A Study on Abstract Meaning Representation

VU Trong Sinh

Japan Advanced Institute of Science and Technology

Doctoral Dissertation

A Study on Abstract Meaning Representation

VU Trong Sinh

Supervisor : Professor NGUYEN Le Minh

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
Information Science
June, 2020

Abstract

Humans are born with the ability to communicate with their natural language. Computing machines, on the other hand, only understand several specific programming languages, with a limit of expressions. To bridge the gap between human languages and computer languages, semantic representation is one such a solution, with the ability to convert natural language utterances into machine-understandable forms. Many semantic schemes have been introduced and developed, such as Combinatory Categorial Grammar (CCG), Groningen Meaning Bank (GMB) or Abstract Meaning Representation. Two traditional problems of semantic representations are producing them from natural language (parsing) as well as producing natural language from them (generation). In this thesis, we present our study in Abstract Meaning Representation (AMR) Parsing and Generation, which are showing lots of potential in the computational linguistics community recently. We also present our first attempt on the domain adaptation in parsing and generation for legal text.

In the first part of our thesis, we present our AMR-to-text generator incorporating the self-attention mechanism. Motivated by the domination of the Transformer architecture in various Natural Language Processing tasks, e.g. machine translation, text summarization, we adopt its core component - the self-attention - to build our generation models. We conduct experiments on both sequence to sequence and graph to sequence strategies, which are dominating in solving this problem. Our proposed method obtains competitive results on a benchmark AMR dataset, with an improvement of 3.2 BLEU score over the baseline sequence-to-sequence model.

Despite several developments in AMR parsing and generation for text in the general domain, current methods for these tasks still struggle in dealing with the legal domain. The legal text is often structurally complicated, consists of longer sentences and contains specific terminologies that are rarely seen in general-domain text. This also causes lots of difficulties in natural language understanding in general, and AMR parsing in our study. In the second part of our thesis, we provide a literature survey over different methods in AMR parsing and show their performances on analyzing legal documents. We conduct empirical experiments of various AMR parsers on a benchmark AMR dataset with various ranges of sentence length, and our annotated legal dataset. Our results show the current limitations and also open a room for improvements of current parsing techniques for legal domain adaptation.

For the generation direction, we observe that text generated from AMR using current deep learning models usually become awkward with lots of "out of vocabulary" tokens. In the second part of our thesis, we propose some modifications in the training and decoding phase of the encoder-decoder AMR generation model to have a better text realization. Our model is tested using an annotated legal dataset extracted from the English version of the Japanese Civil Code, showing an improvement compared to the baseline model.

To summarize, our study in AMR parsing and generation along with the legal domain adaptation contributes to the literature of semantic representation. Despite some improvements and findings, our work still remains specific drawbacks. Since our first results are still preliminary, we figure out several ideas to improve our performance in the future.

Keywords: **Abstract Meaning Representation**, **Deep Learning**, **Semantic Parsing**, **Legal Domain**, **Text Generation**.

Acknowledgment

The small contributions of this thesis to the literature remind that this study cannot be completed without the support of many people.

First, I would like to deeply thank my supervisor, Professor Nguyen Le Minh, for his support and motivation. He gave me a lot of valuable comments, advice, and discussion, which guide me to approach my research problem. He always encourages me with great enthusiasm and encouragement not only in my research but also in many other aspects of the academic career. I still remember one time, in a late evening when I sent my manuscript for him to review before submitting to a conference, he spent nearly the whole night to help me edit and finish the manuscript until the conference deadline. Without his guidance and inspiration, I would have never finished my work.

I appreciate useful comments from committee members: my second supervisor Professor Satoshi Tojo, my sub-theme advisor Associate Professor Kiyoaki Shirai, Associate Professor Shinobu Hasegawa, and Associate Professor Ashwin Ittoo. Through the discussion, they pointed out the limited points of my research in several aspects and provided suggestions for improving my thesis.

I must thank my collaborators in Nguyen's Laboratory for their useful discussions during weekly seminars. I enjoyed fascinating discussions with their clever minds, which inspire several interesting ideas for my research. I love the friendly working atmosphere in our laboratory, in that members can freely discuss research questions and assist together to find out appropriate solutions. I learned many things from my collaborators, especially the former students Dr. Nguyen Minh Tien, Dr. Phan Viet Anh, Dr. Tran Van Khanh and also my tutor Dr. Nguyen Tien Huy.

I want to thank all members of the Vietnamese Community. Thanks to them, I can adapt my lifestyle quickly when I first come to Japan. Another special thank to our Vietnamese Soccer Club, by playing soccer every week, I could maintain my attitude and physical health to avoid stress in doing my research.

Last but not least, I would not be who I am today without the love, encouragement, and unconditional support from my family. Especially, I would like to express the deepest appreciation to my wife for her love and inspiration during my Ph.D. time. She brings a little angel, Suki-chan, to my life, takes care of her and always be on my side during the difficult years.

There are also many other people I have to thank, but it is difficult to list all their names. I will keep all of you in my mind with my best regards.

The work in this dissertation was financially supported by the Vietnamese Government Scholarship (Project 911) and the Japanese JST CREST Grant Number JPMJCR1513. By working in the CREST project, I also have the chance to experience intensive research in various areas, which broadens my viewpoint a lot.

Vu Trong Sinh

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research Direction

Who did what to whom, where, when and how? These types of questions do not cause any difficulties for human to answer in a given context, but they are very challenging for computers to directly understand. For more than twenty years, intelligent language processing heavily based on syntactic treebanks, e.g. Penn Treebank, Prague Dependency Treebank. The syntactic information exploited from these treebank helps natural language understanding systems analyze the grammar inside texts. However, the lack of semantic understanding still causes lots of ambiguities. This leads to a trend of moving from the analysis of the grammatical structure to sentence semantics recently. Many semantic representations (SR) have been proposed, such as Universal Conceptual Cognitive Annotation (UCCA) [3], Groningen Meaning Bank (GMB) [4], Abstract Meaning Representation (AMR) [5]. In which, the AMR has shown great impact and potential, gained lots of attention in the computational linguistics community. Since first introduced in 2013 by Banarescu et al. [5], the number of papers about AMR in top NLP conferences increases year by year. Figure 1.1.1 shows a detailed statistic about this statement. One can easily find an AMR paper in ACL, EMNLP or NAACL at this time. This confirms the interest of NLP researchers for this semantic language.

Research about AMR can be categorized into various types:

- *Parsing*: analyzing a human language string and mapping it to an AMR graph [6][7][8][9][10][11],
- *Generation*: generating a human language string from a source AMR graph [12][13][1][14][15][16][17][18][19],
- *Evaluation*: designing metrics to evaluate the result of parsing or generation task [20][21][22],
- *Multilingual*: developing the Propbank frameset and annotating AMR corpus in different languages [23][24][25][26][27][28][29],
- *Application*: using AMR to solve other tasks in natural language processing [30][31][32][33] [34][35] [36] [37] [38].

AMR can be applied as an intermediate meaning representation beyond human text, help solve various NLP tasks, such as machine comprehension [38], machine translation

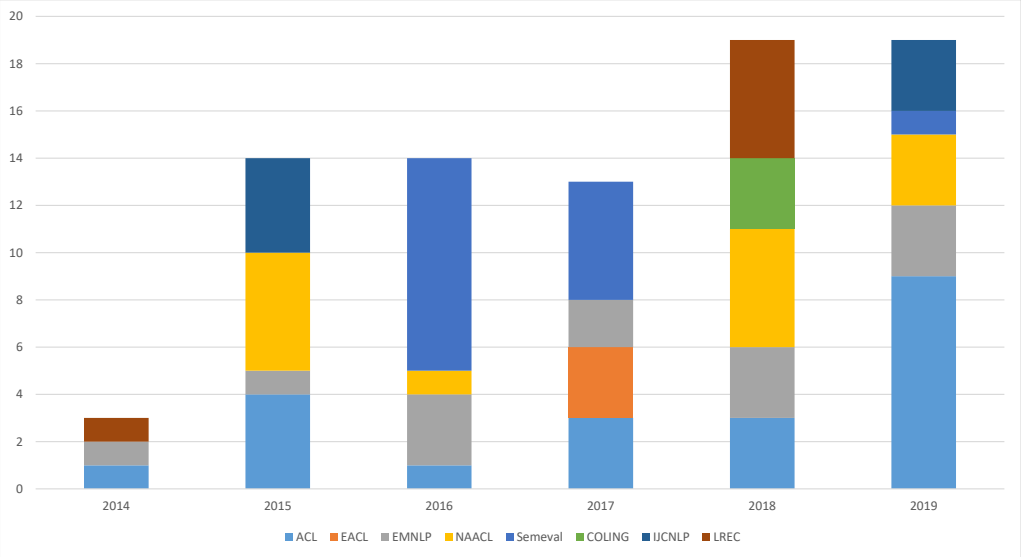


Figure 1.1.1: Number of papers about AMR in top NLP conferences through years

[39][30], text summarization [31][34][35][36], question answering [37], event extraction [32]. Figure 1.1.2 illustrates some typical ideas in applying AMR for summarization and machine translation.

For AMR to be applied in those tasks, the problem of AMR Parsing and AMR-to-text Generation are both important. In this thesis, we focus on these two tasks, aiming to improve the performance of parsing and generation by using deep learning techniques. We describe an example of AMR Parsing and Generation in Figure 1.1.3.

We also study the challenge of AMR adapting to the legal domain, where the text is logically complicated with lots of long sentences and domain-specific terms. We use English as our main language, due to the completeness in the Propbank dictionary as well as the availability of a huge amount of training data.

1.1.1 AMR Generation with the Self-Attention Mechanism

One of the most successful approaches in the AMR-to-text Generation problem relies on the encoder-decoder architecture. Previous works considered this problem as a translation problem and applied sequence-to-sequence models to ”translate” a linearized AMR string into human language. Later, graph-to-sequence models were proposed to reduce the information loss during the linearization process and achieve significant improvement in the quality of AMR generation. In our work, we attempt to improve this encoder-decoder approach by incorporating a self-attention mechanism. This work is motivated by the success of the transformer network [40], in which the self-attention (or multi-head attention) plays a crucial role, in various NLP tasks. On a standard benchmark dataset, our incorporating method obtains comparative results comparing to existing neural methods in the literature.

1.1.2 Legal Domain Adaptation

As mentioned in section 1.1, the legal text is often structurally complicated, and contain specific terminologies that are rarely seen in general-domain text. This causes lots of

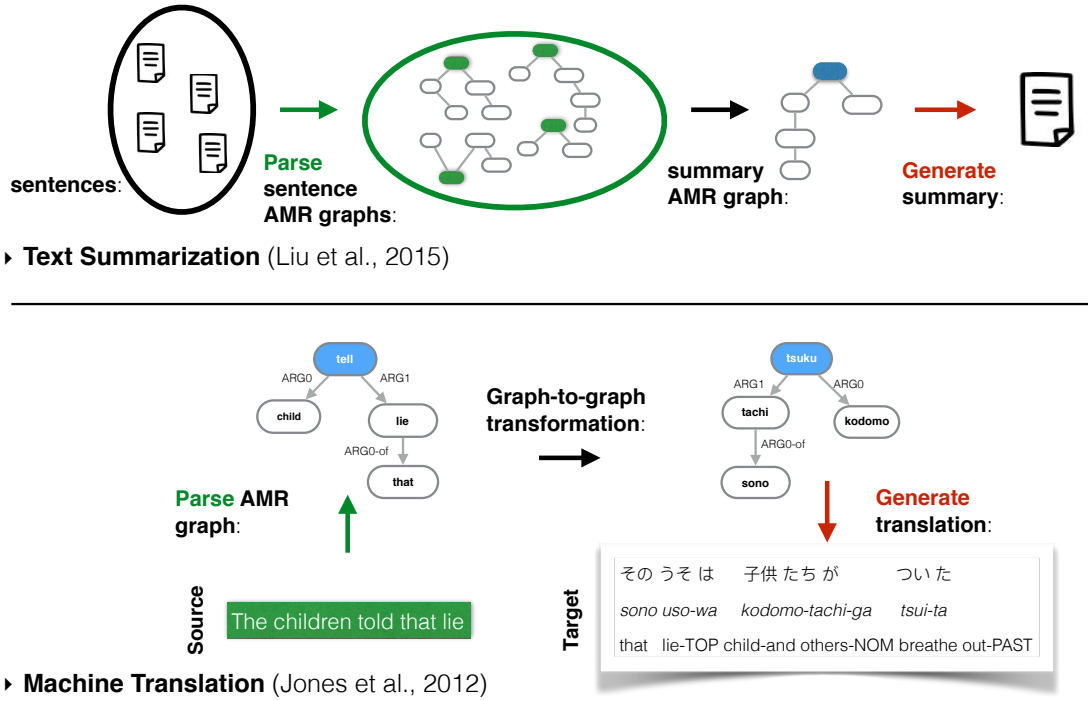


Figure 1.1.2: Applications of AMR in Text Summarization and Machine Translation

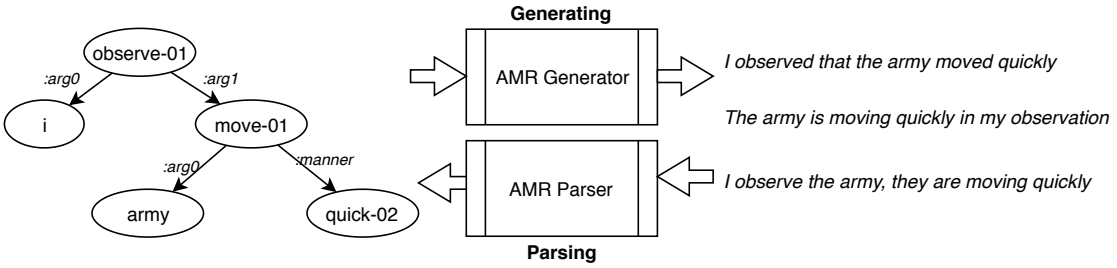


Figure 1.1.3: An example of AMR Parsing and Generation

difficulties in both natural language understanding as well as generation. In this thesis, we study the challenges of both AMR parsing and generation in the legal domain.

Specifically, for the parsing direction, we provide a literature survey over different methods in AMR parsing, which we category in three groups, and show their performances on analyzing legal documents. We conduct empirical experiments of various AMR parsers on a benchmark AMR dataset with various ranges of sentence length, and our annotated legal dataset. Our results show the challenges and also suggest room for improvements of current parsing techniques for legal domain adaptation.

For AMR generation, we observe that text generated from AMR using current deep learning models usually become awkward with lots of ”out of vocabulary” tokens. We propose some modifications in the training and decoding phase of the encoder-decoder AMR generation model to have a better text realization. Our model is tested using the legal dataset above, showing an improvement compared to the baseline model.

1.2 Dissertation contributions

The main contributions of this dissertation can be summarized as follow:

- **Legal Dataset JCivilCode** : We extract the first four chapters in the Japanese civil code (in the English version) and manually annotate in the format of AMR. This is the first AMR dataset in the legal domain, rather than popular datasets mainly taken from news, blog posts. Though the number of samples is still small, this dataset helps develop the research in domain adaptation in the legal domain. We conduct experiments of different AMR parsers in three main parsing approaches on our annotated dataset to see the quality of legal text parsing. Our results show the difficulties as well as suggest several ideas for future improvement in AMR parsing for legal documents.
- **Self-attention text generation from AMR graphs**: We propose a transformer approach in converting an AMR graph into a natural language sentence. We incorporate the self-attention mechanism into the encoder-decoder model in both sequence to sequence and graph to sequence strategies. Evaluating by a benchmark dataset, our method obtains comparative results comparing to existing neural models in the literature.
- **Legal Style Text Generation**: We propose two modifications in the training and decoding phase of the neural graph to sequence AMR generation model. With these modifications, we provide more constraints to tackle the problem of generation from legal AMR. Our model is tested using JCivilCode dataset, showing an improvement compared to the baseline model.

1.3 Dissertation Outline

We have introduced our research direction and presented the abstract of our work in this Chapter. In the following chapters, we provide some necessary background knowledge, then dwell on the details of the implementations of our methods. The detail is as follow:

- In Chapter 2, we present an overview of the AMR with the definition, examples, the datasets used in our thesis and the evaluation metrics. We also briefly describe the encoder-decoder architecture with various types of attention mechanism in this chapter.
- In Chapter 3, we present our approach for incorporating the self-attention mechanism to an AMR generation model.
- In Chapter 4, we report our empirical evaluation of different AMR parsing methods when applying to legal text.
- In Chapter 5, we present our modifications to the training phase and decoding phase to generate legal text from AMR.

Finally, Chapter 6 concludes with a summary of our findings throughout the dissertation and the potential future directions of our research. Figure 1.3.1 summarizes this thesis outline.

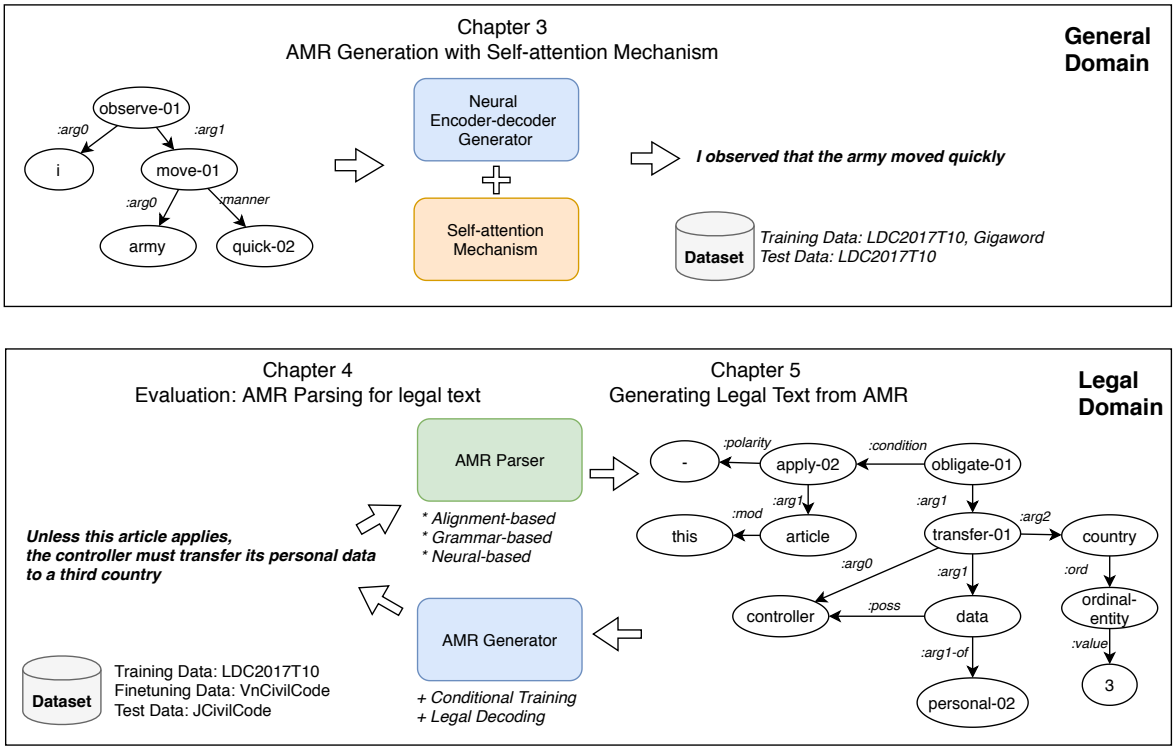


Figure 1.3.1: Dissertation Outline

Chapter 2

Preliminary

In this chapter, we present an overview of the AMR with the definition, examples of annotation in different formats. We introduce the datasets used in our thesis and the evaluation metrics for both parsing and generation. Later, we briefly describe the encoder-decoder architecture, with different types of attention that is originally used in machine translation and adapt to AMR problems recently.

2.1 Abstract Meaning Representation (AMR)

The Abstract Meaning Representation (AMR) is a whole-sentence semantic representation with the ability of capturing relational semantics in that sentence, or "who-is-doing-what-to-whom" in a graph structure. Among various semantic representations being proposed [3][4], the major innovation of AMR is that it has been designed for rapid large-scale annotation. It also has been used to annotate a large corpus of natural language sentences, creating a semantics bank or SemBank [5]. This sizable sembank is believed to lead to new works in natural language understanding (NLU), resulting in semantic parsers that are as ubiquitous as syntactic ones, and support natural language generation (NLG) by providing a logical semantic input [5].

