Artificial Intelligence and Law (2022) 30:221–243
https://doi.org/10.1007/s10506-021-09292-6

ORIGINAL RESEARCH

Abstract meaning representation for legal documents: an empirical research on a human-annotated dataset

Sinh Trong Vu[1] · Minh Le Nguyen[1] · Ken Satoh[2]

Accepted: 24 June 2021 / Published online: 7 July 2021

Abstract

Natural language processing techniques contribute more and more in analyzing legal documents recently, which supports the implementation of laws and rules using computers. Previous approaches in representing a legal sentence often based on logical patterns that illustrate the relations between concepts in the sentence, often consist of multiple words. Those representations cause the lack of semantic information at the word level. In our work, we aim to tackle such shortcomings by representing legal texts in the form of abstract meaning representation (AMR), a graph-based semantic representation that gains lots of polarity in NLP community recently. We present our study in AMR Parsing (producing AMR from natural language) and AMR-to-text Generation (producing natural language from AMR) specifically for legal domain. We also introduce JCivilCode, a human-annotated legal AMR dataset which was created and verified by a group of linguistic and legal experts. We conduct an empirical evaluation of various approaches in parsing and generating AMR on our own dataset and show the current challenges. Based on our observation, we propose our domain adaptation method applying in the training phase and decoding phase of a neural AMR-to-text generation model. Our method improves the quality of text generated from AMR graph compared to the baseline model. (This work is extended from our two previous papers: "An Empirical Evaluation of AMR Parsing for Legal Documents", published in the Twelfth International Workshop on Juris-informatics (JURISIN) 2018; and "Legal Text Generation from Abstract Meaning Representation", published in the 32nd International Conference on Legal Knowledge and Information Systems (JURIX) 2019.).

✉ Sinh Trong Vu
sinhvtr@jaist.ac.jp

✉ Minh Le Nguyen
nguyenml@jaist.ac.jp

Ken Satoh
ksatoh@nii.ac.jp

1 Japan Advanced Institute of Science and Technology, Ishikawa 923-1292, Japan

2 National Institute of Informatics, Tokyo 100-0003, Japan

Keywords Abstract meaning representation · Deep neural network · Legal document

1 Introduction

Since proposed by Katayama (2007), Legal Engineering has grown to become a new trend in computer science. Legal Engineering studies the methodology and application of information science and software engineering in the legal domain. There are two important goals of Legal Engineering: (i) Help experts make complete and consistent laws and (ii) Design a law information system. 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 computers to automatically process the text in legal documents, where NLP techniques get involved. However, most of the methods for representing a legal sentence are based on logical patterns (Nakamura et al. 2007; Navas-Loro et al. 2019), skipping the semantic information among the words in the sentence. That motivates us to investigate a more semantic approach for representing texts in this legal domain.

In NLP community, semantic representation of text plays an important role and receives growing attention in the past few years. Many semantic schemes have been proposed, such as Groningen Meaning Bank (Basile et al. 2012), Abstract Meaning Representation (Banarescu et al. 2013), Universal Conceptual Cognitive Annotation (Abend and Rappoport 2013). In which, AMR has shown a great potential and gained popularity in computational linguistics (Lin and Xue 2019; Zhu et al. 2019; Cao and Clark 2019; Song et al. 2019).

AMR can be applied as an intermediate meaning representation for solving various tasks in NLP, such as machine comprehension (Sachan and Xing 2016), machine translation (Jones et al. 2012; Song et al. 2019), text summarization (Liu et al. 2015; Hardy and Vlachos 2018; Liao et al. 2018; Dohare et al. 2017), question answering (Mitra and Baral 2016), event extraction (Rao et al. 2017). For AMR to be applied in those tasks, the problem of AMR Parsing and AMR-to-text Generation are both important. Converting a natural language sentence into a right AMR graph guarantees the correct semantic understanding of that sentence. While generating text from a given AMR graph is important for downstream applications such as summarization and machine translation. Thus, many approaches have been proposed to tackle these two problems. Almost of them conduct experiments on a general domain dataset from Linguistic Data Consortium (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 (Fig. 1). This leads to lots of challenges in AMR parsing and generating comparing to the general domain. In our article, we aim to study those challenges by empirical experiments and tackle them by various techniques in domain adaptation. We use English as our main language, due to the completeness in the Propbank dictionary as well as the availability of a large amount of training data. Our contributions can be summarized as follow:

