf{S} = \mathbf{0}$, $k = i$, $n = 1$, $z = \mathbf{z}_i$;
forall $j \in \text{Child}[i]$ **do**
  if j notvisited **then**
    $\mathbf{S}', n', z', k' = \text{Greedy}(\mathbf{G}, j)$;
    $\mathbf{S} = \mathbf{S} + \mathbf{S}'$;
    if $n + n' \leq T \wedge z' + z \leq 1$ **then**
      $\mathbf{S}_{kj} = 1$, $n = n + n'$, $z = z + z'$ $k = k'$;
    end
  end
end

Algorithm 3: Greedy Segmentation

Variable $\mathbf{z}_i$ indicates whether node i is copy-able and $T = 4$ represent the maximum subgraph size; n denotes the current subgraph size; z indicates whether the current subgraph contains a copy-able node; k is the last node in the current subgraph, which is used to generate to future nodes in a subgraph. The condition $n + n' \leq T \wedge z' + z \leq 1$ determines whether we combine the current subgraph rooted at node i and the subgraph rooted at node j. Running the algorithm on an AMR graph and the root index will get

us the entire segmentation.

C.2 Hyper-Parameters

We use RoBERTa-large (Liu et al., 2019c) from Wolf et al. (2020) as contextualised embedding before LSTMs. BiLSTM for concept identification has 1 layer, and BiLSTM for relation identification has 2 layers. Both have hidden size 1024. Their averaged representation is used for alignment. RelGCN used 128 hidden units and 1 hidden layer (plus one input layer and output layer). Relation identification used 128 hidden units. The LSTM for the locally auto-regressive model is one layer with 1024 hidden units. Adam (Kingma and Ba, 2015) is used with learning rate $3e-4$ and beta=$0.9, 0.99$. Early stopping is used with a maximum of 60 training epochs. Dropout is set at 0.33. Those hyper-parameters are selected manually. We basically followed the setting as in the Section B.2.

C.3 Proof of Proposition 1

The proof of Proposition 1 is essentially identical to the proof of Theorem 1 in Appendix A.1. Therefore, we skip the proof.

C.4 Generation Order is Discrete by LP

If $\mathbf{O}(\widetilde{\mathbf{W}}, 0)$ is integral valued, it belongs to O by definition. In most cases, there is no guarantee that the linear programming in the relaxed space yields a solution that is also an integer. However, in our cases, we have the following result:

Proposition 3. *With probability 1, a unique* $\mathbf{O}(\widetilde{\mathbf{W}}, 0) \in \{0,1\}^{(n+m)\times(m+1)}$*, where* $\mathbf{O}(\widetilde{\mathbf{W}}, 0)$ *is defined in Equation 5.16.*

Intuitively, this is a generalization of the classical result about perfect matching on bipartite graph (Conforti et al., 2014). To prove this, we need the following theorems from integer linear programming.

Theorem 3 ((Conforti et al., 2014, page 130,133))**.** *l Let A be an* $q \times p$ *integral matrix. For all integral vectors* d, l, u *and* $c \in \mathbb{R}^p$*,* $\max\{\langle c, x\rangle : Ax = d, l \leq x \leq u\}$ *is attained*

by an integral vector x if and only if A is totally unimodular.[1]

Note that this theorem does not say all the solution is integer, nor it's unique. However, one should understand this limitation as some degenerate case of c. However, a total unimodular matrix does characterize the convex hull of its integral points. To prove this, we need an additional lemma.

Lemma 1 ((Conforti et al., 2014, page 21)). *l Let $S \in \mathbb{R}^n$ and $c \in \mathbb{R}^n$. Then* $\sup\{\langle c,s\rangle : s \in S\} = \sup\{\langle c,s\rangle : s \in \text{Conv}(S)\}$. *Furthermore, the supremum of $\langle c,s\rangle$ is attained over S if and only if it is attained over* $\text{Conv}(S)$.

where $\text{Conv}(S)$ is the convex hull of S. Now we have the following proposition:

Proposition 4. *Let A be an $q \times p$ integral matrix. For all integral vectors d,l,u ,and $c \in \mathbb{R}^p$ such that $\{x \in \{0,1\}^p | Ax = d, l \le x \le u\}$ is a finite set, $\{x \in \text{|} Ax = d, l \le x \le u\} = \text{Conv}(\{x \in \{0,1\}^p | Ax = d, l \le x \le u\})$ if and only if A is totally unimodular.*

In other words, we know the LP relaxation is the convex hull when A is totally unimodular. [2]