AMR encodes the meaning of a sentence as a rooted, directed, edge-labeled, leaf-labeled graph. Typically, the main verb of the source sentence will be chosen as the root of the graph. Since each verb has various meaning and can be used in different contexts, a sense ID will be assigned. Then every vertex and edge of the graph are labeled according to this sense. For instance, consider the verb *"look"* in two sentences: *"He is looking at his car"*, and *"It looks like a gun"*. In the first sentence, *"look"* is used to express the vision, so it will be labeled as *look-01*, with the looker (*ARG0*): *"He"* and the thing is being looked at (*"ARG1"*): *"his car"*. But in the second sentence, *"look"* means something seems/appears to be something (*"look-02"*), thus *"It"* and *"like a gun"* will be labeled as *"ARG0"* and *"ARG1"*, respectively.

There are three ways to represent an AMR graph:

- PENMAN notation: or more accurately "Sentence Plan Language" [41], is used to encode arbitrary graphs in the text format. This format is easy to read and write by human, thus convenient to annotators to create their AMR datasets.
- Graph structure: suitable for a computer to store in the memory, e.g. for calculating the neighbors, descendants, siblings, etc.

- Logical triples: suitable for measuring the difference among AMRs, which we will illustrate the evaluation process in the later part of this dissertation.

Figure 2.1.1 illustrates an example of AMR annotation for the sentence *"He is looking at the car"* with the three formats mentioned above. AMR abstracts away many surface form of words in a sentence, e.g. the verb tense, quantity, etc. It also removes words that do not contribute to the core meaning, e.g. the article, preposition.

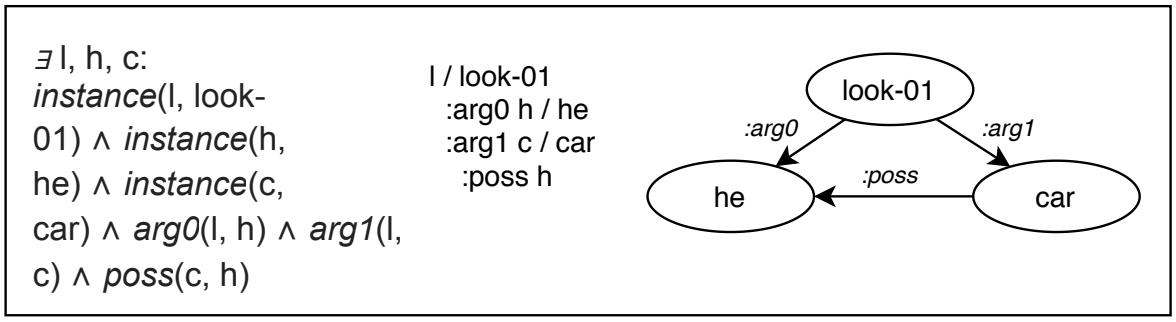


Figure 2.1.1: AMR in three formats: logical triples, PENMAN notation and graph structure (from left to right)

In AMRs, each graph node is named by a variable, specified by a unique ID. This variable represents the semantic concept, which can be a single word (e.g. *he*), a PropBank verb sense (e.g. *look-01*) or a special keyword. The keywords in AMR include: entity types (e.g. *ordinal-entity, percentage-entity, date-entity*), quantities (e.g. *distance-quantity, money-quantity*), and logical operation (e.g. *and, or*). The edge between two nodes is labeled by a relation types. According to Propbank dictionary, there are more than 100 relation types, consisting of:

- Frameset argument, from *":arg0"* to *":arg5"*, following PropBank conventions
- Semantic relations, e.g. *":purpose", ":manner", ":instrument", ":time"*
- Quantity relations, e.g. *":quant", ":unit"*
- Date entity relation, e.g. *":day", ":month", ":year", ":time"*
- Listing relation, from *":op1"* to *":op10"*.

AMR also provides the inverse form of all relations by converting a relation R to *R-of* (e.g. :ARG1 vs :ARG1-of, :purpose vs :purpose-of). Hence, if R is a directed relation of two entities e_1 and e_2, we have $R(e_1, e_2) \equiv R - of(e_2, e_1)$.

Currently, the AMR version does not support inflectional morphology for tense, number and articles. It also has no representation for quantifier, e.g. *"all"*. The purpose of this exclusion is to speed up the annotation process and the authors state that the syntactic representation could be embedded inside via an automatic post-process.

It is necessary to emphasize that AMRs are semantic representation. Unlike syntax, which can be represented as trees (e.g. dependency tree, combinatory categorial grammar), semantics is best represented as a graph structure since an entity can have multiple relations with more than one other entity in the sentence. This motivates the development of new algorithms for handling semantic graphs, e.g. graph neural network, graph to sequence learning which we also based on in our research.

2.2 AMR Datasets

In this section, we provide the datasets information for both AMR Parsing and Generation that we use in our experiments. Since introduced in 2013 [5], several AMR datasets have been annotated and published, e.g. *"The Little Prince"* (2016), *"BioAMR"* (2016), *"LDC AMR Corpora"* (2014, 2015, 2016, 2017). In which, *"LDC AMR Corpora"* are considered the benchmark datasets for AMR related tasks, due to their large amount of samples and also the quality of annotation. We also introduce our legal datasets JCivilCode and VNCivilCode in this section.

2.2.1 Benchmark Dataset

Following many other works in the literature, we use the benchmark dataset developed by the Linguistic Data Consortium (LDC), the University of Colorado's Computational Language and Educational Research group, SDL/Language Weaver, Inc., and the Information Sciences Institute at the University of Southern California. The version we use in our research was released in 2017, namely LDC2017T10, or AMR 2.0 [1]. It contains a semantic treebank (or sembank for short) of nearly 40,000 English natural language sentences from the source of newswire, discussion forum and other web logs, television transcripts. The latest version of this dataset includes 59,255 samples in total. LDC also introduced AMR 1.0 (LDC2014T12), which was common used by previous research in AMR.

The data is collected from various sources, including discussion forums in the DARPA BOLT and DEFT programs, transcripts; English translations of Mandarin Chinese broadcast news programming from China Central TV; Wall Street Journal text, Xinhua news texts (also in English translation); various newswire data from NIST OpenMT evaluations and weblog text used in the DARPA GALE program. The table below gives a summary about the number of samples splitted into training, dev, and test sets for each dataset in the latest release LDC2017T10 as well as the total statistics.

Dataset	Training	Dev	Test	Totals
BOLT DF MT	1,061	133	133	1,327
Broadcast conversation	214	0	0	214
Weblog and WSJ	0	100	100	200
BOLT DF English	6,455	210	229	6,894
DEFT DF English	19,558	0	0	19,558
Guidelines AMRs	819	0	0	819
2009 Open MT	204	0	0	204
Proxy reports	6,603	826	823	8,252
Weblog	866	0	0	866
Xinhua MT	741	99	86	926
Totals	36,521	1,368	1,371	39,260

Table 2.2.1: Statistic of AMR2.0 (LDC2017T10) corpus

[1] `https://catalog.ldc.upenn.edu/LDC2017T10`

2.2.2 Legal Dataset

In our thesis, we study the domain adaptation for AMR Parsing and Generation in the legal domain. Regarding this purpose, we collect data from several legal texts in English version, i.e. the Japanese Civil Code and the Vietnamese Civil Code.

The first one, which we call **JCivilCode**, is collected from the Japanese Civil Code. The Modern Japanese Legal System comprises of six codes: the Civil Code, the Commercial Code, the Criminal Code, the Constitution of Japan, the Code of Criminal Procedure, the Code of Civil Procedure. In which, the Civil Code was first created in 1896 and updated by time. In the current version, the Civil Code is divided into five parts, each part contains chapters and each chapter contains articles. There are totally 1044 articles in the Civil Code. An article can be a single sentence, or a sentence with multiple itemization, or multiple sentences. In our research, we extract the first four chapters in Part I to annotate.

The pre-processing pipeline consists of the following steps: gathering articles, removing all article prefixes and article IDs, then splitting the article into sentences. Then, we labeled each sentence with an ID containing article name, paragraph index and sentence index. To annotate a sentence, we used the web-based editor provided by ISI group [2]. This editor provides a combination of command line, graphical interface and the detailed guidelines. The Propbank frameset is already integrated in the search engine to reduce the time to choose a proper sense of a word. A group of annotators are given a list of article sentences and annotate corpus independently. After finishing their individual works, the annotators will discuss together, choose the most appropriate annotation and aggregate their outcomes into a single result (Table 2.2.2). The first version of this dataset was introduced in 2017 [42] with 63 samples, and the revised version was updated in 2018 with 128 samples [43]. This updated version contain the first four chapters in Part I of the Japanese Civil Code. Each sample includes one sentence and its corresponding AMR in Penman notation. Readers can follow this URL address to find our dataset `https://github.com/sinhvtr/legal_amr`.

(Age of Majority) Article 4 The age of majority is reached when a person has reached the age of 20.	# ::id 4.1.1 # ::snt The age of majority is reached when a person has reached the age of 20. (r / reach-01 :ARG1 (a / age :mod (m / majority)) :time (r2 / reach-01 :ARG0 (p / person) :ARG1 (a2 / age :quant (t / temporal-quantity :quant 20 :unit (y / year)))))

Table 2.2.2: An example of preprocessing and annotating in our dataset JCivilCode

The remaining dataset is created from the Vietnamese Civil Code (**VNCivilCode**). We use two common AMR parsers, namely JAMR and CAMR, to automatically parse the whole corpus. Each source sentence will provide two AMR graphs using these two

[2] `https://www.isi.edu/cgi-bin/div3/mt/amr-editor/login-gen-v1.7.cgi`

parsers. Since this dataset is not verified by human experts, we call it the silver data and only use for fine-tuning our model in chapter 5. Table 2.2.3 shows some overview information about JCivilCode and VNCivilCode.

Table 2.2.3: VNCivilCode and JCivilCode statistic

	VNCivilCode	**JCivilCode**
Total number of samples	3,070	128
Average sentence length	29	31
Max sentence length	151	107
Average number of nodes in a graph	17	28
Max number of nodes in a graph	93	96
Vocabulary size	3,026	796
Total number of tokens	68,614	4,042

2.3 Parsing and Generation Evaluation

2.3.1 Parsing Evaluation

To measure the similarity of two semantic graphs, Cai et al. [20] introduced the Smatch (Semantic Match) score. This score measures the level of element overlapping between two input graphs by aligns variables from one graph to another and compares the matching triples. Smatch score has been widely used in measuring the accuracy of AMR parsers. Given the parsed graph H (hypothesis) and the gold annotation graph G in Penman format, the Smatch metric operates in two steps. First, (i) it aligns the variables in H and G in the best possible way, by finding a mapping function that yields a maximal set of matching triples between H and G. Then (ii) it computes Precision, Recall and F1 score based on the set of matching triples returned by the alignment search.

We give an example of calculating Smatch score as follow: assume that a machine generates the AMR graph of the sentence "The boy eats the pizza" and the gold standard AMR graph is the graph of the sentence "The boy wants his pizza".

Smatch score computes the F-score on the overall of two given graphs. To have a deeper analysis of parsing performance, Damonte et al. [21] proposed a test-suite which calculates F-score on various perspectives as follow:

- *Unlabeled*: the Smatch score computed on the predicted graphs after removing all edge labels (e.g., *:ARG1, :poss*)
- *No WSD*: the Smatch score when ignoring Propbank senses (e.g., *want-01* vs *eat-02*)
- *Name Entity*: the F-score computed on the named entity recognition (every *:name* roles)
- *Wikification*: the F-score computed on the concept related to definition in Wikipedia (*:wiki* roles)

Sentence	The boy eats the pizza		The boy wants his pizza	
PENMAN	(a / eat-01 :arg0 (b / boy) :arg1(c / pizza))		(x / want-01 :arg0 (y / boy) :arg1(z / pizza) :poss y)	
Logical triples	$\exists a, b, c : instance(a, eat-01) \wedge instance(b, boy) \wedge instance(c, pizza) \wedge arg0(a, b) \wedge arg1(a, c)$		$\exists x, y, z : instance(x, want-01) \wedge instance(y, boy) \wedge instance(z, pizza) \wedge arg0(x, y) \wedge arg1(x, z) \wedge poss(z, y)$	
Alignments	Matches	Precision	Recall	F-score
a=x, b=y, c=z	4	4/5	4/6	0.73
a=x, b=z, c=y	0	0/5	0/6	0
a=y, b=x, c=z	1	1/5	1/6	0.18
a=y, b=z, c=x	0	0/5	0/6	0
a=z, b=x, c=y	0	0/5	0/6	0
a=z, b=y, c=x	1	1/5	1/6	0.18
SMATCH				0.73

Table 2.3.1: Example of calculating Smatch score

- *Negation*: the F-score on the negation detection (*:polarity* roles)
- *Concepts*: the F-score on the concept identification task
- *Reentrancies*: the Smatch computed on reentrant edges only (a node participates in multiple semantic relations)
- *SRL*: the Smatch score computed on :ARG-i roles only

2.3.2 Generation Evaluation

Basically, the AMR graph abstracts away many surface form of words in a sentence, e.g. the verb tense, quantity, etc. From a given graph, there are multiple possible sentences represent the idea in that graph. However, in the benchmark AMR dataset, there is only one sentence corresponding with one graph. This leads to the use of machine translation metrics for evaluating AMR generation performances, instead of other metrics used in image captioning or text summarization.

The most common approach to evaluate a text generator is to use the bilingual evaluation understudy score, or BLEU for short [44], that originates from the machine translation task. A perfect match results in a score of 1.0, whereas a perfect mismatch results in a score of 0.0. Given a candidate sentence (hypothesis) c and a reference sentence r, the core idea of BLEU is to count the number of matching phrases, or n-grams (i.e. contiguous phrases consisting of n words), between c and r. This matching number is then divided by the total number of n-grams in c to normalize the final result. In most cases, this computation is done not only for one but for several values of n and the their results are averaged subsequently to obtain the final one. A common choice of n is 1, 2, 3, 4. BLEU metric is considered to use a modified version of the precision

score since machine translation systems often generate more words than the reference text. There are no calculation of Recall in BLEU, instead, the authors introduced "Brevity Penalty" to prevent very short candidates from receiving too high a score. Other versions of BLEU such as clipping the count of candidate n-gram matches must be made in order to make the resulting score more meaningful. In our work, we do not discuss these modifications, readers are recommended to refer to Papineni et al. [44] for the detail computation. We use the multi-bleu script in PERL language that is commonly used in this field [3].

Another metric we use for evaluating the generation performance is METEOR [45], which scores machine translation hypotheses by aligning and counting explicit word-to-word matches between the translation and a given reference translation. Alignments are based on exact, synonym, stem, and paraphrase matches between words and phrases in the hypothesis and the reference. All matches are generalized to phrase matches with a span in each sentence. Any word occurring within the span is considered to be covered by the match. Alignment resolution is conducted as a beam search using a heuristic based on specific criteria. Comparing to BLEU score, METEOR combines Precision and Recall instead of BLEU's brevity penalty, and METEOR matches between translation and references including semantic equivalents (e.g. synonyms, paraphrases). METEOR is also capable of optimizing correlation with different types of human judgments for each language thank to the ability to tune its parameters. Hence, METEOR score shows better correlation with human evaluation, especially at the segment-level of sentence. In our work, we use the METEOR version 1.5, latest released in 2020 [4].

Taking an example of calculating METEOR score, given the human translated sentence (reference): *"the **US weapon** will be handed over to the **army** within **two weeks**"* and the generated sentence (candidate): *"in **two weeks US weapons** will give **army**"*. We have:

$length(reference) = 13, length(candidate) = 8$

$P = \frac{5}{8}, R = \frac{5}{13}, F_{mean} = \frac{10 \times P \times R}{(9 \times P + R)} = 0.4$

Fragment: 3 frags over 5 words, $frag = (3-1)/(5-1) = 0.5$

Discounting factor: $DF = 0.5 \times (frag^3) = 0.0625$

METEOR score = $F_{mean} \times (1 - DF) = 0.4 \times 0.9375 = 0.375$

2.4 Deep Learning for AMR Parsing and Generation

In this section, we provide some background knowledge in neural encoder-decoder architecture, along with the attention mechanism that is widely used in lots of NLP tasks. Recent research in AMR parsing and generation also relies on this architecture to build their models.

2.4.1 Encoder-Decoder Architecture

Introduced by Sutskever et al. [46], sequence to sequence (seq2seq) models opened a successful era for machine translation using deep neural netwoks (Neural Machine

[3] `https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl`

[4] `https://www.cs.cmu.edu/~alavie/METEOR/`

Translation - NMT), rather than traditional methods mostly based on statistics. NMT is a neural network that is trained in an end to end fashion for translating one language into another. In seq2seq models, the idea is to have two recurrent neural networks (RNNs) with an encoder-decoder architecture: read the input words one by one to obtain a vector representation of a fixed dimensionality (encoder), and, conditioned on these inputs, extract the output words one by one using another RNN (decoder). The figure below shows an overview of the encoder-decoder architecture for NMT.

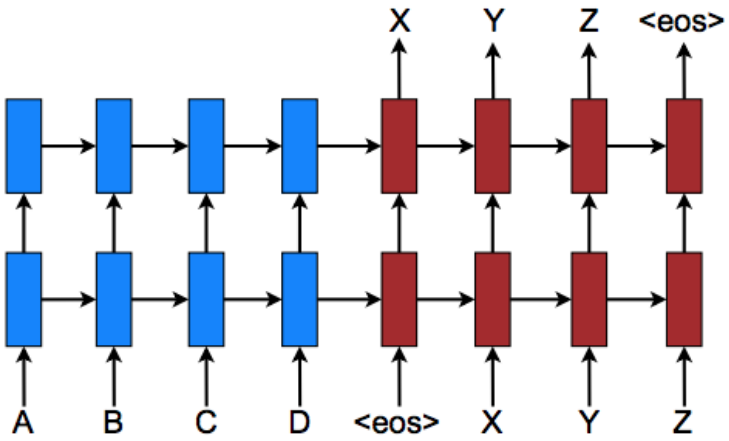


Figure 2.4.1: Encoder-decoder architecture for Neural Machine Translation

NMT directly models the conditional probability $p(y/x)$ of translating a source sentence $(x_1, x_2, ..., x_n)$ into a target sentence $(y_1, y_2, ..., y_n)$. A typical NMT includes two core elements: An encoder which is responsible to compute the representation S from the input sentence; A decoder which generates output words one at a time and decomposes the conditional generation probability as:

$$logP(y|x) = \sum_{j=1}^{m} logP(y_j|y_{<j}, S)$$

One could parameterize the probability of decoding each word $y(j)$ as:

$$P(y_j|y_{<j}, S) = softmax(g(h_j))$$

where $h(j)$ could be modeled as: $h(j) = f(h_{j-1}, S)$
in which:

- g: a transformative function that produces the output as a vocabulary size vector
- h: a Recurrent Neural Network (RNN) hidden unit
- f: a function which computes the current hidden state given the previously one.

The training objective for the translation process could be framed as: $J_t = \sum_{x,y \in D} -logP(y|x)$
The trouble with seq2seq is that the only information that the decoder receives from the encoder is the last encoder hidden state, a vector representation that is like a numerical summary of an input sequence. Thus, for a long input text, we unreasonably expect the decoder to use just this one vector representation to output a translation. Instead of just one vector representation, let's give the decoder a vector representation

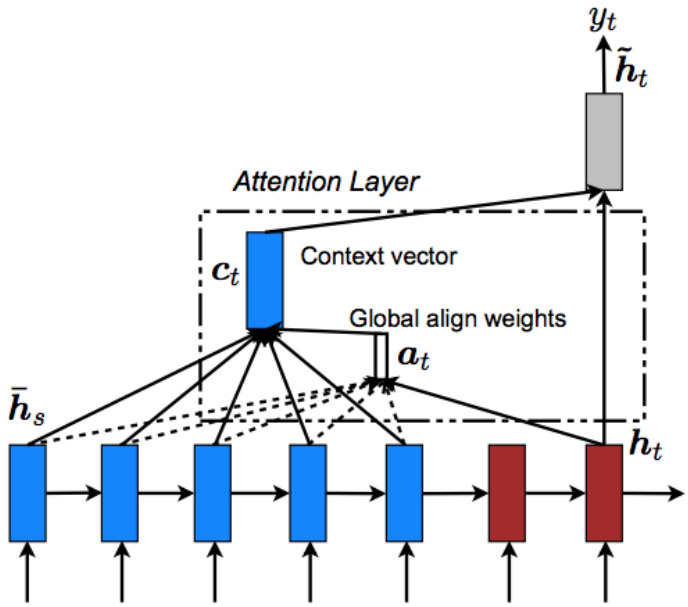


Figure 2.4.2: Encoder-decoder model with global attention

from every encoder time step so that it can make well-informed translations. Attention was created to perform that.