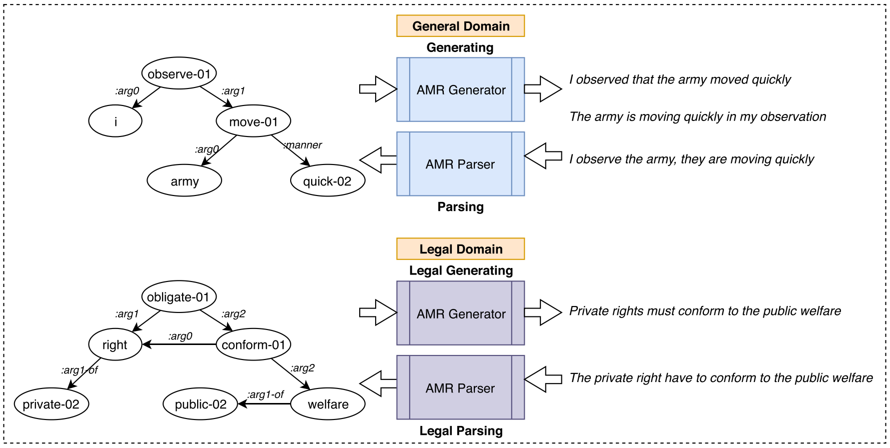


Fig. 1 AMR Parsing and Generation in the general domain and in the legal Domain

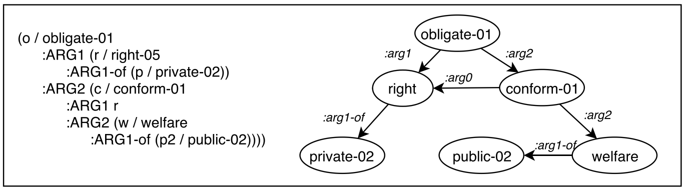
- We manually annotate a legal AMR dataset, extracted from Japanese Civil Code. This is the first AMR dataset in the legal domain, rather than popular datasets mainly taken from news, blog posts. Though the number of samples is still small, this dataset helps evaluate AMR parsing and generation model 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 limitations as well as open a room for improvements of current parsing techniques when applying in this non-trivial domain.
- We propose two modifications in the training and decoding phases of the neural graph to sequence AMR-to-text 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. Our source code is available at https://github.com/sinhvtr/amr_jurix

The rest of our article will be organized as follow: we provide some necessary background knowledge and current approaches to AMR parsing and generation in Sect. 2, we introduce the datasets used in our work and the evaluation metrics in Sect. 3, our experiments will be presented in Sect. 4. We propose our improvement in AMR-to-text generation for legal text in Sect. 5. Finally, we conclude our work in Sect. 6.

2 Preliminaries

2.1 Abstract meaning representation

Abstract Meaning Representation is a semantic representation language that encodes the meaning of a sentence as a rooted, directed, edge-labeled, leaf-labeled graph



```
(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))))
```

Fig. 2 AMR for the sentence *"Private rights must conform to the public welfare"*, in Penman notation and graph format

while abstracting away the surface forms in a sentence. Every vertex and edge of the graph are labeled according to the sense of the words in a sentence. AMR can be represented in PENMAN notation, for a human to read and write easily, or graph structure, for a computer to store in its memory, or decomposed into conjunctions of logical triples, for calculating the difference among AMRs. Figure 2 shows an example of AMR annotation for the sentence *"Private rights must conform to the public welfare"*, in Penman notation and graph format.