Proof. By theorem 3, A is totally unimodular is equivalent to maximum is attained by an integer solution. Clearly, the LP relaxation contains the convex hull. So, we only need to show that the LP relaxation does not contain any more points. Now suppose the LP relaxation contains another point x' that's not in the convex hull. Since, we restrict our discussion on finite set of integer, both the $\{x'\}$ and the convex hull is closed set. Then by separation theorem, we have a vector c s.t. $\langle c,x'\rangle > \langle c,x\rangle \forall x \in \text{Conv}(\{x \in \{0,1\}^p | Ax = d, l \le x \le u\})$, which contradicts to lemma 1. □

Theorem 4 ((Conforti et al., 2014, page 133,134)). *A $0,\pm 1$ matrix A with at most two nonzero elements in each column is totally unimodular if and only if rows of A can be partitioned into two sets, red and blue, such that the sum of the red rows minus the sum of the blue rows is a vector whose entries are $0,\pm 1$ (admits row-bicoloring).*

Our $\mathbf{O}$ should be the column vector x, and constraints should be represented by a matrix A. In particular, we view $\mathbf{O}$ as a column vector, but still access the item by $\mathbf{O}_{ij}$.[3] The matrix $A \in \{0,\pm 1\}^{(m+(m+n))\times((n+m)(m+1))}$. $A_{:,ij}$ denotes the constraints involving

[1] A is totally unimodular if every square submatrix has determinant $0,\pm 1$. We combined a few theorems and definitions from Conforti et al. (2014) into this theorem.

[2] Proposition 4 should be a common knowledge, but we failed to find a direct reference.

[3] Alternatively, one could have a vector x and $x_{i(m+1)+j} = \mathbf{O}_{ij}$. However, this will gets clumsy.

$\mathbf{O}_{ij}$. The first m rows of A correspond to $\forall j < m, \sum_{i=0}^{n+m-1} \mathbf{O}_{ij} = 1$, and the remaining $m+n$ rows correspond to $\forall i, \sum_{j=0}^{m} \mathbf{O}_{ij} = 1$. Therefore, we have $\forall k < m, j < m, i, A_{k,ij} = \delta_{j,k}$ and $\forall k \geq m, j, i A_{k,ij} = \delta_{i,k-m}$, else $A_{k,ij} = 0$, where $\delta_{j,k} = [[j == k]]$. We have the linear constraints in standard form as $A\mathbf{O} = \mathbf{1}$.

Lemma 2. *The A defined above is totally unimodular.*

Proof. First, we show A admits row-bicoloring. We color the first m rows red, and remianing $n+m$ rows blue. The sum of red rows is: $R_{ij} = \sum_{k=0}^{m-1} A_{k,ij} = \sum_{k=0}^{m-1} \delta_{j,k} = [[j < m]]$ and the sum of blues is $B_{ij} = \sum_{k=m}^{2m+n-1} A_{k,ij} = \sum_{k=m}^{2m+n-1} \delta_{i,k-m} = 1$. Therefore, $R_{ij} - B_{ij} = [[j == m]] \in \{0, \pm 1\}$, and A admits a row-bicoloring. Since A has only $0, \pm 1$ value, and one variable in $\mathbf{O}$ at most participates in two constrains (in-coming and out-going), by theorem 4, A is totally unimodular. □

Now, we prove proposition 3.

Proof. We have A being totally unimodular. Take $c = \widetilde{W}$, $l = 0, u = 1$. By theorem 3, the LP solutions contain an integer vector. Since the Gumbel distribution has a positive and differentiable density, by (Paulus et al., 2020, Proposition 3), $\arg\max_{\mathbf{O} \in O} \langle \widetilde{\mathbf{W}}, \mathbf{O} \rangle$ yields a unique solution with probability 1. Clearly, this solution is the only integer solution in our LP solutions. Now, suppose another non-integer solution exist. We know the linear programming domain is the convex hull by proposition 4. Clearly, another integer solution exist, which contradicts the uniqueness of integer solution. Hence, the $\mathbf{O}(\widetilde{\mathbf{W}}, 0)$ yields a unique integer solution with probability 1. □

Appendix D

Appendix of Semantic Role Labeling with Iterative Structure Refinement

D.1 Hyper-Parameters

Hyper-parameter	Value		Description
	English	Others	
p	0.3		dropout rate for all neural modules
p_r	0.3		recurrent dropout rate for BiLSTMs
d_w	1024	300	tokens embedding dimension
d_δ	64		dependency label embedding dimension
d_p	64		part-of-speech tags embedding dimension
d_h	500	428	BiLSTM hidden state dimension in one direction
d_{ρ_0}	300		dimension for feature for null role logits
d_{ρ_1}	128		dimension for feature for other role logits
d_g	200		dimension for feature for refinement networks
d_r	200		hidden dimension of refinement networks
λ_g^π	50		multiplier of Gumbel noise for sense logits
λ_g^α	5		multiplier of Gumbel noise for role logits

Table D.1: Hyper-parameters value and description. Note that the input and output dimensions of MLP and BiLSTM can be decided by the other hyper-parameters at each occurrence.