The above figure illustrates an RNN based encoder-decoder architecture with attention. Where:

- $h(t)$: The hidden target state
- $c(t)$: The context vector from source side
- $y(t)$: The current target word
- $\overline{h}(t)$: The attention hidden state
- $a(t)$: The alignment vector

There are 2 types of attention, as introduced in Luong et al. [47]. The type of attention that uses all the encoder hidden states is also known as global attention. In contrast, local attention uses only a subset of the encoder hidden states.

Both two attention based methods above follow these common steps:

- Both attention types first take the hidden state $h(t)$ at the top layer of a stacking Long Short Term Memory (LSTM) network as their input.
- They derive the context vector $c(t)$ to capture relevant source side information in order to predict $y(t)$. Basically, $c(t)$ is the context built for every word depending upon its alignment weights and hidden state of encoders.
- Compute $\overline{h}(t)$ by concatenating $h(t)$ and $c(t)$.

$$\overline{h}_t = tanh(W_c[c_t; h_t])$$

- The attention vector is transformed using the softmax layer to produce the predictive distribution. The softmax layer is commonly used in this circumstance to help find the most probable word from all the available words in the vocabulary.

The global attention takes into account all encoder hidden states to derive the context vector ($c(t)$). In order to calculate $c(t)$, a variable length alignment vector $a(t)$ has to be computed. It is obtained by computing a similarity measure between the source hidden state $h(t)$ and the target hidden state $\overline{h}(t)$. The greater similar states in encoder and decoder, the more the same meaning should be referred.

Alignment Vector ($a(t, s)$):

$$a_t(S) = \frac{exp(score(h_t, \overline{h}_s))}{\sum'_s exp(score(h_t, \overline{h}_s))}$$

Where $score(h_t, \overline{h}_t)$ could be computed by dot product, general or concatenation.

2.4.2 The Rise of Attention

RNN with attention mechanism provided an effective translation architecture. However, some problems are still not solved, e.g. processing inputs (words) in parallel is not possible. For a large corpus of text, a parallel computing mechanism increases the translating time a lot. In 2017, Vaswani et al. [40] introduced the Transformer network, which solves this problem by using Convolutional Neural Networks (CNN) together with attention models.

The Transformer employs an encoder-decoder structure, consisting of stacked encoder and decoder layers. Given a list of vectors as input, the encoder passes these vectors into a self-attention layer (or multi-head attention), then into a position-wise feed-forward neural network, and then sends out the output to the next encoder. The decoder uses masking in its self-attention to prevent a given output position from incorporating information about future output positions during training. It uses residual connections around each of the sub-layers, followed by layer normalization [48]. Figure 2.4.3 illustrates the overall architecture of the Transformer network.

In detail, self-attention is a sequence-to-sequence operation: it takes a sequence of vectors $x = (x_1, ..., x_n)$ as input, and produce a sequence of vectors $z = (z_1, ..., z_n)$ by taking a weighted sum over all the input vectors:

$$z_i = \sum_j w_{ij} x_j$$

where the weight w_ij is not a parameter, it is derived from a function over x_i and x_j. In the basic self-attention, every input vector x_i is used in three different ways in the self attention operation:

- It is compared to every other vector to determine the weights for its own output z_i (*query*)
- It is compared to every other vector to determine the weights for the output of the j-th vector z_i (*key*)
- It is used as part of the weighted sum to compute each output vector once the weights have been established. (*value*)

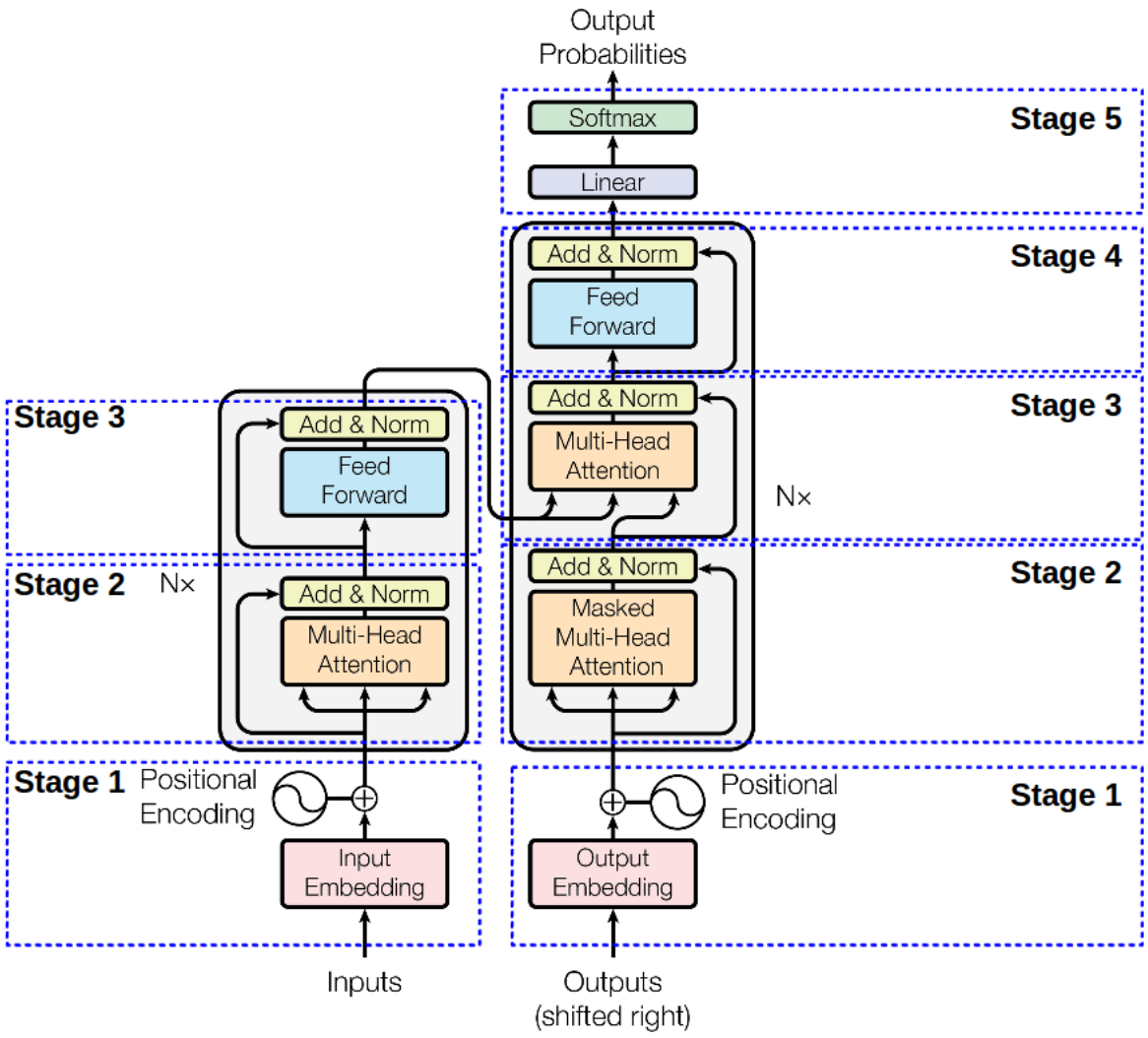


Figure 2.4.3: Transformer architecture

Let's denote new vectors for each role, by applying a linear transformation to the original input. In practice, we add three weight matrices W_q, W_k, W_v and compute three linear transformations of each x_i, for the three different role. These controllable weight matrices allow the self-attention to modify the incoming vectors to suit the three roles they must play.

$$q_i = W_q x_i,\ k_i = W_k x_i,\ v_i = W_v x_i$$
$$w'_{ij} = \frac{q_i^T k_j}{\sqrt{k}}$$
$$w_{ij} = softmax(w'_{ij})$$
$$z_i = \sum_j w_{ij} x_j$$

In the original paper[40], the author refined the self-attention layer by adding a mechanism called *multi-headed* attention. For an input x_i, each attention head produces a different output vector z_i^r. We concatenate these, and pass them through a linear transformation to reduce the dimension back to k. This improves the performance of the attention layer by expanding the model's ability to focus on different positions of the source sequence. Furthermore, multi-head mechanism gives the attention layer multiple sets of Query/Key/Value weight matrices, enlarging the representation spaces.

Unlike RNN networks, the self-attention network cannot naturally understand the order of the words in the input sequence. Hence, the output of the self-attention network could be the same if we shuffle the words in the input. That is why the authors introduced *Positional encoding* method to encode the relative/absolute positions of the input sequence as vectors and added them to the input embeddings. There are many choices of positional encodings, by choosing a function $f : N \rightarrow R^k$ to map the positions to real valued vectors, and let the model figures out how to interpret these encodings (in the original paper[40], the author use sine and cosine functions of different frequencies).

At the time of introduction in 2017, the Transformer outperformed other NMT approaches in translation quality while being more parallelizable, helps reduce training time significantly. This succcess leads to a broaden applications of the Transformer architecture in various tasks, not only in NLP field.

2.5 Chapter Summary

In this chapter, we already presented the necessary background knowledge for the AMR problems. We introduced the AMR with definition, basic notations, illustration examples and potential application in NLP. We introduced the AMR datasets used in our experiment, including our annotated legal dataset JCivilCode. We provided useful statistical information about these datasets as well as the evaluation methods for both parsing and generation tasks that will be used in our experiments in the next chapters. Recent development in deep learning techniques for AMR, i.e. the encoder-decoder architecture and the transformer network, were also briefly described. In the upcoming chapter, we will propose our first model for AMR-to-text generation using the self-attention mechanism.

Chapter 3

AMR Generation with Self-attention Mechanism

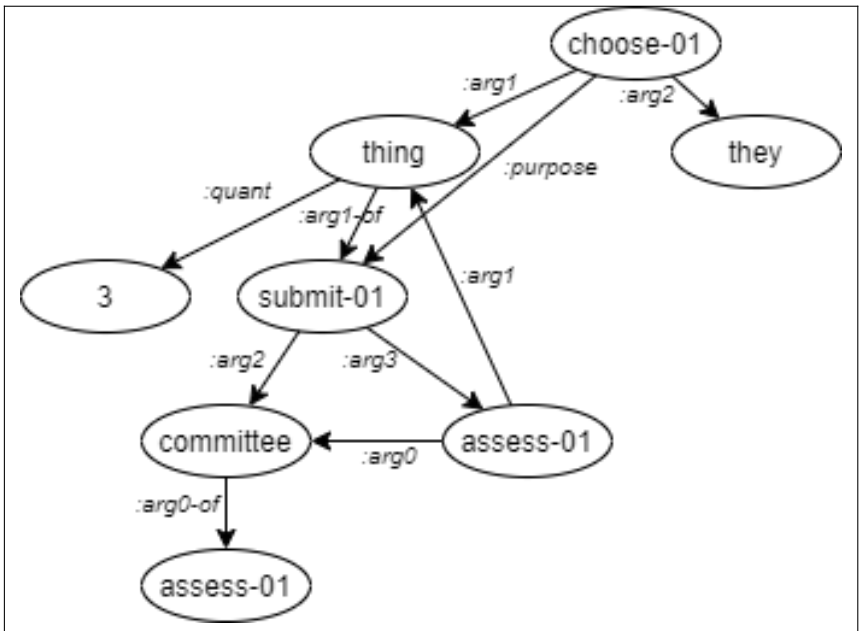
3.1 Introduction

As mentioned in Chapter 1, AMR can be applied as an intermediate meaning representation to help solve various tasks in NLP, such as machine translation [39], text summarization [31], or machine comprehension [38]. For applying in those tasks, the AMR parsing and generation problems have to be done effectively. Comparing to a number of methods proposed for the parsing task, there are not so many published generation approaches. This generation problem is challenging because AMR graphs abstracted away tense, number as well as functional words such as articles, prepositions. In the example shown in Table 3.1.1, the AMR graph is parsed from the source sentence: *"Choose 3 from them to submit to an assessment committee to assess."*. There are two nodes expressing the concept *assess-01*, but the surface words corresponding to them are different: *assessment* and *assess*. Several words are excluded from the AMR, e.g. *from, to, for* and several nodes have reentrancies, e.g. *assess-01, submit-01*.

Recent methods for AMR generation problem are based on the success in deep neural network with the encoder-decoder architecture, which is commonly used in machine translation problems. Previous works considered the input for the encoder side could be a sequence of AMR in the Penman notation ([13][1][15]) or a graph structure ([14][16] [18][19]). Both these two approaches are capable of decoding natural text, but still suffer from several types of error, e.g. word repetition, ungrammatical sentence, etc.

In this work, inspired by the transformer architecture introduced by Vaswani *et al.* [40] in 2017 that achieved outstanding performance on machine translation task, we investigate the use of its core component, the self-attention mechanism. Specifically, we attempt to incorporate this component to the encoder-decoder architecture in order to generate text from AMR graphs. We conduct experiments on both sequence-to-sequence model and graph-to-sequence model. Our experimental results show a comparative BLEU score on the benchmark AMR dataset LDC2017T10, i.e. 18.36 and 19.45 BLEU score with sequence-to-sequence approach and graph-to-sequence approach, respectively.

Table 3.1.1: AMR graph corresponding to the sentence *"Choose 3 from them to submit to an assessment committee to assess."* in graph format and Penman notation format




```
(c / choose-01
    :ARG1 (t / thing :quant 3
        :ARG1-of (s / submit-01
            :ARG3 (a2 / assess-01
                :ARG1 t
                :ARG0 c1)))
            :ARG2 (c1 / committee
                :ARG0-of (a / assess-01))
    :ARG2 (t2 / they))
    :purpose s
```

3.2 Related Works

In this section, we provide a brief summary of previous methods in AMR generation using deep neural network approaches. Most studies in AMR-to-text generation regard it as a translation problem and are motivated by the recent advances in neural machine translation (NMT). The first NMT model for AMR generation was proposed by Pourdamghani *et al.* [13]. Given an AMR graph in Penman notation format, the authors converted that graph to a sequence of text through a linearization and simplification process. For instance, the graph in the previous example can be linearized as follow:

choose-01 :ARG1 (thing :quant 3 :ARG1-of (submit-01 :ARG2 (committee :ARG0-of (assess-01)) :ARG3 (assess-01 :ARG0 committee :ARG1 thing))) :ARG2 they :purpose submit.

With these pairs of linearized AMRs and the corresponding sentences, the AMR generation task could be treated as a translating a sequence from AMR language to a sequence in human language. The authors implemented a phrase-based translation model (PBMT), followed string-based statistical machine translation that ignored much of the structure in both "language" but provided a strong baseline.

Following this approach but instead of using phrase-based, Konstas *et al.* [1] pro-

posed the first neural model (NeuralAMR). Their method could be applied for both the parsing and generation directions. The authors used an encoder-decoder architecture with a LSTM neural network, which we illustrate in Figure 3.2.1. Since this model required a huge amount of training data, Konstas *et al.* (2017) adopted a self-training algorithm. Specifically, they first built a baseline parser with the labeled data only, then used this parser to automatically produce millions of unlabeled sentences extracted from a text corpus before training their main system. The AMR graphs obtained from this process are then used as additional training data for both parsing and generation system. Since many tokens in the training data do not appear in the vocabulary of AMR, there is a risk of data sparsity which was addressed by Peng el al. [49]. To tackle this problem, NeuralAMR adopts an anonymization algorithm. Specifically, the subgraphs that represent open-class tokens (such as *"country :name name :op1 United :op2 Kingdom"*) was first replaced with several kinds of predefined placeholders (such as *loc0*) before feeding to the decoder. Then the corresponding surface tokens (such as "United Kingdom") would be recovered after decoding.

After the success of this neural method, Cao and Clark [15] factor the generation process by leveraging syntactic information to improve the performance. However, both the AMRs and the constituency graphs are linearized, which implies that important parts of the graphs are unable to be represented well (e.g., coreference).

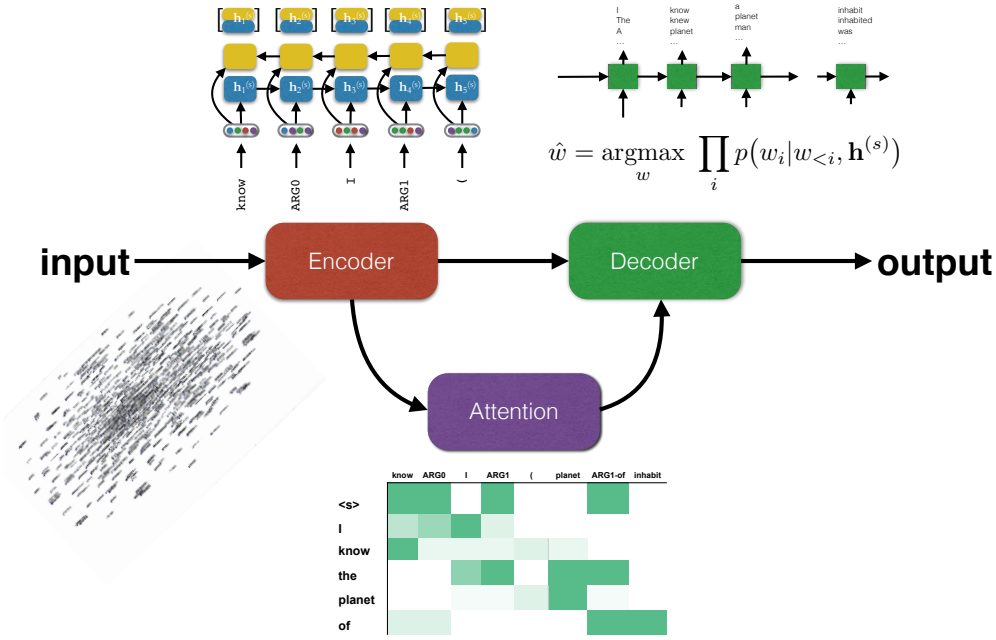


Figure 3.2.1: Sequence to sequence model in NeuralAMR [1]

Another neural-based method for AMR generation was proposed by Song *et al.* [14]. The author incorporated a character-level LSTM over the characters of input tokens and a copy network [50] on top of the decoder side. This architecture not only prevented the data sparsity problem, but also helped generate the named entities, dates and numbers effectively. The authors also proposed the first novel graph to sequence model (Graph2Seq) for AMR generation, in which they encoded the AMR with a graph state encoder, performing through a graph-state transition rather than normal hidden state transition in a typical LSTM network. This graph encoder reduced the problem of information loss through the linearization and anonymization process, especially

when the graph becomes large. The author did not use a self-training strategy like Konstas et al., instead, they used an external AMR parser call JAMR to automatically annotate AMR. With two millions of external data samples, this graph to sequence model achieved the state of the art result on the benchmark dataset LDC2017T10 in the year of 2018.

In 2019, deep learning approaches continued to dominate in AMR-to-text generation problem. Damonte et al. [16] provided a survey of various types of neural encoders. They investigate the impact of reentrancies (nodes with multiple parents) on the generation performance by comparing graph encoders to tree encoders, where reentrancies are not satisfied. Their results showed that the information obtained from reentrancies and long-range dependencies contribute to higher overall scores for graph encoders architecture. Later, Leonardo et al. [18] proposed a graph encoder with a dual graph representation method, which encodes different but complementary perspectives of the structural information contained in the AMR graph. Their model was capable of learning both top-down and bottom-up representations of AMR nodes, which capture contrasting views of the graph. The author also investigated different node message passing strategies by employing different types of graph encoders to compute node representations based on incoming and outgoing nodes. Their method achieved competitive result on BLEU and METEOR scores on the benchmark dataset LDC2017T10.