In AMRs, each node is named by an ID (variable). It contains the semantic concept, which can be a word (e.g. *boy*) or a PropBank frameset (e.g. *want-01*) or a special keyword. The keywords consist of entity type (e.g. *date-entity, ordinal-entity, percentage-entity*), quantities (e.g. *distance-quantity*), and logical conjunction(e.g. *and, or*). The edge between two vertices is labeled using more than 100 relations including frameset argument index (e.g. *":ARG0", ":ARG1 "*), semantic relations (e.g. *":location", ":name"*), relations for quantities (e.g. *:quant, :unit, :scale*), relations for date-entities, relations for listing (e.g. *:op1, :op2, :op3*). AMR also provides the inverse form of all relations by concatenating -of to the original relation (e.g. :location vs :location-of).

2.2 AMR parsing

The task of parsing a natural language text into an AMR graph faces a lot of challenges, such as word-sense disambiguation, semantic graph construction, data sparsity. To tackle these, many approaches have been proposed, which we group into three main categories: alignment-based, grammar-based and machine-translation-based (Fig. 3).

Alignment-based One of the pioneer AMR parsing solutions is **JAMR** introduced by Flanigan et al. (2014), which build a two-part algorithm that first identifies concepts with an automatic aligner and then identifies 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 provided a strong baseline in AMR parsing. Follow this approach, Zhou et al. (2016) extended the relation identification tasks with a component-wise beam search algorithm. Chunchuan and Titov (2018) 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 (Fig. 4).

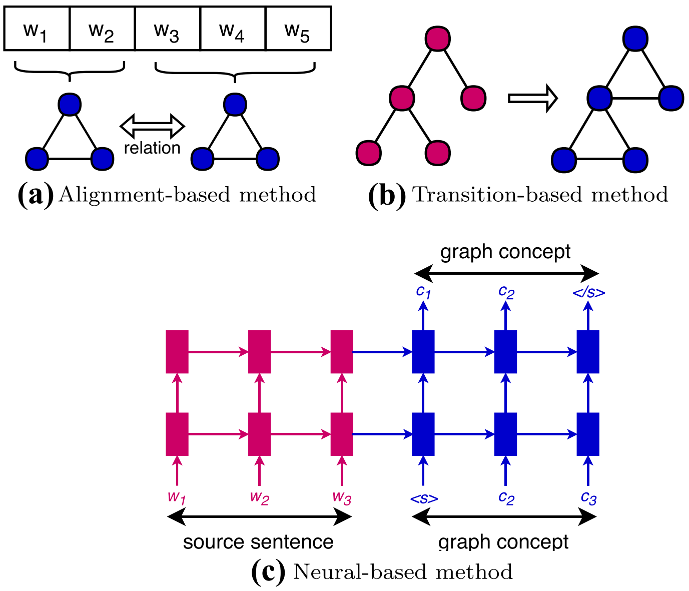
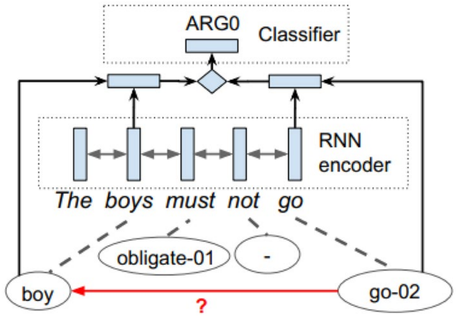


Fig. 3 Main approaches in AMR parsing

Fig. 4 Relation identification: predicting a relation between boy and go-02 relying on the two concepts and corresponding RNN states



Transition-based Wang et al. (2016) introduced a transition-based parser called **CAMR**. The authors first use a dependency parser to generate the dependency tree for the sentence, then transform the dependency tree to an AMR graph through some transition rules. This method takes advantages of the achievements in dependency parsing, with a training set much larger than the training set of AMR parsing. Damonte et al. (2017). Brandt et al. (2016), Goodman et al. (2016) and Peng et al. (2015) also applied the grammar-based algorithm in their works and obtained competitive results. Figure 5 shows an example of the dependency tree and the AMR graph parsed from the same sentence *"Private rights must conform to the public welfare"*.