3.3 The incorporated self-attention mechanism

3.3.1 Our baseline model

In our work, we adopt both the sequence to sequence and graph to sequence model provided by Song *et al.* work [14] to build our baseline model. The input for these models could be either the linearized sequence from an AMR graph (in Penman notation) or the graph itself in the graph format. To tackle the data sparsity problem, we keep using the character-level LSTM over input tokens and the copying mechanism. As reported in the original paper, this baseline model obtained 20.6 and 22.8 BLEU scores when testing with the dataset LDC2015E86 for sequence-to-sequence model and graph-to-sequence model, respectively. However, when we retrain these models with the default configurations along with the source code provided by the authors, the BLEU scores only reach 15.49 and 20,76, respectively.

3.3.2 Self-attention in the sequence-to-sequence model

With an input AMR in the Penman notation format, we adopt the linearization and simplification algorithm introduced by Konstas *et al.* [1] to obtain a sequence of tokens $v_1, \ldots, v_n$, where n is the number of tokens. We remove all verb senses and variable names from the annotation so that all vertices in the preprocessed AMR could be considered as a word in the normal dictionary. This helps the popular word embedding like Word2Vec, Glove, Fastext can easily apply in the embedding layer. For example, the AMR annotation shown in Table 3.1.1 can be preprocessed as follow:

choose :ARG1 (thing :quant 3 :ARG1-of submit :ARG2 (committee :ARG0-of assess) :ARG3 (assess :ARG0 committee :ARG1 thing))) :ARG2 they :purpose submit.

3.3The incorporated self-attention mechanism

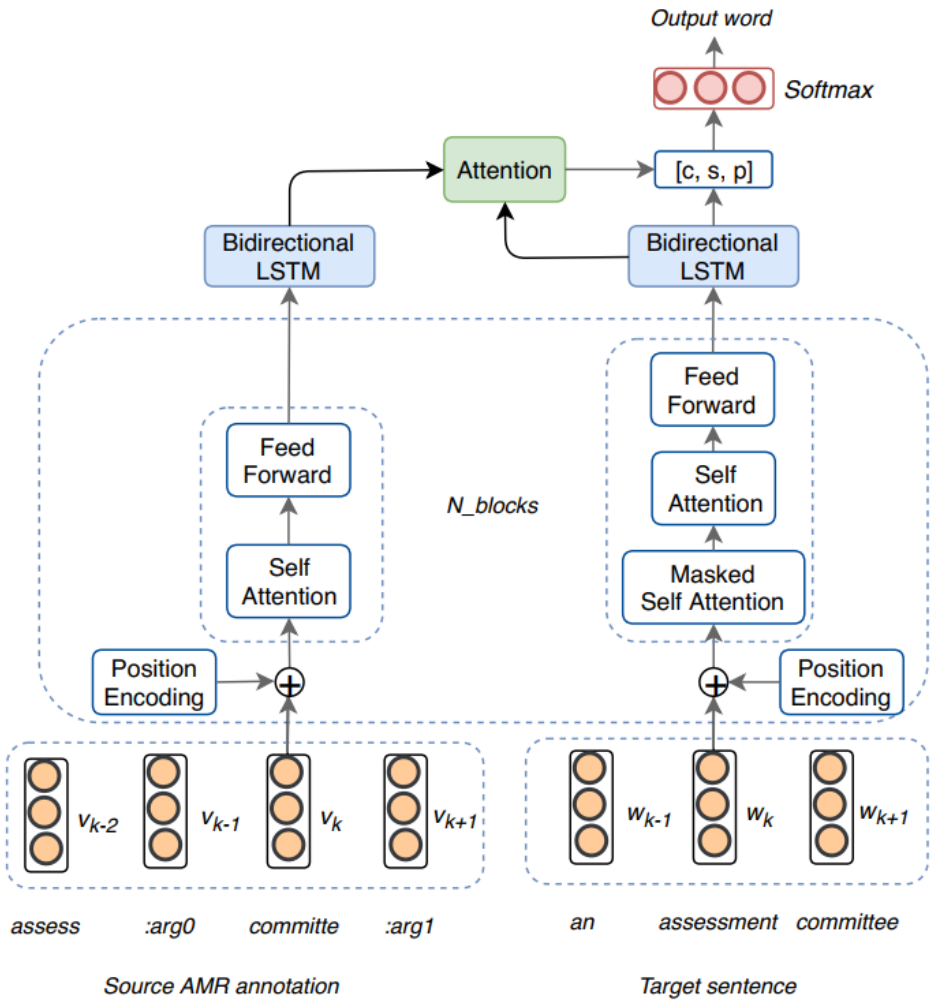


Figure 3.3.1: Sequence to sequence model with the incorporated self-attention layers for AMR generation

(Remove *-01* from *choose-01*, *-01* from *assess*, all the variables *p, t, s*, etc. are excluded)

The whole model architecture is represented in Figure 3.3.1. We follow the transformer architecture in Vaswani *et al.* work [40] to build the lower layers for our sequence to sequence model. Specifically, the source tokens will be feed into two sub-layers: self-attention layer followed by a position-wise feed forward layer; and the target tokens will be processed with three sub-layers: self-attention layer followed by a vanilla attention, then followed by a position-wise feed forward layer. The outputs of these layers are sent to a bidirectional LSTM model similar to the one designed in Song *et al.* work. In the decoder sidce, the self-attention layers use masking to prevent a given output position from incorporating information about future output positions during the training process.

In both the encoder and decoder side, the self-attention sub-layers employ h attention heads. To produce the sub-layer output, results from each head are concatenated together by applying a parameterized linear transformation. Each attention head operates on an input sequence of tokens, $v = (v_1, ..., v_n)$ of n elements where $v_i \in R^{d_v}$, and computes a new sequence $z = (z_1, ..., z_n)$ of the same length where $z_i \in R^{d_z}$. When generating the t-th word, the decoder relies on the attention memory, the previous

hidden state of the LSTM layers, the probability distribution to decide whether to copy the word from source tokens or generate a new one (the value c, s and p in Figure 3.3.1, respectively).

3.3.3 Self attention in the graph-to-sequence model

Differ from the sequence input, to deal with the graph structure of the original AMR, we adopt the graph encoding method in Song *et al.* [14]. For a graph $G = \{V, E\}$, each node $v_i \in V$ is represented by a hidden state vector h^i. The state of the graph can thus be represented by the set $g = \{h^i\}$. We capture the information exchange between a current node v_i and all nodes connected to it (both incoming and outgoing nodes) by a sequence of state transitions $\{g_0, g_1, ..., g_k\}$. In detail, the transition from g_{t-1} to g_t consists of a hidden state transition for each node h^i_{t-1} to h^i_t. The initial state g_0 includes a set of initial node states $h^j_0 = z_0$, where z_0 is a vector of zero valuee.

A LSTM network is used to model the graph-state transition process. At each graph-state transition step t, the model performs direct communication between a node and all of its incoming and outgoing nodes. The memory for h^j_t is stored in a cell c^j_t, with the use of an input gate i^j_t, an output gate o^j_t and a forget gate f^j_t to control information flow from the inputs and to the output. Figure 3.3.2 demonstrates the graph state transition process, in which, the information from the current node h^3_{t-1}, its incoming node h^2_{t-1} and outgoing nodes h^4_{t-1}, h^5_{t-1} are all captured by LSTM gates and transferred to h^3_t.

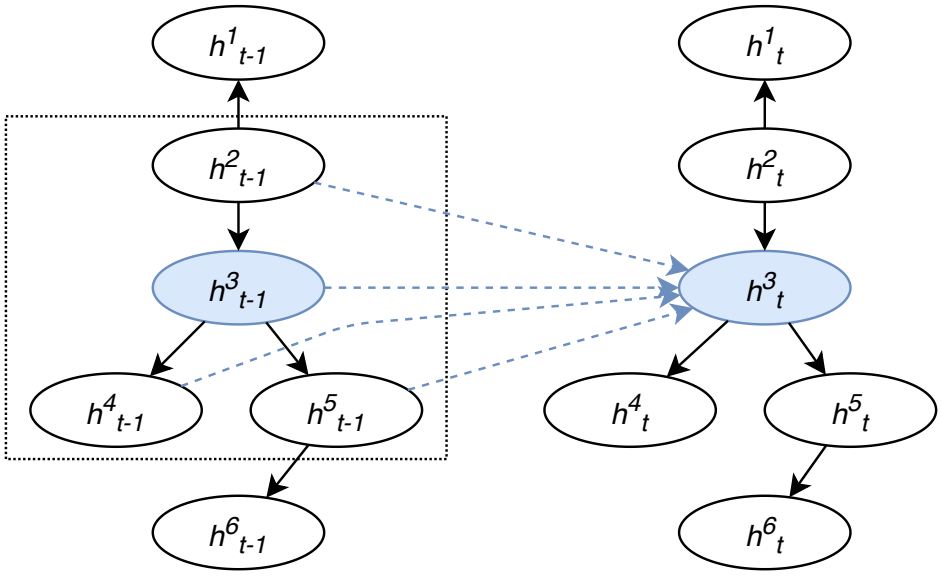


Figure 3.3.2: Graph state transition from g_{t-1} to g_t

The inputs include the edge representations of any edges that are connected to a node v_j, where v_j can be either the source or the target node of the edge. Each edge is defined as a triple $(i; j; l)$, where i and j are the indices of the source and the target nodes, respectively, and l is the edge label (e.g. *ARG0, location, purpose.* $x^l_{i;j}$ is the representation of edge $(i; j; l)$. The inputs for v_j are grouped into incoming and outgoing edges:

$$\phi_{j,in} = \sum_{(i,j,l)\in E_{in}(j)} x^l_{i,j}$$
$$\phi_{j,out} = \sum_{(j,k,l)\in E_{out}(j)} x^l_{j,k}$$

where $E_{in}(j)$ and $E_{out}(j)$ are the sets of incoming and outgoing edges of the node v_j, respectively. We also use an edge embedding to represent the vector of an edge label, with the initial embedding values are all set randomly.

In addition to the edge representations, the model also takes the hidden states of the incoming and outgoing neighbors of each node during a graph-state transition. For example, the states of the incoming and outgoing neighbors of v_j are summed up before being passed to the cell and the gate nodes:

$$\delta_{j,in} = \sum_{(i,j,l)\in E_{in}(j)} h^i_{t-1}$$
$$\delta_{j,out} = \sum_{(j,k,l)\in E_{out}(j)} h^k_{t-1}$$

Based on the above definitions of ϕ_j, δ_j, the state transition from g_{t1} to g_t, as represented by h^j_t, can be defined as:

$$i^j_t = \sigma(W_{i,in}\phi_{j,in} + W_{i,out}\phi_{j,out} + U_{i,in}\delta_{j,in} + U_{i,out}\delta_{j,out} + b_i)$$
$$o^j_t = \sigma(W_{o,in}\phi_{j,in} + W_{o,out}\phi_{j,out} + U_{o,in}\delta_{j,in} + U_{o,out}\delta_{j,out} + b_o)$$
$$f^j_t = \sigma(W_{f,in}\phi_{j,in} + W_{f,out}\phi_{j,out} + U_{f,in}\delta_{j,in} + U_{f,out}\delta_{j,out} + b_f)$$
$$u^j_t = \sigma(W_{u,in}\phi_{j,in} + W_{u,out}\phi_{j,out} + U_{u,in}\delta_{j,in} + U_{u,out}\delta_{j,out} + b_u)$$
$$c^j_t = f^j_t \cdot c^j_{t-1} + i^j_t \cdot u^j_t$$
$$h^j_t = o^j_t \cdot tanh(c^j_t)$$

where i^j_t, o^j_t and f^j_t are the input, output and forget gates mentioned earlier. $W_{x,in}, W_{x,out}, U_{x,in}, U_{x,out}, b_x$, where $x \in \{i; o; f; u\}$, are model parameters.

After k iterations, the last hidden state of the graph is obtained. It contains all the hidden vectors of nodes inside. k is often chosen to be the maximum graph diameter in the dataset (we choose $k = 9$ in our experiments). These node hidden vectors are then passed through the self-attention encoder similar to section 3.3.2.

3.4 Experiments and Results

3.4.1 Dataset and Experiment setup

Table 3.4.1: Hyper-parameter settings for our proposed models

Word vocab size	27,876
Edge vocab size	119
Edge label dimension	100
d_{model}	300
N_heads	6
N_blocks	6
Feed forward dimension	1200

We use the AMR corpus released by LDC (*LDC2017T10*) to conduct our experiments. This dataset contains 36,521 instances for training, 1,368 for development and 1,371 for testing. Each instance contains an English sentence and an AMR graph in Penman notation, as introduced in Chapter 2. Due to the lacking of hardware resources, we skip the experiments on silver data sampled from a external corpus (like NeuralAMR and Graph2Seq using two million sentences from Gigaword corpus).

Table 3.4.2: Results on LDC2017T10 test set in BLEU and METEOR scores

Model	Corpus	Number of training samples	BLEU score	METEOR
Seq2Seq	LDC2017T10	36,521	15.49	23.15
Seq2Seq + Selfatt (Ours)	LDC2017T10	36,521	**18.36**	25.77
Seq2Seq + Selfatt (Ours) + silver data	LDC2017T10 + Gigaword	36,521 + 200k	21.15	27.70
Graph2Seq	LDC2017T10	36,521	20.76 [1]	26.71
Graph2Seq + Selfatt (Ours)	LDC2017T10	36,521	19.45	24.76
Graph2Seq + Selfatt (Ours) + silver data	LDC2017T10 + Gigaword	36,521 + 200k	24.44	29.74
Graph2Seq	LDC2015E86 + Gigaword	16,833 + 2M	**33.0**	31.81
NeuralAMR	LDC2015E86 + Gigaword	16,833 + 2M	32.3	28.32
TSP	LDC2015E86	16,833	22.4	-
SNRG	LDC2015E86	16,833	25.6	-
JAMR-generator	LDC2014T12	10,000	22.0	-
PBMT	LDC2014T12	10,000	26.9	-

In the sequence to sequence experiment, we share the vocabulary between the encoder and the decoder side. The input word embeddings are initialized from the Glove pretrained word embeddings [51] with the embedding size is set to 300. In the graph to sequence experiments, we extract the edge label vocabulary to be used in the graph encoder. The total number types of edge is 118 and the edge label dimension is set to 100. All the baseline hyper-paramenters for the graph-to-sequence and the sequence-to-sequence model are kept the same as in [14]. Other hyper-parameters for the self-attention mechanism, as well as dataset information can be found in Table 3.4.1. Following previous work, we evaluate the generation results with the BLEU metric [44] and METEOR metric [45].

3.4.2 Experimental results

Beside our baseline model, we also compare the performance of our self-attention incorporated models with other works in the literature, i.e. JAMR-generator (alignment-based) [12], PBMT (phrase-based) [13], TSP [52] (graph-based), SNRG (graph-based) [53], NeuralAMR [1] and Graph2Seq [14]. In term of training data, JAMR-generator and PBMT were trained on the old AMR corpus *LDC2014T12* with a small number of training samples (13,051 samples); TSP, SNRG, NeuralAMR and Graph2Seq were trained by a newer version, LDC2015E86 (19,572 samples). In our experiments, we train our models with the latest release AMR corpus, LDC2017T10, which remains the same test set as LDC2015E86, but a bit less than the number of samples in the splitted test set of LDC2014T12. Since Song *et al.* did not publish their results of sequence to sequence model trained on gold data only, we use their default configuration to train a new one on *LDC2017T10* (Seq2Seq). We shows the BLEU scores of all the models

on the LDC test set in Table 3.4.2 .

From the result table, it can be recognized that Selfatt+Seq2Seq outperforms the baseline sequence to sequence model with 2.9 BLEU score increased (from 15.49 to 18.36), and 2.6 METEOR score increased (from 23.15 to 25.77). This proves the effectiveness of our method in incorporating the self-attention mechanism with the baseline sequence to sequence model. However, the graph-to-sequence model combining with the default transformer architecture does not produce high-quality text. It is reported that the Selfatt+Graph2Seq obtains a slightly lower score than the original Graph2Seq model (20.76 to 19.45 in BLEU score, and 26.71 to 24.76 in METEOR score). This result is probably due to the use of position encoding method in our architecture, which is naturally suitable for a sequence of words rather than graph nodes. This motivates us to investigate more further techniques to adapt the graph encoding mechanism.

Comparing to other neural generation model, i.e. the full Graph2Seq model, the NeuralAMR model as well as other traditional methods, our BLEU and METEOR scores are still lower by a large margin. Even when we train our models with more 200,000 samples generated from the Gigaword corpus, similar to the strategy in Song *et al.* [14], our BLEU score only increases to 24.44, a little lower than SNRG and PBMT which were trained with small dataset. As this neural approach requires a huge amount of training data (e.g. Graph2Seq obtains more than 10 BLEU scores improvement after this process), we plan to explore more transfer learning and semi-supervised learning techniques to solve this problem, such as the use of pretrained language model or Noisy Student approach.

3.5 Analysis

3.5.1 Effect of the input size

To observe the effect of various sizes of the source input, we divide the test set to several subsets based on two criteria: the number of tokens in the source linearized AMR and the number of nodes in the source graph. In the first criteria, we divide by the following ranges: *less than 10, 10-20, 20-40, 40-60, 60-80, 80-100* and *more than 100* tokens. We obtain corresponding subsets with the following number of samples: *126, 120, 258, 265, 231, 161, 210*. In the second criteria, we split the test set by different ranges: *less than 10, 10-20, 20-30, 30-40* and *more than 40* nodes, we obtain corresponding subsets with the following number of samples: *323, 397, 337, 193, 121*.

We present the detailed BLEU scores by using Graph2Seq models to generate text from each subsets in Figure 3.5.1, 3.5.3, and by using Seq2Seq models in Figure 3.5.2, 3.5.4. For each architecture, we take the baseline model, the model trained with silver annotated data, the model incorporated with the self-attention mechanism and the model combined both the incorporated self-attention and trained with silver data (200.000 samples extracted from Gigaword corpus and parsed automatically by JAMR [9]).

[1]Though we train the Graph2Seq model with the same setting as the publish source code in `https://github.com/freesunshine0316/neural-graph-to-seq-mp`, but the BLEU score is lower than reported in the original paper (20.76 vs 22.7)

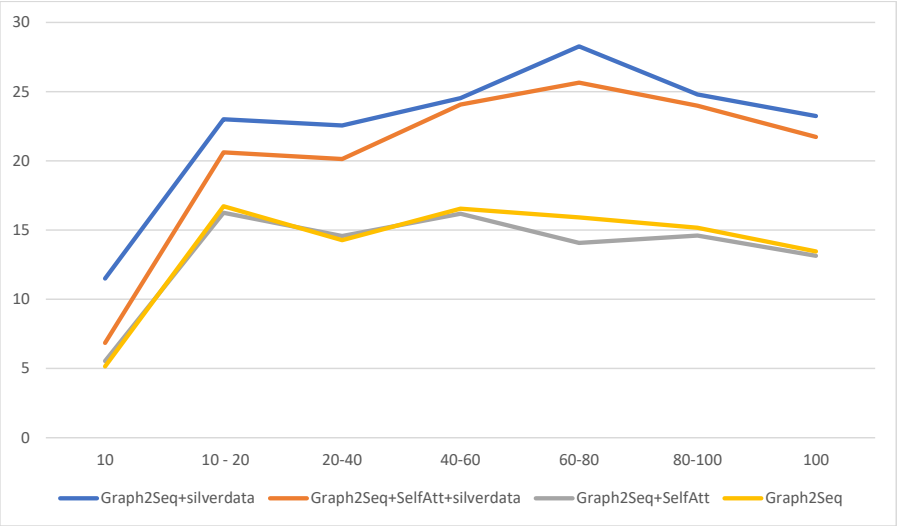


Figure 3.5.1: BLEU scores of Graph2Seq models when generating text from each range of length

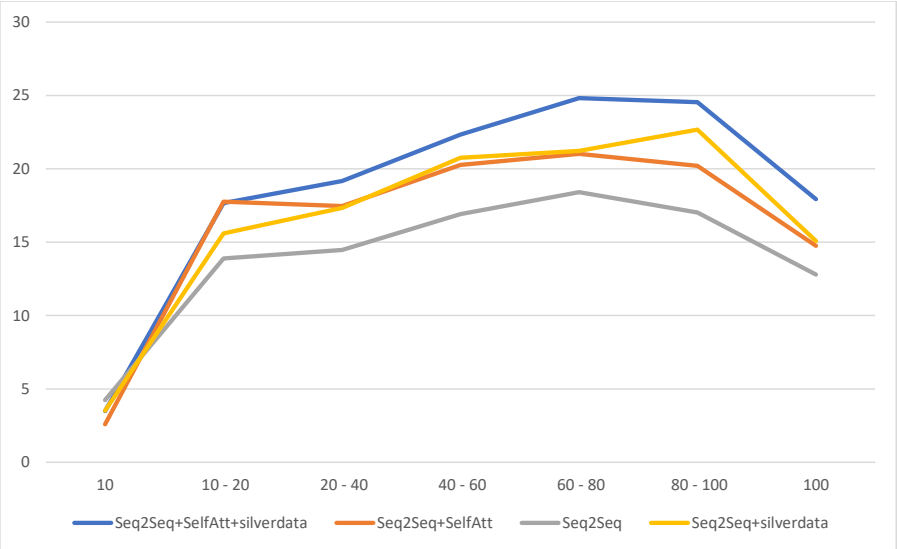


Figure 3.5.2: BLEU scores of Seq2Seq models when generating text from each range of length

Both the Graph2Seq and Seq2Seq models perform well with the source input length in the range between 20 and 100 tokens, and 20 to 40 nodes in the graph, and perform worst with the very small source input. The reason for this low performance may comes from the fact that almost of these short input are single words, dates, name entities and non-grammatically short sentences. Thus, a minor wrong generated word will cause the BLEU scores to decreases a lot.

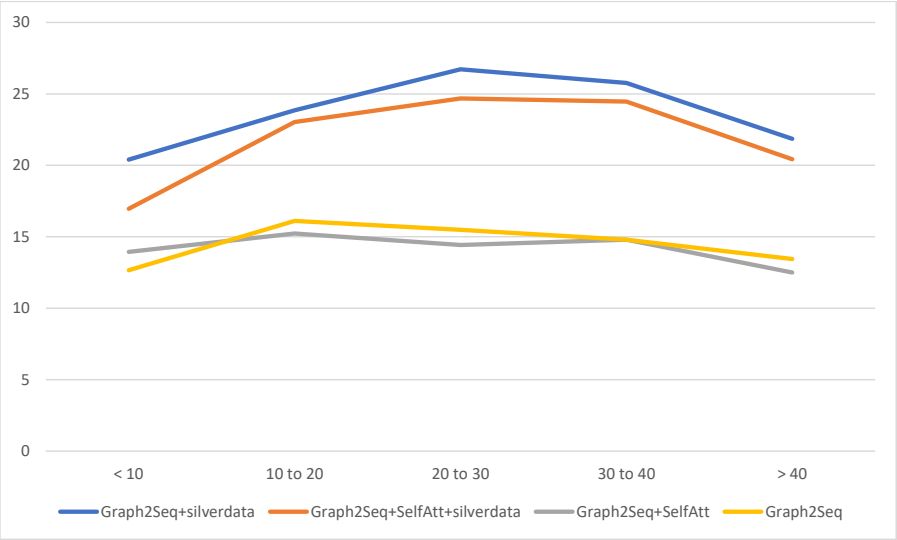


Figure 3.5.3: BLEU scores of Graph2Seq models when generating text from each range of nodes number

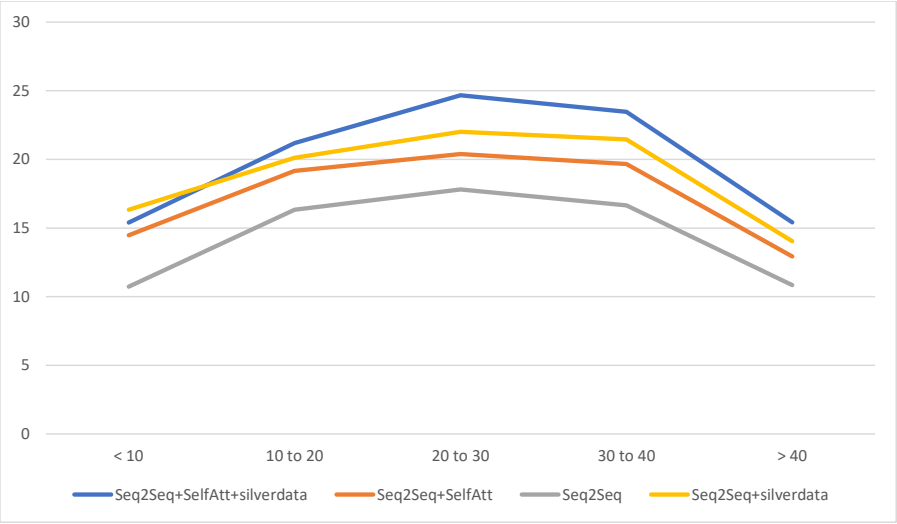


Figure 3.5.4: BLEU scores of Seq2Seq models when generating text from each range of nodes numbe

When the number of tokens in the source input becomes longer or the graph size becomes bigger, Seq2Seq models' performances drop significantly (e.g. from 24.5 in range 80-100 to 17.9 in range over 100, with the Seq2Seq+SelfAtt+silverdata), while Graph2Seq models perform more consistently. In Figure 3.5.1, the BLEU score of all the four models only decrease by a little margin.

The two figures also show the difference between Graph2Seq and Seq2Seq architecture when incorporating the self-attention mechanism. In Seq2Seq, the baseline model combined with the self-attention performs much better than the baseline only. Seq2Seq+self_attention even got higher BLEU score than the Seq2Seq+silverdata in

some ranges. While in Graph2Seq architecture, incorporating the self-attention does not improve the performance. The gap in Figure 3.5.1 only shows the effective of silver annotated data to the baseline model.

3.5.2 Error Analysis

We extract some real outputs of each model in Table 3.5.1 to analyze. Since the AMR graph abstracts away all the tense, number and functional words, there is no information for any model to generate exactly the original text. We select some featured samples to see how different each model provides the output.

Gold Annotation	**SelfAtt + Seq2seq**	**Seq2Seq**	**Graph2Seq**
it's the same old problem	old problem is the same	the same problem is	the same problem is that the same old problem
a large virus repository is located in Siberia	a large virus repository is located in siberia	large virus is in 1	the large virus is in siberia in siberia
the pirates have consistently expressed willingness to negotiate the financial figures	the pirate has consistently expressed a s willing to negotiate with financial	the pirate will consistent with the pirate will will will negotiate a finance figure	the pirates consistently expressed the willingness to negotiate financial figure

Table 3.5.1: Output comparison among our proposed model with the baseline Seq2Seq and Graph2Seq models

In the two first samples, our proposed model Seq2seq+SelfAtt performs better than the others. The outputs generated are close in meaning with the original sentences, despite the differences in words order. Graph2Seq also generates meaningful sentences, however there exists some unnecessary words or phrases (*"the same problem"* and *"in siberia"*. Regarding this aspect, Seq2Seq+SeflAtt helps decrease the repetitive words generated comparing to the two baseline models.

In the third sample, when the sentence becomes longer, the source graph is also more complex. Our model makes some spelling errors (*" a s "*), compared to the Graph2Seq, but the result still expresses the meaning of the sentence. In this case, Graph2Seq recovers almost correctly the source sentence. While the baseline Seq2Seq even produces more repetitive words (*"will will will"*).

3.6 Chapter Summary

To summarize this chapter, we investigated the use of self-attention mechanism in generating natural language from AMR graphs. We incorporated this mechanism with both the sequence-to-sequence and the graph-to-sequence baseline models to observe the effective. Experimental results show that self-attention mechanism improves the performances in sequence-to-sequence models, but does not contribute to the graph-to-sequence models. Overall, our proposed approach obtaine promising results compared

to other deep learning ones, however it is still lower than the state of the art results with the similar amount of training data.

For the future work, we would like to explore more transfer learning and semi-supervised learning techniques, such as the use of pretrained model like BERT, XL-NET to provide a better semantic embedding representation for the input tokens, or Teacher-Student architecture to acquire more high quality training data rather than using silver annotated data like many previous work. We also aim to build a graph to sequence transformer model for AMR generation by applying the relative position encoding introduced by Shaw *et al* recently [54] as a replacement to the current positional encoding, that is naturally suitable for sequence input.

Chapter 4

AMR Parsing for Legal Document

4.1 Introduction

Legal Engineering is a field that studies the methodology and applies information science and software engineering to legal systems in order to support legislation and to implement laws and rules using computers. Since proposed by Katayama *et al.* in 2007 [55], Legal Engineering has grown to become a new trend in computer science. Many tasks related to Legal Engineering need to be solved, e.g. Law Search System, Law Question Answering, Law Summarization System. All of them require the computer to automatically process the text from legal documents, where Natural Language Processing techniques get involved. However, most of the methods for representing a legal sentence are based on logical patterns [56] [57], skipping the semantic information among the words in the sentence. That motivates us to investigate a more semantic approach, i.e. Abstract Meaning Representation, for representing texts in this domain.

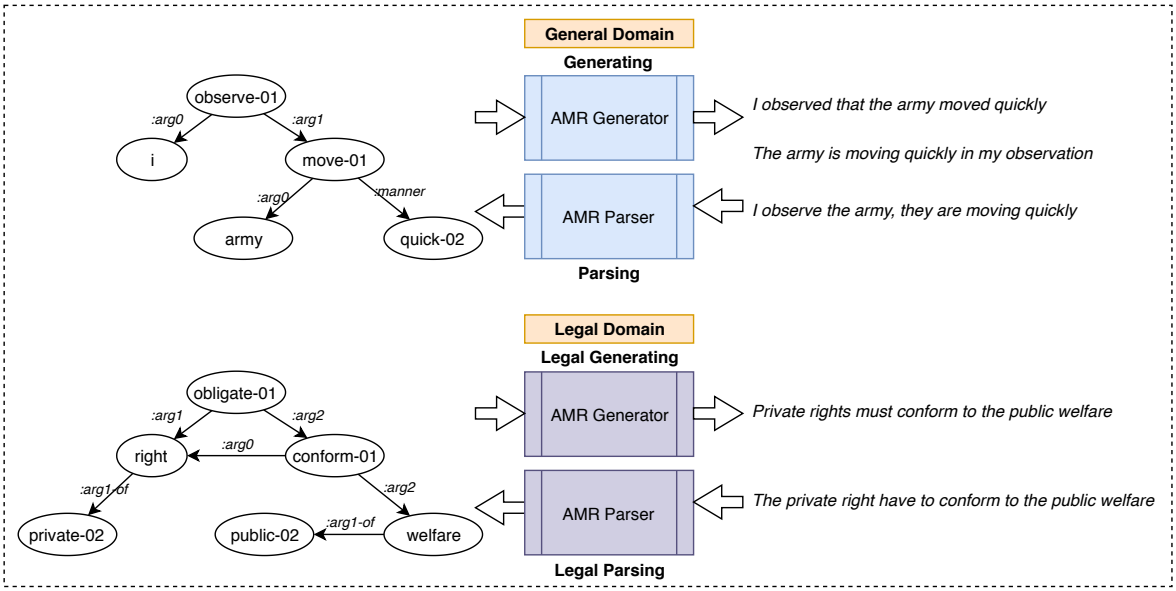


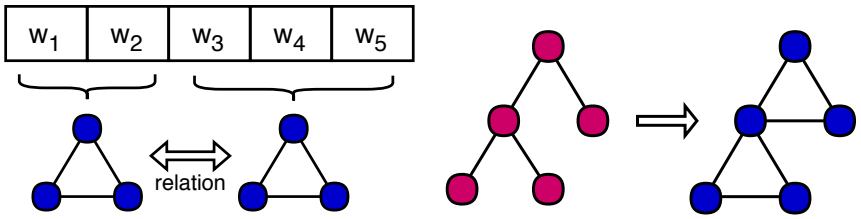
Figure 4.1.1: AMR Parsing and Generation in Legal Domain

Generally, the AMR parsing problem faces a lot of challenges, such as name entity recognition, word-sense disambiguation, data sparsity, etc. Those challenges lead to many proposed approaches, from grammar-based to machine translation methods. Almost of them conduct experiments on a general domain dataset released by Linguistic

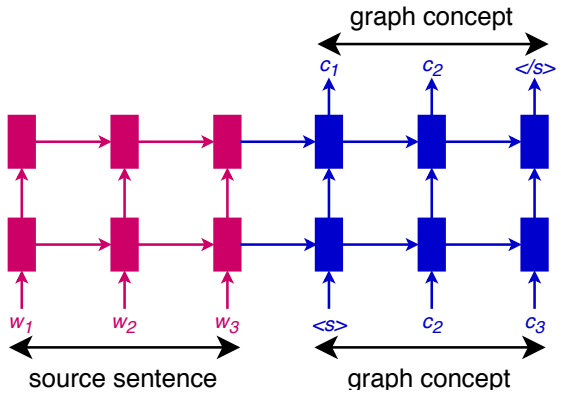
Data Consortium, i.e. LDC2014T10, LDC2015E86, LDC2017T10. However, there are no research for AMR in the legal domain yet, despite lots of differences between text in the general documents and text in the legal documents, e.g. longer sentences, complex logical structures, domain specific terminologies (Figure 4.1.1). These legal characteristics cause lots of difficulties for current AMR parsers comparing to the general domain. In this chapter, we will provide an empirical evaluation of various approaches in AMR parsing on our annotated legal dataset to prove this claim. Specifically, we divide AMR parsing models into three main approaches: Alignment-based, Grammar-based and Neural-based. From each approach, we choose typical models which achieve high scores in the benchmark dataset and re-implement their source codes to parse texts in our legal dataset to AMR. We further report the score of each model and deeper analyze the parsing output.

4.2 AMR Parsing - Main Approaches

As mentioned in the previous section, the task of parsing a natural language text into an AMR graph faces a lot of challenges. To tackle these, many approaches have been proposed, which we group into three main categories: alignment-based, grammar-based and neural-based (Figure 4.2.1). From each approach, we provide a brief description, then choose the best systems that already published the source code to conduct our experiments.



(a) Alignment-based method (b) Grammar-based method



(c) Neural-based method

Figure 4.2.1: Three main approaches in Abstract Meaning Representation parsing

4.2.1 Alignment-based parsing

Flanigan et al. in 2014[2] introduced **JAMR** in 2014, which build a two-stage algorithm: (i) first identifying concepts with an automatic aligner and then (ii) detecting the relations that it obtains between these by searching for the maximum spanning connected subgraph from an edge-labeled, directed graph representing all possible relations between the identified concepts. This method was considered the pioneer in AMR parsing, provided a strong baseline in AMR parsing. Follow this alignment approach, Zhou et al. [58] extended the relation identification tasks with a component-wise beam search algorithm. Chunchuan *et al.* [8] improved this method by considering alignments as latent variables in a joint probabilistic model. They used variational autoencoding technique to perform the alignment inference and archieved the state-of-the-art in AMR parsing in 2018. In our work, we take the JAMR model [2] [1] and LatentGraph [8] [2] to analyze and to conduct experiments.

Basically, the idea of alignment-based methods is to obtain the alignment between words in the source sentence and the graph fragment in AMR. Flanigan *et al.* construct a concept set by aligning the Propbank concepts with the words that evoke them. The authors build an automatic aligner that uses a set of heuristic rules to align concepts to words greedily. They use WordNet dictionary to generate candidate lemmas and a fuzzy match of a concept, that is a word in the sentence containing the longest string prefix match with that concept's label. For example, the fuzzy match for **extend-01** could be aligned with *"extension"* if this word is the best match in the given sentence. We show an example of alignment in Figure 4.2.2, where the span includes three words that express a location name and the aligned graph fragment contains five nodes. This JAMR aligner is widely used in many later works, both for AMR parsing and generation [59][60][13].

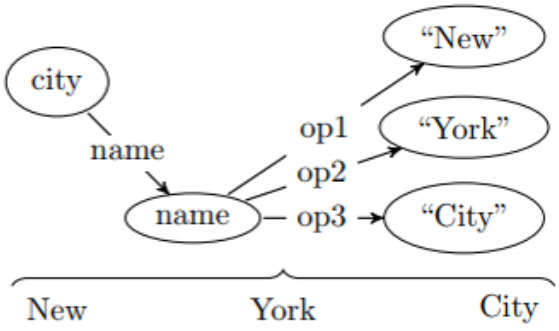


Figure 4.2.2: Alignment between the words span "New York City" and its corresponding AMR fragment [2]

In Flanigan *et al.* method, the first stage is to identify concepts. Given a sentence $w = \{w_1, w_2, ..., w_n\}$, their system segments w into subsequences, denoted $\{w_{b_0:b_1}, w_{b_1:b_2}, ..., w_{b_{k-1}:b_k}\}$, which they called contiguous spans. A span $\{w_{b_{i-1}:b_i}\}$ is then assigned to a concept c_i from the concept set $clex\{w_{b_{i-1}:b_i}\}$, or to θ for words that evoke no concept. The assigning score between the sequence of spans b and the graph

[1] `https://github.com/jflanigan/jamr`
[2] `https://github.com/ChunchuanLv/AMR_AS_GRAPH_PREDICTION`

fragment c is calculated as follow:

$$score(b, c; \theta) = \sum_{i=1}^{k} \theta^T f(w_{b_{i-1}:b_i}, b_{i-1}, c_i), \tag{4.1}$$

where f is a feature vector representation of a span and one of its concept graph fragments in the sentence. The features can be fragment of given words, name entity recognition, length of the matching span or bias.

To find the highest-scoring between b and c, the authors use a semi-Markov model. Let $S(i)$ be the score of the first i words of the sentence $(w_{o:i})$. Then $S(i)$ will be calculated recurrently via the previous scores, and $S(0)$ will be initialized as zero. Obviously, $S(n)$ becomes the highest score. To obtain the best scoring concept labeling, JAMR uses a back-pointers method, similar to the idea of the conventional Viterbi algorithm [61].

The second stage of this algorithm is relation identification, which decides the edge label among the concept subgraph fragments assigned in the previous stage. The authors consider this problem like a graph-based techniques for non-projective dependency parser problem. While the dependency parser aims to find the maximum-scoring tree over words from the sentence, the relation identifier tries to find the maximum-scoring among subgraphs (MSCG) that preserve concept fragments from the previous stage.

A later success in alignment-based parsing is the work of Lyu et al. [8] in 2018. They introduce a neural parser which considers alignments as latent variables within a joint probabilistic model of concepts, relations and alignments. In their system, every graph node is assumed to be aligned to a word in a sentence: each concept is predicted based on the corresponding RNN state. Similarly, graph edges (or relations between nodes) are predicted based on the representations of concepts and aligned words. As alignments are latent, exact inference requires marginalizing over latent alignments, which is infeasible. Instead they use variational inference, specifically the variational autoencoding framework.

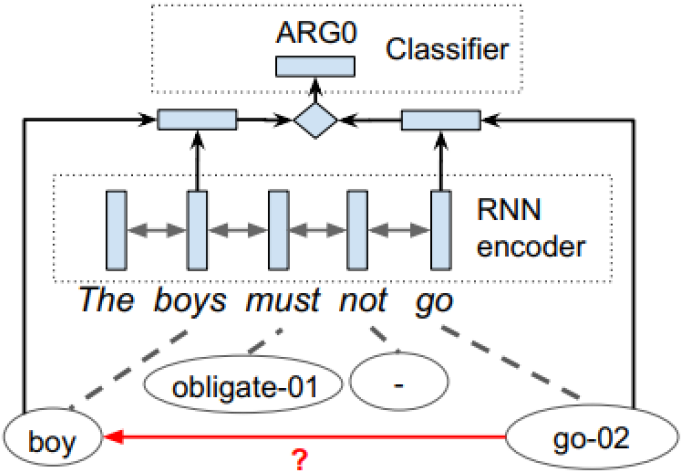


Figure 4.2.3: Relation identifier: predicting the relation between two nodes: *boy* and *go-02* relying on the two concepts and their corresponding RNN states

The author 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 injective alignment is necessary for two reasons. Firstly, it allows them to treat the concept identification task as sequence tagging at the test time. For every word they simply predict the corresponding concept or give the value NULL to signify that no concept should be generated at this position. Secondly, Gumbel-Sinkhorn can only work under this assumption. This constraint, though often appropriate, is problematic for certain AMR constructions (e.g., named entities). In order to deal with these cases, they re-categorized AMR concepts in a similar way to previous works. They obtained 73.1 Smatch score, a great improvement in the year of 2018.