Transition-based approach has been investigated by combining with deep neural network and obtained some improvement. For instance, Ge et al. (2019) modeled

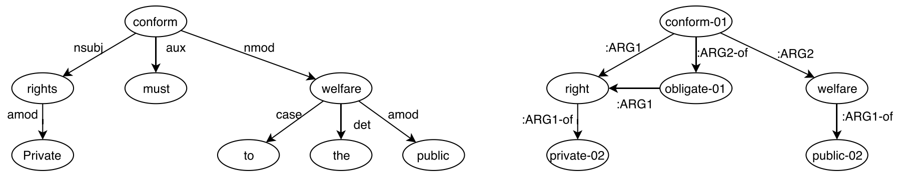


Fig. 5 Dependency tree and AMR graph generated from the sentence *"Private rights must conform to the public welfare"* (Wang et al. 2016)

source syntax and semantics into neural seq2seq AMR parsing, Liu et al. (2018) 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. (2019) 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 (Wang et al. 2015) and AMREager (Damonte et al. 2017) to analyze the text.

Neural-based Recently, with the achievement of the encoder-decoder architecture in deep neural networks, several supervised learning approaches have been proposed in order to deal with AMR parsing task. They attempt to linearize the AMR in Penman notation to sequences of text, at character-level (van Noord and Bos 2017) or at word-level (Konstas et al. 2017; Peng et al. 2017; Ballesteros and Al-Onaizan 2017; Foland and Martin 2017), so that the parsing task can be considered as a translation task, which transforms a sentence into an AMR-like sequence. Later, Zhange et al. (2019) proposed an attention-based model that treats AMR parsing as sequence-to-graph transduction (Fig. 6).

2.3 AMR-to-text generation

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). Pourdamghani et al. (2016) was the first one who applies machine translation approach in AMR-to-text generation. They converted AMR graphs, which were written in Penman notation form, to a sequence of text through a linearization process. With these pairs of linearized AMRs and corresponding sentences, they considered AMR-to-text generation task as a machine translation problem and implemented a phrase-based model to obtain the final text.

Following this approach, Konstas et al. (2017) proposed the first neural model for both the AMR parsing and AMR-to-text generation problem (NeuralAMR). The authors used an encoder-decoder model built upon a long short term memory (LSTM) neural network (Fig. 7. As this architecture required a large set of training data to achieve good results, Konstas et al. (2017) used their own AMR parser to automatically annotate millions of unlabeled sentences before training their main system; the obtained AMR graphs are then used as additional training data. To deal with the problem of data sparsity addressing in Peng el al. (2017),

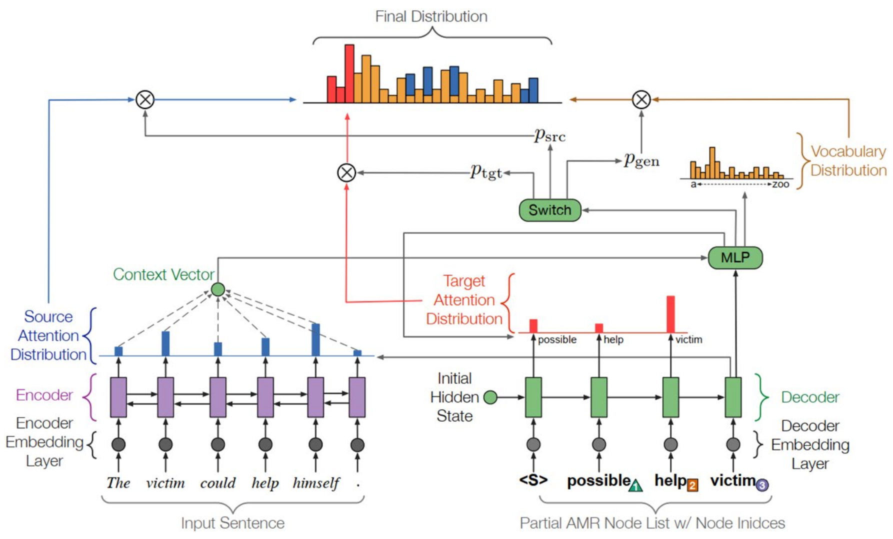


Fig. 6 Sequence to graph transduction model (Zhang et al. 2019)

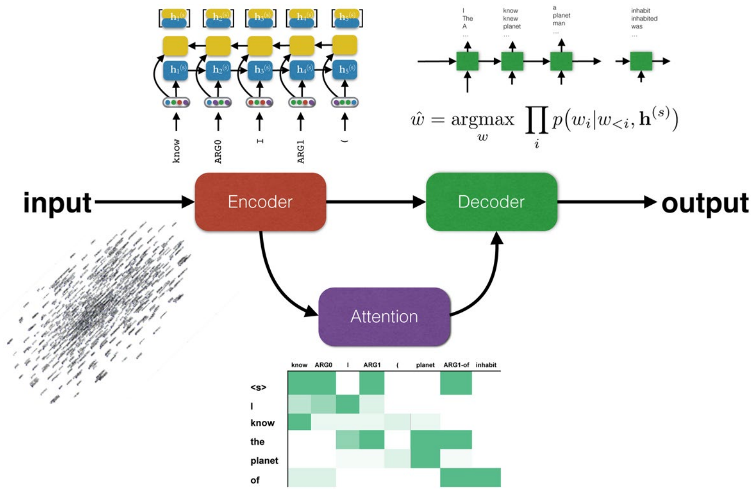

know
ARG0
I
ARG1
(
$\hat{w} = \text{argmax}_w \prod_i p(w_i|w_{<i}, \mathbf{h}^{(s)})$
input
Encoder
Decoder
output
Attention
know ARG0 I ARG1 (planet ARG1-of inhabit
<s>
I
know
the
planet
of

Fig. 7 Sequence to sequence model in NeuralAMR (Konstas et al. 2017)

NeuralAMR adopts an anonymization algorithm. In detail, they first replaced the subgraphs that represent open-class tokens (such as *"country :name name :op1 United :op2 States"*) with predefined placeholders (such as *loc_0*) before decoding and then recovered the corresponding surface tokens (such as "United States") after decoding. Cao and Clark (2019) factor the generation process leveraging syntactic information to improve the performance. However, they linearize both AMR and constituency graphs, which implies that important parts of the graphs cannot well be represented (e.g., coreference).

Different from the rule-based anonymization algorithm above, Song et al. (2018) incorporated a char-level LSTM over the characters of input tokens and a copy network (Gu et al. 2016) on top of the decoder side. This architecture also helps generate the named entities, dates and numbers effectively. Song *et al.* also proposed a novel graph to sequence model (Graph2Seq), in which the authors encoded the AMR graph with a bidirectional LSTM encoder, performing through a graph-state transition, rather than a transition over linear sequence. This graph encoder helped prevent information loss through the linearization process, especially when the graph becomes large. With the same amount of training data as NeuralAMR, this graph to sequence model achieved the state of the art result on a benchmark test set in the year of 2018.

In 2019, deep learning approaches continued to dominate in AMR-to-text generation problem. Damonte et al. (2019) provided a survey of various types of neural encoders. They investigate the extent to which reentrancies (nodes with multiple parents) have an impact on AMR-to-text generation by comparing graph encoders to tree encoders, where reentrancies are not preserved. They show that improvements in the treatment of reentrancies and long-range dependencies contribute to higher overall scores for graph encoders. Leonardo et al. (2019) proposed a dual graph representation that encodes different but complementary perspectives of the structural information contained in the AMR graph. The model learns parallel top-down and bottom-up representations of nodes capturing contrasting views of the graph. They also investigate the use of different node message passing strategies, employing different state-of-the-art graph encoders to compute node representations based on incoming and outgoing perspectives.

3 Dataset and evaluation

3.1 Dataset

Several datasets have been manually annotated for conducting research in AMR, such as the datasets released by Linguistic Data Consortium LDC2017T10 (39,269 sentences), the Little Prince dataset extracted from the novel by Antoine de Saint-Exupery (1562 sentences). However, there is no open AMR resource for any specific domain such as the legal document. That motivates us to create our first legal dataset for AMR called JCivilCode, extracted from the English version of 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

Table 1 Statistics of the three dataset used in our experiments

Dataset	LDC2017T10	VN Civil Code	JCivilCode
Number of samples	36.521	3073	128
Max sentence length	119	151	107
Average nodes number	13	17	28
Max nodes number	98	93	96
Average length	20	29	31
Vocabulary size	29,943	3026	778
Condition edge	1794	190	69

extract the first four chapters in Part I to annotate. The pre-processing consists of the following steps: gathering articles, removing all article prefixes and article IDs, then splitting the article into sentences. All the AMRs are annotated by two annotators to ensure the neutrality of evaluation.