4.2.2 Grammar-based parsing

After the success of Flanigan et al., Wang et al. [59] introduced another approach, namely **CAMR**. This parser employs a greedy strategy to select among a set of shift-reduce actions (transition rules) performed on the generated dependency graph from the input sentence. The authors first use a dependency parser to create the dependency tree for the input sentence, then transform the dependency tree to an AMR graph by applying some transition rules. This method takes advantages of the greate achievements in dependency parsing, with the available of a much larger training set than AMR parsing. This grammar-based approach is also applied by a number of later researchers, e.g. Damonte et al. [21]. Brandt et al. [62], Goodman et al. [63] and Peng et al. [64] [65][66][11], obtaining competitive results. In Figure 4.2.4, we describe an example of the dependency tree and the AMR graph parsed from the same sentence *"Private rights must conform to the public welfare"* to see the similarity and differences between these two structures.

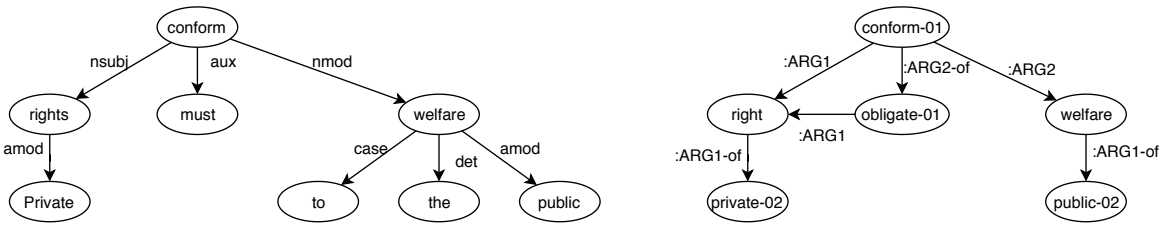


Figure 4.2.4: The dependency tree (on the left) and the AMR graph (on the right) corresponding to the sentence *"Private rights must conform to the public welfare"*

Unlike the dependency tree of a sentence, where each word corresponds to a single node in the tree, in AMR graph, some words become abstract concepts or relations while other words are even removed because they do not contribute to the whole meaning of the sentence. This difference causes some challenges for aligning words in the sentence and concepts in the graph. In order to learn the transition from the dependency tree to AMR graph, the authors adopt the algorithm from JAMR parser to produce the alignment. This heuristics based aligner that can align the words in a sentence and concepts in its AMR with a 0.9 F1 score (but there are some concepts in the AMR that cannot be aligned to any word in a sentence, e.g. *"have-org-role-91"*). They also construct a span graph to represent an AMR fragment that is aligned with the word tokens in a sentence. This span graph is a directed and labeled, denoted $G = (V, A)$, where $V = \{s_{i,j} | i, j \in (0, n) \text{ and } j > i \}$ is a set of nodes, and $A \subseteq V \times V$ is a set of arcs. Each node $s_{i,j}$ in G is assigned a concept label from concept set L_V

and is mapped with a continuous span $(w_i, ..., w_{j-1})$ in the sentence w. Each arc is also assigned a relation label from relation set L_A.

Basically, CAMR will perform three types of transition actions to transform the dependency tree into the AMR graph:

- actions performed when an edge is visited: NEXT-EDGE, SWAP, REATTACH, REPLACE-HEAD, REENTRANCE and MERGE
- actions performed when a node is visited: NEXT-NODE, DELETE-NODE
- actions used to infer abstract concepts in AMR that do not correspond to any word in the sentence: INFER

For the details of these transition rules, readers can refer to the original work [59], the Boosting version [60] and their paper at Semeval2016 contest [7]. One disadvantage of this method is that it limits the parsing ability to a single sentence, because the dependency tree can cover only the structure inside a sentence.

Follow this transition method, Damonte et al. [21] developed a model called **AMREager** that also parses the AMR graph based on a linear-time transition system, inspired by the ARCEAGER transition system for dependency tree parsing. Differs from CAMR which requires the full dependency tree to be obtained and then process the tree in a bottom-up manner, this parser process the sentence in the left-to-right direction. Specifically, AMREager defines a stack, a buffer and a configuration to perform the transition actions, which can be: *Shift, LArc, RArc* or *Reduce*. An example of transition process can be found in Figure 4.2.5. AMREager also uses the alignment obtained from JAMR aligner to map indices from the sentence to AMR graph fragments. Although the result in Smatch score is still lower than CAMR and JAMR by a small margin, AMREager obtains best results on several subscores such as Name Entity Recognition and Negation. The authors also proposed a test suite to evaluate F-score on the specific subtasks, which is widely used by later AMR parsing models.

action	stack	buffer	edges
-	[○]	[the,boy,and,the,girl]	{}
Shift	[○]	[boy,and,the,girl]	{}
Shift	[○, *boy*]	[and,the,girl]	{}
Shift	[○, *boy*, *and*]	[the,girl]	{}
LArc	[○, *and*]	[the,girl]	$\{\langle$*and*,:*op1*,*boy*$\rangle\} = A_1$
RArc	[○, *and*]	[the,girl]	$A_1 \cup \{\langle$○,:*top*,*and*$\rangle\} = A_2$
Shift	[○, *and*]	[girl]	A_2
Shift	[○, *and*, *girl*]	[]	A_2
RArc	[○, *and*, *girl*]	[]	$A_2 \cup \{\langle$*and*,:*op2*,*girl*$\rangle\} = A_3$
Reduce	[○, *and*]	[]	A_3
Reduce	[○]	[]	A_3

Figure 4.2.5: Transition rules when parsing the sentence *"The boy and the girl."*

The idea of transition-based parsing is simple but effective. Researchers often adopt this method to build the baseline system for AMR parsing in other languages than

English. Some examples can be figured out: A rule-based AMR parser for Portuguese by Rarael et al. [67], Transition-based Chinese AMR Parsing by Wang et al. [26], Cross-lingual AMR Parsing by Damonte et al. [27].

Grammar-based or transition-based approach has been investigated by combining with deep neural network and obtained some improvement. For instance, Ge et al. [68] modeled source syntax and semantics into neural seq2seq AMR parsing, Liu et al. [66] used a transition-based parser to tune their aligner via picking the alignment that leads to the highest scored achievable AMR graph, Naseem et al. [11] enriched the Stack LSTM transition-based AMR parser by augmenting training with Policy Learning and rewarding the Smatch score of sampled graphs. All of these extensions lead to an increasing of accuracy for parsing. However, source code for their models are still not available yet. In our experiments, we choose CAMR [59] [3] and AMREager [21] [4] to analyze the text.

4.2.3 Neural-based parsing

Recent advances in deep neural networks, with the great achievement of encoder-decoder architecture, lead to the rising of neural supervised learning models in AMR parsing. To adopt the encoder-decoder architecture, they attempt to linearize the AMR in the Penman notation format to a sequences of text, at word-level [1][49][69] [70] or at character-level [71]. After a linearization process, the parsing task can be considered as a translation task, which transforms a sentence into an AMR-like sequence. Later, Zhange et al. [72] proposed an attention-based model that treats AMR parsing as sequence-to-graph transduction. In this work, we choose **NeuralAMR** (word-level linearization) [1] and **Ch-AMR** (character-level linearization) [71] and Seq2Graph [72] to run our experiments.

Table 4.2.1: The AMR-like sequence obtained from linearizing the graph corresponding to the sentence *"How Long are We Going to Tolerate Japan?"*, which we extract from the dataset LDC2017T10

(t / tolerate-01 :ARG0 (w / we) :ARG1 (c / country :wiki "Japan" :name (n / name :op1 "Japan")) :duration (a / amr-unknown))	(tolerate-01 :ARG0 (we) :ARG1 (country :name (name :op1 "Japan")) :duration (amr-unknown))

Given an AMR graph represented in Penman notation format, NeuralAMR preprocesses the graph through a series of steps: linearization, anonymization, and further modifications. These steps aim to reduce the complexity of the linearized sequences and to address sparsity from certain out of vocabulary tokens, such as named entities and quantities (Table 4.2.1). By this preprocessing method, NeuralAMR takes advantage of encoder-decoder architecture by using a stack bidirectional LSTM network to encode the input sequence and a stacked LSTM network to decode from the hidden states produced by the encoder. The output string of the model is converted back to AMR

[3] `https://github.com/c-amr/camr`

[4] `https://github.com/mdtux89/amr-eager`

format by a delinearization algorithm to produce the final parsing result. Since this approach requires a huge amount of labeled data, NeuralAMR uses a paired training procedure to bootstrap a high-quality AMR parser from millions of unlabeled sentences extracted from the Gigarword corpus [73]. With this extra dataset, the parsing result increases significantly, from ***0.55*** to ***0.62*** in Smatch score. However, one disadvantage of this linearization method is to keep the semantic relation among nodes in the original graph. In the example shown in Table 4.2.1, the distance between two nodes: *"tolerate-01"* and *"amr-unknown"*, which are directly connected in the graph, becomes 10 (tokens) in the linearized sequence, as shown in Table 4.2.1. This distance can be even larger in long sentences with complicated structure, which can cause unexpected noises in the learning data distribution.

Private *JJ* rights *NNS* must *MD* conform *VB* to *TO* the *DT* public *NN* welfare *NN*
(obligate-01 :ARG1 (right-05 :ARG1-of (private-02)) :ARG2 (conform-01 :ARG1 (right-05) :ARG2 (welfare :ARG1-of (public-02))))

Figure 4.2.6: Sentence and AMR linearization in Ch-AMR

In 2017, Noord and Bos [71] introduced another approach in linearization which transforms the AMR graph to the character-level sequence. Their method removes all variables from the AMRs and duplicates co-referring nodes. After tokenizing the input sentence to sequence of characters, their model attaches the part-of-speech (POS) tag of the original tokens to provide more linguistic information to the decoder. We show an example of such preprocessed AMR in Figure 4.2.6, where the original sentence is extracted from our legal dataset JCivilCode: *"Private rights must conform to the public welfare"* (the original corresponding AMR will be presented in the later section of this chapter). It can be seen that this preprocessing method causes loss of information, since the variables of the original AMR cannot be recovered perfectly. To tackle this limitation, the authors proposed an approach to restore the co-referring nodes to the final output. Specifically, all wikification relations present in AMRs in the training set are also removed and restored in a post-processing step. They use OpenNMT [5], an open source neural machine translation system, to build their bidirectional LSTM model with general attention. This model archives a better result comparing to word-level preprocessing method in NeuralAMR, with **0.71** Smatch score reported in their paper.

Lastly, Zhang et al. introduced a different way to handle reentrancy, and proposed an attention-based model that treats AMR parsing as sequence-to-graph transduction. They considered the AMR tree with indexed nodes as the prediction target, then their approach to parsing is formalized as a two-stage process: node prediction and edge prediction. They designed their model with two main modules: an extended pointer-generator network for node prediction; and a deep biaffine classifier for edge prediction. The two modules correspond to the two-stage process for AMR parsing, and they are jointly learned during training. Figure 4.2.7 shows the overview architecture of their model. With the powerful pretrained language model BERT as a part of the input embedding, this sequence to graph model obtained state-of-the-art result for

[5] `https://opennmt.net/`

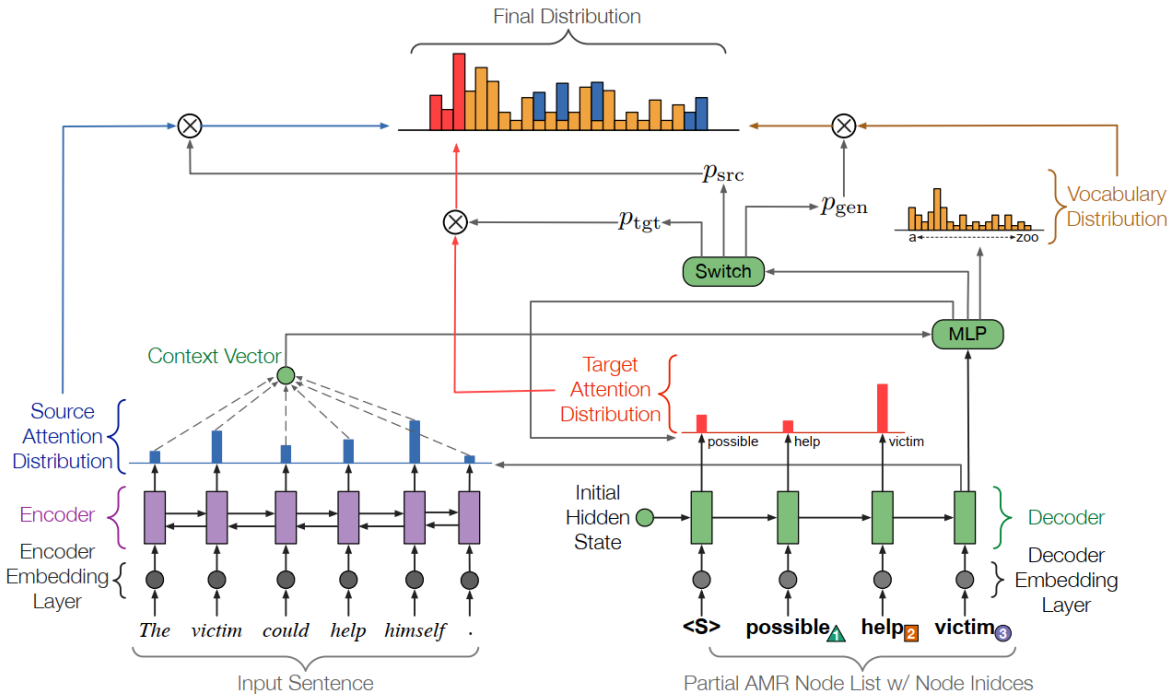


Figure 4.2.7: Sequence to graph transduction model

AMR parsing in 2019. The best reported Smatch score is 76.3 for the latest version LDC2017T10 dataset.

4.3 Experiments

4.3.1 Datasets

In our experiments, we mainly use two datasets: JCivilCode and LDC2017T10. The first one, which is first introduced in Lai *et al.* work in 2017 [42], is revised carefully with some modifications and additions. Table 4.3.1 shows some statistics of this dataset after our revision.

Table 4.3.1: Statistic for our dataset JCivilCode

Total number of samples	128
Average sentence length	31
Maximum sentence length	107
Average number of graph nodes	28
Maximum number of graph nodes	96
Vocabulary size	796
Number of tokens	4.042

As we mentioned in section 4.1, one of the major difficulties in analyzing legal documents is dealing with long sentences. In our experiments, we also aim to assess the performances of the seven models with various ranges of length of the sentence. Since the current legal dataset JCivilCode is still small to have a full evaluation, we use sen-

tences extracted from the well-known LDC2017T10 dataset, which consists of nearly 40,000 sentences in the news domain. We divide the test set of LDC2017T10 into four subsets LDC-20, LDC-20-30, LDC-30-40, LDC-40 with the lengths of the sentences in range 0-20, 21-30, 31-40 and greater than 40 words, respectively. Samples containing sub-sentences inside (which are annotated *"multi-sentence"* by the annotators) are excluded from the subsets. This exclusion guaranteed a fair comparison among the seven parsers since CAMR is unable to parse a text containing multiple sentences.

Table 4.3.2: Statistic of dataset LDC2015E86 and LDC2017T10 with our subsets division

Dataset	Total	Train	Dev	Test
LDC2015E86	19,572	16,833	1,368	1,371
LDC2017T10	39,260	36,521	1,368	1,371
LDC-20	-	-	-	694
LDC-20-30	-	-	-	284
LDC-30-40	-	-	-	143
LDC-40	-	-	-	82

In Table 4.3.2, we provide some statistics about the dataset LDC2017T10 and its subset after our division. The first row of the table show the number of samples in the previous version, namely LDC2015E86, which is commonly used to train several parsers as reported in their corresponding papers. English sentences in these two datasets are collected from web blogs, TV program transcriptions and forums in general domain. Each sample in these datasets includes a pair of sentence and the corresponding AMR graph, annotated by a group of expert from Linguistic Data Consortium (LDC).

4.3.2 Metrics for Evaluation

We mainly use Smatch score [20] to evaluate the parsing performance. Given the parsed graphs from each parser and the gold annotation graphs in the Penman format, Smatch first tries to find the best alignments between the variable names for each pair of graphs and then calculates precision, recall and F1 of the concepts and relations.Smatch score computes the F-score on the overall of two given graphs. To have a deeper analysis of parsing performance, we use the test-suite introduced by Damonte et al. [21] which calculates F-score on various perspectives, as mentioned in Chapter 2. In our legal experiment with the JCivilcode dataset, we do not include the Wikification and Name Entity subscores, since there are no Wiki concepts included in this dataset, and the number of existing named entities is small to evaluate.

4.3.3 Experimental Results and Discussions

To evaluate the performance of different parsing strategies on legal text, we conduct experiments on seven models that already provided their source codes: JAMR [2] and LtGraph [8] for alignment-based, CAMR [59] and AMR-Eager [21] for grammar-based, NeuralAMR [1], Ch-AMR [71] and Seq2Graph [72] for neural-based. While JAMR, CAMR and AMR-Eager were trained with the LDC2015E86 dataset only (the older version of LDC2017T10), NeuralAMR and Ch-AMR initialized the parser by LDC2015E86 and then used an extra corpus of 2 millions sentences extracted from a

free text corpus Gigaword [73] to train the complete models. LtGraph and Seq2Graph were not trained by this extra corpus, but they used the latest version of LDC data (LDC2017T10) [6].

Table 4.3.3: Smatch scores on the divided subsets of LDC2017T10

Method	Model	LDC-20	LDC-20-30	LDC-30-40	LDC-40
Alignment-based	JAMR	0.71	0.68	0.66	0.65
	LtGraph	**0.74**	0.73	0.72	0.68
Grammar-based	CAMR	0.66	0.62	0.60	0.59
	AMR-Eager	0.69	0.64	0.62	0.62
Neural-based	NeuralAMR	0.65	0.59	0.56	0.54
	Ch-AMR	0.45	0.43	0.42	0.40
	Seq2Graph	0.72	**0.76**	**0.75**	**0.71**

Table 4.3.4: Smatch scores and sub-scores on the dataset JCivilcode

Method	Model	Smatch	Unl	WSD	Neg	Con	Ree	SRL
Alignment-based	JAMR	0.45	0.50	0.47	0.23	0.59	0.32	0.43
	LtGraph	**0.53**	**0.57**	**0.55**	**0.47**	**0.67**	**0.38**	**0.52**
Grammar-based	CAMR	0.48	0.56	0.50	0.16	0.63	0.35	0.47
	AMREager	0.43	0.53	0.45	0.32	0.62	0.31	0.41
Neural-based	NeuralAMR	0.39	0.46	0.40	0.35	0.52	0.29	0.40
	Ch-AMR	0.28	0.37	0.28	0.19	0.35	0.22	0.28
	Seq2Graph	0.46	0.54	0.47	0.31	0.62	0.34	0.45

Parsing results are summarized in Table 4.3.3 (LDC2017T10 long sentences experiments) and Table 4.3.4 (JCivilCode experiments). Overall, the Smatch score of all the parsers on JCivilCode is still lower than on LDC2017T10 by a large margin (0.2 Smatch score on average). This downgrade is predictable since all the models were trained in general domain and test in a different one.

In LDC long sentences experiments, Seq2Graph remains the best parser in almost ranges of sentence length, except from the shortest one that witnesses the excellence of LtGraph. Other neural-based methods still produce low-quality AMRs, despite the huge amount of training data. The oldest parser JAMR got competitive results comparing to other recent methods. In this experiment, grammar-based methods does not obtain high score overall, but they perform consistently when the input sentence become longer. One of the reason for this consistent performance is that they are capable of recover the syntactic structure of the sentence through its dependency tree.

In the legal experiment, LtGraph outperforms other methods with a score of 0.53. All the subscores of LtGraph remain the first rank. More specifically, in the subscore of Negation and SRL, there are big gaps between this method and the others. Despite the best performance in general domain, Seq2Graph stands at third position, even lower than CAMR and only competitive with JAMR. Looking at the detail subscore, it can be figured out that all the models struggle with detecting the negation parts and reentrancy concepts in the input sentence. The highest Smatch subscores for these aspect are only 0.47 and 0.38, respectively.