The current version of JCivilCode consists of 128 samples. Since this amount of data is too small for training a neural model for any AMR-related task, we consider using an extra *silver-annotated* dataset, which means AMR dataset generated automatically and not verified by human experts. Following this strategy, we collect legal sentences from Vietnamese Civil Code (English version), then use two AMR parsers to create 3,073 AMR-sentence pairs. We call this dataset VNCivilCode. Although the quality level of this data is far from human-annotated standard, it is still useful in finetuning our AMR-to-text generation model, presented in Sect. 5. We shows some statistics of VNCivilCode dataset, JCvilCode dataset comparing to the benchmark LDC dataset in Table 1.

As we mentioned in Sect. 1, one of the main difficulty in analyzing legal documents is dealing with long sentences. In our experiments, we also would like to assess the performances of the five models with different lengths of the sentence. Since the current legal dataset is still small, we use extra sentences 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. We excluded the samples containing sub-sentences inside (annotated *"multi-sentence"* by the annotators). This exclusion guaranteed a fair comparison among the five parsers because CAMR is unable to analyze multiple sentences at the same time.

3.2 Evaluation

Parsing AMR parsers are evaluated mainly by Smatch score (Cai and Knight 2013). Given the parsed graphs and the gold graphs in the form of Penman annotations, Smatch first tries to find the best alignments between the variable names for each pair of graphs and it then computes precision, recall and F1 of the concepts and relations. In this paper, to test the performance of AMR parser on legal text, which contains

Table 2 SMATCH score between two annotations of JCivilCode

Metric	Score
Precision	91.18
Recall	93.35
F-score	92.25

sentences in complicated structures, we analyze the parsing results in a deeper measurement. Specifically, we use the test-suite introduced by Damonte et al. (2017), which assesses the parsing results on various sub-scores as follow:

- *Unlabeled (Unl)*: Smatch score computed on the predicted graphs after removing all edge labels (e.g., *:ARG0, :condition*)
- *No WSD (WSD)*: Smatch score while ignoring Propbank senses (e.g., *perform-01* vs *perform-02*)
- *Name Entity*: F-score on the named entity recognition (:name roles)
- *Wikification*: F-score on the wikification (:wiki roles)
- *Negation (Neg)*: F-score on the negation detection (:polarity roles)
- *Concepts (Con)*: F-score on the concept identification task
- *Reentrancies (Ree)*: Smatch computed on reentrant edges only
- *Semantic Role Labeling (SRL)*: Smatch computed on :ARG-i roles only

In our experiment with JCivilcode-1.0, we do not include the Wikification and Name Entity criteria since there are no Wiki concepts included in this dataset, and the number of existing named entities is small. We also use this Smatch metric to evaluate the agreement of annotations, by calculate the Smatch score between two annotations. The results are presented in Table 2. As can be seen, the similarity of two annotations is very high at 92.25 points of SMATCH score. The precision and recall are fairly equivalent. These results suggest that the dataset was built with high quality.

Generation 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-to-text generation performances, instead of other metrics used in image captioning or text summarization. In our experiments, we use BLEU (Papineni et al. 2002) and METEOR (Denkowski and Lavie 2014) for evaluating text generation.