[6]We keep the original trained models without retrained on the new dataset LDC2017T10

4.3.4 Error Analysis

We analyze some common errors in parsing outputs of legal sentences, which we divided by four main types: Incorrect concept identification (e.g. *case-03* vs. *case*, Missing concept (e.g. missing node *obligate-01*), Incorrect relation identification (e.g. *:arg1* vs. *:arg3* and Missing attribute (e.g. missing *:polarity* - in the negative clause). We use the script provided along with the JAMR parser to collect these statistical information. We provide the error statistics in Figure 4.3.1 and the examples provided in Table 5.4.3.

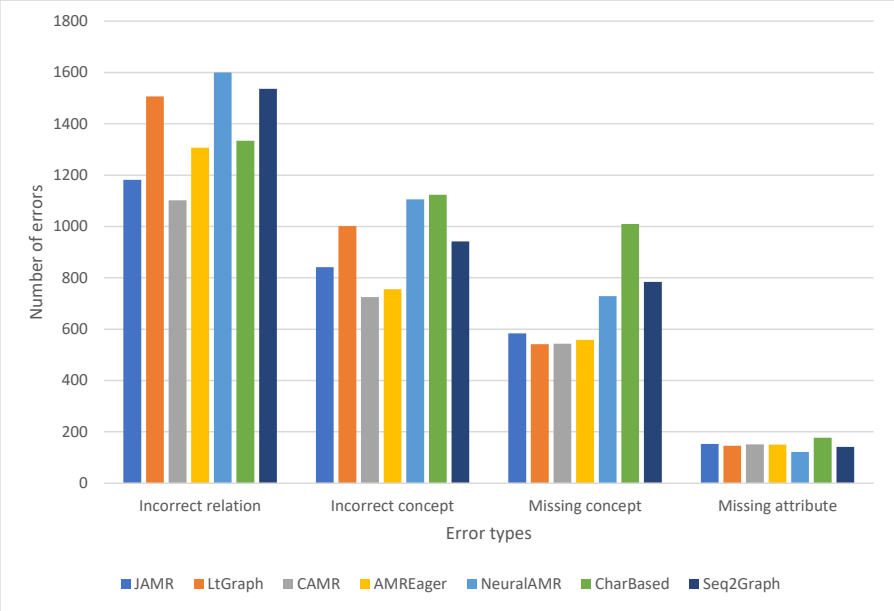


Figure 4.3.1: Statistic about common parsing errors

One of the most common errors that alignment-based and grammar-based produce is concept missing and incorrect relation assignment, often related to *modal verbs* (e.g. *"may", "can", "must"*. In general domains, these words do not contribute a lot to the sentence meaning, thus the verbs following these modal verbs are often labeled as the root of a graph or sub-graph in AMR as they express more important meaning. However, in legal text, modal verbs play a crucial role in a sentence and decide whether an action is permitted or not. Statistically, the word "may" appears in 29%, the word "must" appears in 19% of samples in our dataset. As shown in the example 1 in Table 5.4.3, only LtGraph and NeuralAMR are capable of identifying the concept *"obligate-01"* while other models totally ignore it.

Table 4.3.5: Example of common error types: **Incorrect concept identification** - **Missing concept** - **Incorrect relation identification** - **Missing attribute**

Example	*Private rights must conform to the public welfare* (1)	*No abuse of rights is permitted* (2)
Gold annotation	(o / obligate-01 :ARG1 (r / right-05 :ARG1-of (p / private-02)) :ARG2 (c / conform-01 :ARG1 r :ARG2 (w / welfare :ARG1-of (p2 / public-02))))	(p / permit-01 :polarity - :ARG1 (a / abuse-01 :ARG1 (r / right-05)))
JAMR	(c / conform-01 :ARG1 (r / **right** :ARG1-of (p2 / **private-03**)) :ARG2 (w / welfare **:domain-of** (p / **public**))) (missing concept **"obligate-01"**)	(p / permit-01 :ARG1 (a / abuse-01 :ARG1 (r / **right**)) :polarity -)
LtGraph	(o2 / obligate-01 :ARG1 (r1 / right-05 :ARG1-of (p0 / **private-03**) **:ARG1-of c3)** :ARG2 (c3 / conform-01 :ARG2 (w5 / welfare :ARG1-of (p4 / public-02))))	(p3 / permit-01 :ARG1 (a1 / abuse-01 :ARG1 (r2 / right-05) :polarity -))
CAMR	(x2 / right-05 :ARG1-of (x1 / **private-03**) :ARG1-of (x4 / conform-01 :ARG2 (x8 / welfare **:mod** (x7 / **public**)))) (missing concept **"obligate-01"**)	(x6 / permit-01 :ARG1 (x2 / abuse-01 :ARG1 (x4 / **right**))) (missing attribute **":polarity -"**)
AMR-Eager	(v3 / conform-01 :ARG1 (v2 / **right** :ARG1-of (v1 / **private-03**)) :ARG2 (v5 / welfare **:mod** (v4 / **public**))) (missing concept **"obligate-01"**)	(v3 / permit-01 :ARG1 (v1 / abuse-01 polarity - :ARG1 (v2 / **right**)))
Neural-AMR	(o / obligate-01 :arg2 (r / **rule-out-02** :arg0 (r2 / **right** :arg1-of (p / **private-03**)) :arg1 (w / welfare :mod (p2 / public))))	(p / permit-01 :polarity - :arg1 (a / **abuse-02** :arg1 (r / **right**)))

Table 4.3.5 – continued from previous page

Example	*Private rights must conform to the public welfare* (1)	*No abuse of rights is permitted* (2)
Ch-AMR	(vv1conform-01 / conform-01 :ARG1 (vv1person / **person** :ARG1-of (vv1private-03 / **private-03**)) :ARG2 (vv1welfare / welfare :ARG1-of **vv1**)) (missing concept **"right-05", "public-02", "obligate-01"**)	(vv3permit-01 / permit-01 :ARG1 (vv3no-abuse / **no-abuse**)) (missing concept **"right-05"**)

Another challenge in analyzing text in the legal domain is the complexity of logical relations. Regarding this aspect, all the parsers in the experiments still produce lots of errors when facing negative clause. The reason almost comes from the fact that many negations are encoded with morphology (e.g., such as *"un-"* prefix in *"unless"* or *"unable"*), thus cause difficulties for parsing models to detect. In Table 5.4.3, we show the outputs from all the parsers for the source sentence: *"No abuse of rights is permitted"*. It can be seen that NeuralAMR and JAMR succeeded in converting the negation to *:polarity* -, while AMREager and LtGraph did not put this edge in the exact position (However, in this case, it is still acceptable because the meaning of the sentence is not changed a lot). Char-AMR labels this negation as a phrase *"no-abuse"* and CAMR even skips this important information. In term of Smatch score, all the parsers still obtain very low results. The highest negation F-score is LtGraph with 0.47, and the lowest is CAMR with only 0.16 score.

4.4 Chapter Summary

In this chapter, we provided a brief survey on three main approaches in AMR parsing, where we choose seven parsers to conducte experiments. All the seven parsers are tested on the legal dataset JCivilCode and general domain dataset LDC2017T10 with different ranges of sentence length. We evaluate the parsing outputs by Smatch score in several aspects including overall F-score and sub-score on specific tasks. Experimental results showed the domain adaptation abilites of seven models in the legal domain. All the performances decreased by approximately 0.2 on the Smatch score, comparing to their parsing results for text in general domain. This result shows the difficulties in AMR parsing for this legal field.

Currently, our legal dataset JCivilcode is still too small, with only 128 samples, to have a complete evaluation. In order to improve the domain adaptation ability for current parsing techniques, we need to enlarge our dataset, in both the quantity and the quality aspects. This work is challenging since it requires efforts from experts in both linguistic and legal domain. We also plan to adopt semi-supervised or unsupervised learning methods in the future work. Specifically, we plan to investigate the Teacher-Student approach, in which the teacher model is trained by the labeled data first, then this model is used to annotate the unlabeled data. We select a subset of this unlabeled data by filtering out the predictions using a threshold for the score, then combine this subset with the original labeled data to train the student model. Recently, this

approach got significant improvements by adding noise to the student model during its training [74][75] (NoisyStudent), obtaining new state of the art results on image classification tasks.

Chapter 5

Legal Text Generation from Abstract Meaning Representation

5.1 Introduction

As mentioned in the previous chapter, several neural-based approaches have been proposed to tackle the AMR-to-text generation problem by leveraging a large amount of silver data [14], [1]. Most of them rely on the encoder-decoder architecture with attention mechanism, resulting in high scores in the BLEU metric on a benchmark AMR dataset. Despite acceptable performance on general domain text, those generating models struggle in dealing with the legal domain, where the sentences are complicated structure and contain domain-specific terms.

We give an example of AMR annotation in Figure 5.1.1, where the nodes (e.g. *"enjoy-01", "right-05"*, ...) represent concepts, and the edges (e.g. *":arg0", ":condition"*, ...) represent relations between those concepts. This example is extracted from our legal dataset JCivilCode [43]. When conducting a survey with different generation methods, we figure out that lots of out-of-vocabulary words are generated, and almost the negation and conditional sentences are generated incorrectly.

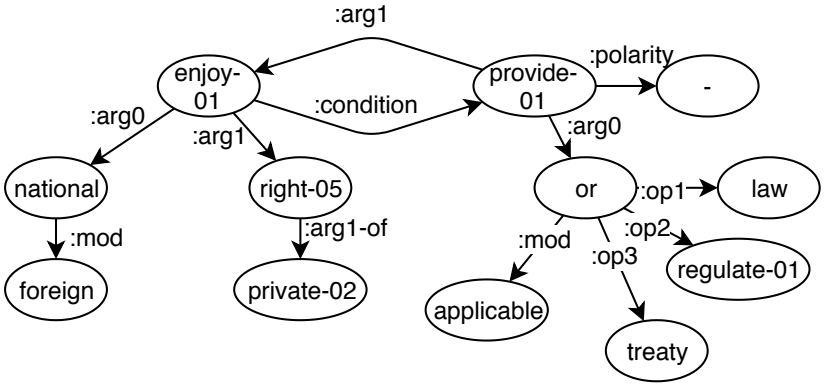


Figure 5.1.1: AMR graph for the sentence *"Unless otherwise provided by applicable laws, regulations or treaties, foreign nationals shall enjoy private rights"*.

In this chapter, we propose a our domain adaptation method applied to the training phase and decoding phase of the baseline graph-to-sequence model to improve the generation quality. Specifically, in the training process, we constrain the encoder-decoder model by a controllable variable to avoid the repetitive token generating as

well as guiding the model to recognize the negation and conditional sentences more appropriately. After training, the model is fine-tune with a silver-annotated dataset generated from a civil code in the English version. Moreover, we adopt weighted decoding with a modified beam-search algorithm to avoid out-of-vocabulary words. The model is tested using our legal dataset JCivilCode, showing improvement over the baseline model.

5.2 Preliminaries

5.2.1 Deep learning approaches in AMR-to-text Generation

Given an AMR graph $G = (V; E)$, where V and E are the sets of vertices and edges, respectively, the goal is to generate a sentence $W = (w_1, w_2, ..., w_n)$ expresses the meaning in the graph, where w_i are words in the vocabulary. Since first introduced as a shared task at SemEval-2017 [76], several approaches have been proposed to tackle this generation problem, with a dominance of deep learning models. Konstas et al. [1] linearized AMR graphs, then adopt an encoder-decoder model to translate these string-like objects into the natural language (NeuralAMR). Song et al. [14] modified the encoder side architecture to capture the graph structure data more properly. This resulted in a graph-to-sequence model (Graph2Seq) capable of generating well-written text, obtaining the state of the art BLEU score in this generation problem in 2018.

However, these models still struggle when dealing with legal text, i.e. Graph2Seq obtains 9.86 BLEU score on JCivilCode [43], comparing to the score of 32.0 on LDC2017 test set. In our paper, we rely on Graph2Seq to build our baseline model.

5.2.2 The baseline model

As mentioned before, we adopt the graph-to-sequence model in [14] as our baseline. With a given AMR graph $G = (V; E)$, each vertex v_i is represented by a hidden state vector h_i, initializing by the word embedding of that node. The graph state g is defined as the set of h_i. The information exchange between a current vertex v_i and all incoming nodes and outgoing nodes are captured through a sequence of state transitions $g_0, g_1, ..., g_k$. The encoder side used a long short term memory (LSTM) network to perform this graph state transition. By using this mechanism, information containing in each vertex is propagated to all of its neighboring vertices after each transition step. After k steps, each vertex hidden state contains the information of a large context, including the vertex's ancestors, descendants, and siblings, where k is the maximum graph diameter in the dataset (we choose $k = 9$ in our experiments). The decoder side is also an LSTM network incorporated with a copy mechanism [50] to deal with decoding objects like name entities, numbers, and dates. The detail computation in each step can be found in Chapter 3.

5.3 Legal AMR generation

5.3.1 Conditional training

Conditional Training (CT) [77] is a method to learn an encoder-decoder model $P(y|g, z)$, where z is a discrete control variable and g is the AMR graph. We de-

sign z by annotating every (g, y) pair in the training set with the attribute we wish to control, e.g. the number of nodes in the graph, the length of the linearized graph, or whether g contains negation or not. This attribute value will be determined during training, depend on each training sample. We use an embedding value with size v to represent the control variable z, where v is a hyper-parameter.

There are several possible ways to condition the sequence-to-sequence model on z – for example, append z to the end of the input sequence, or use z as the START symbol for the decoder. We find it most effective to concatenate z to the decoder's input on every step.

The objective function of training is given by the cross-entropy loss:

$$loss_{CT} = -\frac{1}{T}\sum_{t=1}^{T} logP(y|g, z, y_1, ..., y_{t-1}).$$

Where $y = y_1, ..., y_n$ is the expected output that the model has to produce.

Parameters of the model are initialized when training with the benchmark general domain dataset, then finetuning with the silver legal dataset to optimize $loss_{CT}$.

5.3.2 Decoding in legal style

To enhance the probability of generating words with certain features, we adopt Weighted Decoding (WD) that was introduced by Ghazvininejad et al. [78]. On the t_{th} step of decoding, the generated hypothesis $y_{<t} = y_1, \ ... \ , y_{t-1}$ is expanded by computing the score for each possible next word w in the vocabulary by the formula:

$$score(w, y_{<t}; g) = score(y_{<t}; g) + logP_{LSTM}(w|y_{<t}, g) + \sum_i w_i * f_i(w; y_{<t}, g).$$

In which $\log P_{LSTM}(w|y_{<t}, g)$ is the log probability of the word w calculated by the bi-LSTM network, score$(y_{<t}; g)$ is the accumulated score of the generated words in the hypothesis $y_{<t}$ and $f_i(w; y_{<t}, g)$ are decoding features with the corresponding weights w_i. There can be multiple features f_i to control multiple attributes, and the weights w_i are hyperparameters. A decoding feature $f_i(w; y_{<t}, g)$ assigns a real value to the word w. The feature can be continuous (e.g. the unigram probability of w) or discrete (e.g. the length of w in characters). A positive weight w_i increases the probability of words w that scores highly with respect to f_i and vice versa.

Another problem of generating text from legal AMR is the out of vocabulary (OOV) tokens, where lots of words in the legal domain are not included in well-known word embedding, e.g. Word2Vec or Glove. We collect the vocabulary of three datasets: a benchmark dataset in general domains and two datasets obtained from Vietnamese and Japanese civil code. Our observation shows that more than 30% of the words in these vocabulary sets do not appear in Glove [51].

To deal with this OOV problem, we modified the beam search decoding algorithm. Specifically, after collecting an extra-vocabulary from the legal finetune set, we assign a binary feature to each word w in the test set representing whether w is in the legal vocab or not. This increases the probability of words in the legal vocabulary to be selected to the *top-k* generation, where k is the beam size.

5.4 Experiments and Results

5.4.1 Dataset Preparation

In our experiments, we use three datasets: (i) the benchmark dataset LDC2017T10 for training the baseline model, (ii) silver data generated from a Vietnamese Civil Code for fine-tuning the model, and (iii) the JCivilCode dataset [1] [43] for testing the performance. We train our model without using silver annotated data sampled from external corpora (like NeuralAMR and Graph2Seq extracted external sentences from Gigaword and used a parser to automatically annotate their AMRs).

Table 5.4.1: Statistics of the three dataset used in our experiments

Dataset	LDC2017T10	VN Civil Code	JCivilCode
Number of samples	36.521	3,073	128
Vocabulary size	29,943	3,026	778
Number of words out of vocab	4,453	602	270
:condition edge	1,794	190	69
Negation	10,947	356	57

In dataset (i) we use the linearization and anonymization algorithm provided by Song et al. [14] and Konstas et al. [1]. For dataset (ii), the silver data is obtained by performing two best parsers for legal text: JAMR [2] and CAMR [7] as suggested by Vu et al. [43]. Each sample sentence in the corpus will provide two AMR graphs, this also enlarges the dataset for finetuning our models. The statistics of these datasets can be found in Table 5.4.1.

5.4.2 Results and Analysis

We evaluate our models mainly by BLEU score [44] and METEOR score [45]. We also report the number of OOV words generated from each model.

Table 5.4.2: Generation results in BLEU score, METEOR score and number of OOV generated. The baseline Graph2Seq is trained on benchmark dataset only. The next four lines show our proposed modifications, with and without finetuning data. The last two lines are the results of two best pretrained models with extra corpus.

Model	BLEU	METEOR	OOV
Baseline Graph2Seq	5.50	16.78	135
Graph2Seq + CT	6.82	17.42	112
Graph2Seq + Finetune data	8.31	17.74	145
Graph2Seq + Finetune data + Conditional Training	**8.56**	**18.61**	143
Graph2Seq + Finetune data + LD	8.42	17.98	57
Graph2Seq + Finetune data + CT + LD	8.43	18.04	**57**
Graph2Seq Pretrained on 2M Gigaword corpus	9.31	21.38	29
NeuralAMR Pretrained on 2M Gigaword corpus	9.07	20.55	35

From Table 5.4.2, it can be observed that our both proposed modifications improve the performance of text generation. Compared to the baseline model, Conditional

[1] `https://github.com/sinhvtr/legal_amr`

Training (CT) helps increase the BLEU score and METEOR score by a little margin (1.32 and 0.76, respectively). While Legal Decoding (LD) helps reduce the OOV rate significantly (from 143 tokens to 57 tokens). However, combining both two techniques does not result in the best score overall, where BLEU and METEOR score decrease slightly after LD, since this algorithm sometimes eliminates non-legal words from the *top-k* space.

Our experimental results also confirm the important role of in-domain data. After finetuning with the legal dataset VNCivilCode, we obtain 2.81 and 0.96 improvement on BLEU and METEOR score, respectively. We also report the results of text generation from several pre-trained neural models, i.e. Graph2Seq [14] and NeuralAMR [1]. When comparing to those pre-trained models, with a huge amount of data (2 millions samples extracted from Gigaword corpus), our proposed modifications still got lower results by a small margin.

Table 5.4.3: Output comparison with an example from JCivilCode dataset

Gold data Unless otherwise provided by applicable laws, regulations or treaties, foreign nationals shall enjoy private rights.
Baseline model the foreign national enjoy a private right not if the applicable law or economic treaty
Baseline model + finetune data when it is not provided for by law or the treaties to enjoy the private rights , the foreign national shall have the enjoy private rights .
Baseline model + finetune data + CT the foreign national will enjoy private rights without providing applicable regulate regulate or treaty
Baseline model + finetune data + CT + LD when a foreign national enjoys the private right , if not provided for by law or the provisions of law or the provisions of law .
Graph2Seq Pretrained on 2M Gigaword foreign nationals will enjoy private rights while there are no laws or regulations if the or or without the regulations are provided .

To have a closer look, we provide some output examples for each model in Table 5.4.3. All the models still generate low-quality sentences, with grammatical errors and repetitive words. The baseline model trained without any legal data provides an out-domain word that does not appear in the source AMR graph. After finetuning, the sentences generated become longer but not so meaningful except for the output of CT model, which includes almost correct information. LD, as mentioned earlier, could help reduce the OOV rate overall, but may cause some words or fragments missing and repetitive.

5.5 Chapter Summary

In this chapter, we figure out the difficulties of AMR generation in the legal domain, where the logical structure is complicated and lots of domain-specific terms are not in the well-known vocabulary. To tackle these difficulties, we propose two modifications

to the training and decoding phases of the neural graph to sequence model. We then finetune our models using a silver annotated dataset in legal domain. The experimental results prove the effectiveness of our method over the baseline model. Despite the improvement, all models in our experiments still generate low-quality text from legal AMR. The best-reported score is only 9.31 for BLEU and 21.38 for METEOR, leaving a challenge for research in this domain. This also confirms the important role of in-domain training data. Currently, the number of labeled data in AMR format is still small comparing to other sematic role labeling datasets, even in general domain. In the future work, we plan to discover some data augmentation techniques, such as data recombination [79], to obtain more high-quality legal data.

Chapter 6

Conclusion and Future Work

In this thesis, we study the problem of AMR Parsing and Generation for both general domain text and legal domain adaptation. In this chapter, we summarize our study, point out main findings, and discuss some directions for future work.

6.1 Conclusion and Main Findings

The objective of this thesis is to improve the performance of parsing and generation by using deep learning techniques, aiming to apply to text in the legal domain. In order to achieve our objective, we first provide necessary knowledge in Chapters 1 and 2, including the definition of our parsing and generation tasks, data preparation, evaluation procedure and recent techniques in deep neural networks. We present our main work in three next chapters.

- Chapter 3: in this chapter, we propose our AMR-to-text generation model with the incorporating of the self-attention mechanism. Motivated by the domination of the Transformer architecture in various Natural Language Processing tasks, we adopt its core component - the self-attention - to build our generation models. We investigate in both the sequence-to-sequence and graph-to-sequence strategies. In the sequence-to-sequence model, we incorporate the self-attention layer followed by a position-wise feed forward layer to the encoder, and the self-attention layer followed by a vanilla attention, followed by a position-wise feed forward layer in the decoder. With this method, we obtain an improvement of 2.9 BLEU score and 2.6 METEOR score comparing to the baseline model. In the graph-to-sequence model, we apply self-attention layers on the graph hidden state, derived through a LSTM state transition process. However, the generation result decreases by a little margin, i.e. 1.3 BLEU score and 1.9 METEOR score.

- Chapter 4: in this chapter, we provide a survey of different methods in AMR parsing. We category them into three main approaches: alignment-based, grammar-based and machine translation based. We show their performances when analyzing legal documents by conducting experiments of seven parsers, chosen from the three main approaches, on our annotated legal dataset JCivilCode and the benchmark dataset LDC2017T10 in various ranges of sentence length. We figure out major challenges, i.e. negation detection, modal verb annotation, long dependency parsing. These difficulties cause a decreasing by approximately 0.2 on the Smatch score of the seven models we choose.

- Chapter 5: beside the challenges we figure out in chapter 4, we observe that text generated from AMR using current deep learning models usually become awkward with lots of "out of vocabulary" tokens. This is due to the fact that lots of words in the legal domain are not included in well-known word embeddings, e.g. Word2Vec or Glove, which are commonly used in neural generation models. To tackle this problem, we propose our method in modifying the training and the decoding phase of the encoder-decoder AMR generation model to have a better text realization. We then finetune our models using a silver annotated dataset in legal domain. Our model is tested using our legal dataset JCivilCode, showing an improvement compared to the baseline model.

Limitations: Despite some improvements and findings, our work still remains drawbacks.

- In chapter 3, we succeed in improving the performance of the sequence-to-sequence generating model with the self-attention mechanism. But incorporating this mechanism into the graph-to-sequence model does not produce better results compared to the baseline. This means our proposed method could not benefit from the graph structure in the source AMR.
- We publish the first AMR dataset in legal domain, namely JCivilCode. This annotated dataset is used in chapter 5 and 4 for evaluating our proposed methods in legal domain adaptation. However, the amount of data is still small, with only 128 samples available, to show a complete evaluation of both parsing and generation performances.
- We figure out the challenges in parsing legal text to AMR, i.e. the complicated logical structure and long sentences, but we have not proposed the solution for tackling them.

6.2 Future Work

Based on promising results of this thesis, we figure out some directions for future:

- Data augmentation: Training data play an important role in training deep neural network models for parsing and generation. Currently, the number of labeled data in AMR format is still small comparing to other sematic role labeling datasets, even in general domain. To tackle this problem, we need to discover some data augmentation techniques, semi-supervised or unsupervised leaning strategies. Specifically, we plan to investigate the Teacher-Student approach, in which the teacher model is trained by the labeled data first, then this model is used to annotate the unlabeled data. We select a subset of this unlabeled data by filtering out the predictions using a threshold for the score, then combine this subset with the original labeled data to train the student model. Recently, this approach got significant improvements by adding noise to the student model during its training [74][75], obtaining new state of the art results on image classification tasks.

- Long sentence dependency is still the major challenge for AMR parsing, in both general and legal domain. We aim to tackle this problem by discovering recent technique namely SPk Languages to explore the characteristics of long-distance dependencies [80]. In detail, we try using Strictly k-Piecewise languages to generate AMR datasets with various properties. From this, we can compute the characteristics of the long distance dependencies in these datasets and analyze the impact of factors such as the length of the long dependencies, the vocabulary size, or the dataset size.

- Logical complexity: as mentioned in Chapter 4 and 5, this complexity causes lots of errors for both AMR parsing and generation models. Several research have been proposed to generate logical forms from text and entities graph [81][82]. We plan to explore these works to build a logical attention mechanism to capture these information more effetively.

- AMR applications: we plan to apply AMR in several downstream problems such as Legal Question Answering. Despite the capablity of AMR in expressing the "who is doing what to whom" aspects, there are not many works investigate the application of AMR in question answering (QA), especially for legal domain. Previous work built a QA system for the Little Prince dataset (nearly 1.500 samples of AMR) and use CAMR [7] to parse a given question to AMR. Then the authors performed a graph matching algorithm to find the best candidate answers in the dataset. They relied on the heuristic rules of AMR annotation, e.g. *who* corresponding to the relation *:arg0* in the graph, or *where* corresponding to *:location*. We expect our legal dataset can be enlarged and contributes to a similar legal QA system.

Bibliography

[1] Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 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, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1014.

[2] Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 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, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/P14-1134.

[3] Omri Abend and Ari Rappoport. 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, August 2013. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/P13-1023`.

[4] Valerio Basile, Johan Bos, Kilian Evang, and Noortje Venhuizen. 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, May 2012. European Language Resources Association (ELRA). URL `http://www.lrec-conf.org/proceedings/lrec2012/pdf/534_Paper.pdf`.

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

[6] Zi Lin and Nianwen Xue. Parsing meaning representations: Is easier always better? In *Proceedings of the First International Workshop on Designing Meaning Representations*, pages 34–43, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-3304.

[7] Chuan Wang, Sameer Pradhan, Xiaoman Pan, Heng Ji, and Nianwen Xue. 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, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/S16-1181.

[8] Chunchuan Lyu and Ivan Titov. 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, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1037.

[9] Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 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, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1048.

[10] Deng Cai and Wai Lam. Core semantic first: A top-down approach for AMR parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3790–3800, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1393. URL `https://www.aclweb.org/anthology/D19-1393`.

[11] Tahira Naseem, Abhishek Shah, Hui Wan, Radu Florian, Salim Roukos, and Miguel Ballesteros. 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, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1451.

[12] Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. Generation from abstract meaning representation using tree transducers. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 731–739, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1087.

[13] Nima Pourdamghani, Kevin Knight, and Ulf Hermjakob. Generating English from abstract meaning representations. In *Proceedings of the 9th International Natural Language Generation conference*, pages 21–25, Edinburgh, UK, September 5-8 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-6603.

[14] Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. A graph-to-sequence model for AMR-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1616–1626, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1150.

[15] Kris Cao and Stephen Clark. Factorising AMR generation through syntax. 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 2157–2163, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1223.

[16] Marco Damonte and Shay B. Cohen. Structural neural encoders for AMR-to-text generation. In *Proceedings of the 2019 Conference of the North Amer-*

ican Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 3649–3658, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1366.

[17] Emma Manning. A partially rule-based approach to AMR generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 61–70, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-3009.

[18] Leonardo F. R. Ribeiro, Claire Gardent, and Iryna Gurevych. Enhancing AMR-to-text generation with dual graph representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3174–3185, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1314. URL `https://www.aclweb.org/anthology/D19-1314`.

[19] Jie Zhu, Junhui Li, Muhua Zhu, Longhua Qian, Min Zhang, and Guodong Zhou. Modeling graph structure in transformer for better AMR-to-text generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5462–5471, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1548. URL `https://www.aclweb.org/anthology/D19-1548`.

[20] Shu Cai and Kevin Knight. 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, August 2013. Association for Computational Linguistics.

[21] Shay B. Cohen Marco Damonte and Giorgio Satta. An incremental parser for abstract meaning representation. In *Proceedings of European Chapter of the ACL (EACL)*, 2017.

[22] Linfeng Song and Daniel Gildea. SemBleu: A robust metric for AMR parsing evaluation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4547–4552, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1446.

[23] Ha Linh and Huyen Nguyen. A case study on meaning representation for vietnamese. In *Proceedings of the First International Workshop on Designing Meaning Representations*, pages 148–153, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-3317.

[24] Bin Li, Yuan Wen, Weiguang Qu, Lijun Bu, and Nianwen Xue. Annotating the little prince with Chinese AMRs. In *Proceedings of the 10th Linguistic Annotation Workshop held in conjunction with ACL 2016 (LAW-X 2016)*, pages 7–15, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-1702.

[25] Noelia Migueles-Abraira, Rodrigo Agerri, and Arantza Diaz de Ilarraza. Annotating abstract meaning representations for Spanish. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, Miyazaki, Japan, May 2018. European Languages Resources Association (ELRA).

[26] Chuan Wang, Bin Li, and Nianwen Xue. Transition-based Chinese AMR parsing. 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 247–252, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2040. URL `https://www.aclweb.org/anthology/N18-2040`.

[27] Marco Damonte and Shay B. Cohen. 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, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1104.

[28] Rafael Anchiêta and Thiago Pardo. 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, May 2018. European Languages Resources Association (ELRA).

[29] Hyonsu Choe, Jiyoon Han, Hyejin Park, and Hansaem Kim. Copula and case-stacking annotations for Korean AMR. In *Proceedings of the First International Workshop on Designing Meaning Representations*, pages 128–135, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-3314.

[30] Linfeng Song, Daniel Gildea, Yue Zhang, Zhiguo Wang, and Jinsong Su. Semantic neural machine translation using AMR. *Transactions of the Association for Computational Linguistics*, 7:19–31, March 2019. doi: 10.1162/tacl_a_00252.

[31] Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. 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, May–June 2015. Association for Computational Linguistics. doi: 10.3115/v1/N15-1114.

[32] Sudha Rao, Daniel Marcu, Kevin Knight, and Hal Daumé III. Biomedical event extraction using abstract meaning representation. In *BioNLP 2017*, pages 126–135, Vancouver, Canada,, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-2315. URL `https://www.aclweb.org/anthology/W17-2315`.

[33] Fuad Issa, Marco Damonte, Shay B. Cohen, Xiaohui Yan, and Yi Chang. 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, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1041.

[34] Hardy Hardy and Andreas Vlachos. 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, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1086.

[35] Kexin Liao, Logan Lebanoff, and Fei Liu. 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, August 2018. Association for Computational Linguistics.

[36] Shibhansh Dohare, Harish Karnick, and Vivek Gupta. Text summarization using abstract meaning representation. *arXiv preprint arXiv:1706.01678*, 2017.

[37] Arindam Mitra and Chitta Baral. Addressing a question answering challenge by combining statistical methods with inductive rule learning and reasoning. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[38] Mrinmaya Sachan and Eric Xing. Machine comprehension using rich semantic representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 486–492, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-2079.

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

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[41] Robert T. Kasper. 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*, 1989. URL `https://www.aclweb.org/anthology/H89-1022`.

[42] Lai Dac Viet, Vu Trong Sinh, Nguyen Le Minh, and Ken Satoh. Convamr: Abstract meaning representation parsing for legal document. *arXiv preprint arXiv:1711.06141*, 2017.

[43] Vu Trong Sinh and Nguyen Le Minh. An empirical evaluation of amr parsing for legal documents. In *New Frontiers in Artificial Intelligence JSAI-isAI 2018 Workshops, JURISIN, AI-Biz, SKL, LENLS, IDAA*, pages 131–145, Yokohama, Japan, 2018.

[44] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL `https://www.aclweb.org/anthology/P02-1040`.

[45] Michael Denkowski and Alon Lavie. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*, 2014.

[46] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014. URL `http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf`.

[47] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1166. URL `https://www.aclweb.org/anthology/D15-1166`.

[48] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016. URL `http://arxiv.org/abs/1607.06450`.

[49] Xiaochang Peng, Chuan Wang, Daniel Gildea, and Nianwen Xue. 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, Valencia, Spain, April 2017. Association for Computational Linguistics.

[50] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640. Association for Computational Linguistics, 2016. doi: 10.18653/v1/P16-1154. URL `http://aclweb.org/anthology/P16-1154`.

[51] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics, 2014. doi: 10.3115/v1/D14-1162. URL `http://aclweb.org/anthology/D14-1162`.

[52] Linfeng Song, Yue Zhang, Xiaochang Peng, Zhiguo Wang, and Daniel Gildea. AMR-to-text generation as a traveling salesman problem. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2084–2089, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1224.

[53] Linfeng Song, Xiaochang Peng, Yue Zhang, Zhiguo Wang, and Daniel Gildea. AMR-to-text generation with synchronous node replacement grammar. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 7–13, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-2002.

[54] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. 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 464–468. Association for Computational Linguistics, 2018. doi: 10.18653/v1/N18-2074. URL `http://aclweb.org/anthology/N18-2074`.

[55] Takuya Katayama. Legal engineering-an engineering approach to laws in e-society age. In *Proc. of the 1st Intl. Workshop on JURISIN*, 2007.

[56] Makoto Nakamura, Shunsuke Nobuoka, and Akira Shimazu. Towards translation of legal sentences into logical forms. In *Annual Conference of the Japanese Society for Artificial Intelligence*, pages 349–362. Springer, 2007.

[57] María Navas-Loro, Ken Satoh, and Víctor Rodríguez-Doncel. Contractframes: Bridging the gap between natural language and logics in contract law. In Kazuhiro Kojima, Maki Sakamoto, Koji Mineshima, and Ken Satoh, editors, *New Frontiers in Artificial Intelligence*, pages 101–114, Cham, 2019. Springer International Publishing. ISBN 978-3-030-31605-1.

[58] Junsheng Zhou, Feiyu Xu, Hans Uszkoreit, Weiguang Qu, Ran Li, and Yanhui Gu. AMR parsing with an incremental joint model. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 680–689, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1065.

[59] Chuan Wang, Nianwen Xue, and Sameer Pradhan. 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, May–June 2015. Association for Computational Linguistics. doi: 10.3115/v1/N15-1040.

[60] Chuan Wang, Nianwen Xue, and Sameer Pradhan. 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, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-2141.

[61] L. R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, Feb 1989. ISSN 0018-9219. doi: 10.1109/5.18626.

[62] Lauritz Brandt, David Grimm, Mengfei Zhou, and Yannick Versley. ICL-HD at SemEval-2016 task 8: Meaning representation parsing - augmenting AMR parsing with a preposition semantic role labeling neural network. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1160–1166, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/S16-1179.

[63] James Goodman, Andreas Vlachos, and Jason Naradowsky. UCL+Sheffield at SemEval-2016 task 8: Imitation learning for AMR parsing with an alpha-bound. In

Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), pages 1167–1172, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/S16-1180.

[64] Giorgio Satta Xiaochang Peng, Daniel Gildea. AMR Parsing With Cache Transition Systems. In *AAAI*, 2018.

[65] Zhijiang Guo and Wei Lu. 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, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1198.

[66] Yijia Liu, Wanxiang Che, Bo Zheng, Bing Qin, and Ting Liu. An AMR aligner tuned by transition-based parser. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2422–2430, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1264.

[67] Rafael Torres Anchiêta and Thiago Alexandre Salgueiro Pardo. A rule-based amr parser for portuguese. In Guillermo R. Simari, Eduardo Fermé, Flabio Gutiérrez Segura, and José Antonio Rodríguez Melquiades, editors, *Advances in Artificial Intelligence - IBERAMIA 2018*, pages 341–353, Cham, 2018. Springer International Publishing. ISBN 978-3-030-03928-8.

[68] DongLai Ge, Junhui Li, Muhua Zhu, and Shoushan Li. Modeling source syntax and semantics for neural amr parsing. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4975–4981. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/691. URL `https://doi.org/10.24963/ijcai.2019/691`.

[69] Miguel Ballesteros and Yaser Al-Onaizan. AMR parsing using stack-LSTMs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1269–1275, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1130.

[70] William Foland and James H. Martin. 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, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1043.

[71] Rik van Noord and Johan Bos. Neural semantic parsing by character-based translation: Experiments with abstract meaning representations. *Computational Linguistics in the Netherlands Journal*, 7:93–108, 2017.

[72] Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 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, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1009.

[73] Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. Annotated Gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*, pages 95–100, Montréal, Canada, June 2012. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W12-3018.

[74] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V. Le. Self-training with noisy student improves imagenet classification, 2019.

[75] Shayne Shaw, Maciej Pajak, Aneta Lisowska, Sotirios A Tsaftaris, and Alison Q O'Neil. Teacher-student chain for efficient semi-supervised histology image classification, 2020.

[76] Jonathan May and Jay Priyadarshi. SemEval-2017 task 9: Abstract meaning representation parsing and generation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 536–545, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2090.

[77] Angela Fan, David Grangier, and Michael Auli. Controllable abstractive summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 45–54, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-2706. URL https://www.aclweb.org/anthology/W18-2706.

[78] Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. Hafez: an interactive poetry generation system. In *Proceedings of ACL 2017, System Demonstrations*, pages 43–48, Vancouver, Canada, July 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P17-4008.

[79] Robin Jia and Percy Liang. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1002. URL https://www.aclweb.org/anthology/P16-1002.

[80] Abhijit Mahalunkar and John Kelleher. Multi-element long distance dependencies: Using SPk languages to explore the characteristics of long-distance dependencies. In *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*, pages 34–43, Florence, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-3904. URL https://www.aclweb.org/anthology/W19-3904.

[81] Peter Shaw, Philip Massey, Angelica Chen, Francesco Piccinno, and Yasemin Altun. Generating logical forms from graph representations of text and entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 95–106, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1010. URL https://www.aclweb.org/anthology/P19-1010.

[82] Li Dong and Mirella Lapata. 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, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1004. URL `https://www.aclweb.org/anthology/P16-1004`.

Publications

[1] Vu Trong Sinh and Le Minh Nguyen. A study on self-attention mechanism for amr-to-text generation. In *Natural Language Processing and Information Systems - 24th International Conference on Applications of Natural Language to Information Systems, NLDB 2019, Salford, UK, June 26-28, 2019, Proceedings*, pages 321–328, 2019. doi: 10.1007/978-3-030-23281-8_27. URL `https://doi.org/10.1007/978-3-030-23281-8_27`.

[2] Vu Trong Sinh and Nguyen Le Minh. An empirical evaluation of amr parsing for legal documents. In *New Frontiers in Artificial Intelligence JSAI-isAI 2018 Workshops, JURISIN, AI-Biz, SKL, LENLS, IDAA*, pages 131–145, Yokohama, Japan, 2018.

[3] Lai Dac Viet, Vu Trong Sinh, Nguyen Le Minh, and Ken Satoh. Convamr: Abstract meaning representation parsing for legal document. *arXiv preprint arXiv:1711.06141*, 2017.

[4] Vu Trong Sinh, Nguyen Le Minh, and Satoh Ken. Legal text generation from abstract meaning representation. In *Proceedings of the 32nd International Conference on Legal Knowledge and Information Systems*, Madrid, Spain, 2019.

[5] Vu Trong Sinh, Nguyen Le Minh, and Satoh Ken. Abstract meaning representation for legal documents. *submitted to the Journal of Artificial Intelligence and Law - Springer Nature*, 2020.