Table 3 Smatch scores on LDC2017T10 (different ranges of length)

Method	Model	LDC-20	LDC-20-30	LDC-30-40	LDC-40
Alignment	JAMR	0.71	0.68	0.66	0.65
	LtGraph	**0.74**	0.73	0.72	0.68
Grammar	CAMR	0.66	0.62	0.60	0.59
	AMR-Eager	0.69	0.64	0.62	0.62
Neural	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**

4 Empirical evaluation of AMR parsing methods for legal documents

4.1 Experimental setup

To evaluate the performance of different parsing strategies on legal text, we conduct experiments on seven models that already provided their source codes: JAMR (Flanigan et al. 2014; LtGraph Lyu and Titov 2018) for alignment-based, CAMR (Wang et al. 2015; AMR-Eager Damonte et al. 2017) for grammar-based, NeuralAMR (Konstas et al. 2017), Ch-AMR (van Noord and Bos 2017) and Seq2Graph (Zhang et al. 2019) for neural-based. While JAMR, CAMR and AMR-Eager were trained with the LDC2015E86 dataset only (the older version of LDC2017T10 with only 13,051 samples), 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 (Napoles et al. 2012) 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)[1].

4.2 Results in smatch score

Parsing results are summarized in Table 3 (LDC2017T10 long sentences experiments) and Table 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

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

Table 4 Smatch scores and sub-scores on JCivilcode

Score	Method & Models						
	Alignment-based		Grammar-based		Neural-based		
	JAMR	LtGraph	CAMR	AMR-Eager	Neural-AMR	Ch-AMR	Seq2-Graph
Smatch	0.45	**0.53**	0.48	0.43	0.39	0.28	0.46
Unl	0.50	**0.57**	0.56	0.53	0.46	0.37	0.54
WSD	0.47	**0.55**	0.50	0.45	0.40	0.28	0.47
Neg	0.23	**0.47**	0.16	0.32	0.35	0.19	0.31
Con	0.59	**0.67**	0.63	0.62	0.52	0.35	0.62
Ree	0.32	**0.38**	0.35	0.31	0.29	0.22	0.34
SRL	0.43	**0.52**	0.47	0.41	0.40	0.28	0.45

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 sub-score 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 sub-score, 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.

4.3 Parsing error analysis

We analyze some common errors in parsing outputs of legal sentences, with the statistics given in Fig. 8 and the examples provided in Table 5. One of the most common errors in alignment-based and grammar-based performances is missing concept and relation related to *modal verbs*. In legal documents, modal verbs (e.g. the word "may" appears in 29%, the word "must" appears in 19% of samples in the dataset) play a crucial role in a sentence and decide whether an action is permitted or not. This differs from other domains, where these words do not often contribute a lot to the sentence meaning. As shown in example 1 in Table 5, only NeuralAMR is capable of identifying the concept *"obligate-01"* while other models totally ignore it.

Another challenge in parsing legal text is the logical complexity. In this aspect, all the parsers still show limitation when parsing negative clause. This is not too surprising as many negations are encoded with morphology (e.g., such as *"un- "* prefix in *"unless"* or *"unable"*) and cause difficulties for detection. In Table 5, we show an example of outputs from all the parsers for a sentence: *"No abuse of rights is permitted"*. NeuralAMR and JAMR succeeded in converting negation to *:polarity -*, AMREager did not put this edge in the exact position, but in this case, it does

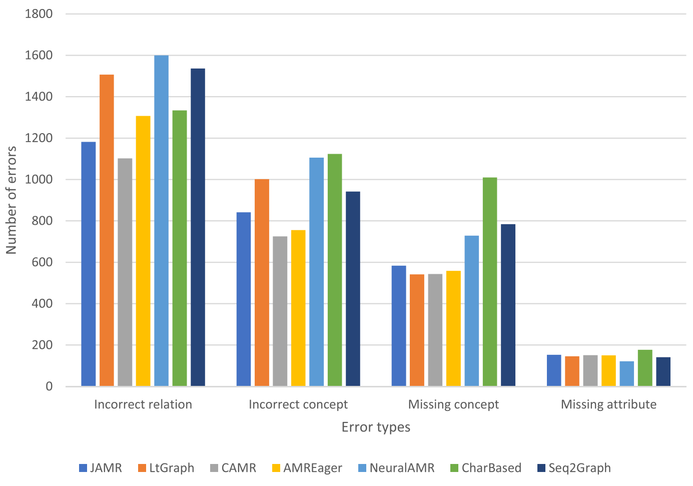


Fig. 8 Common error types statistics

not change the meaning of the sentence. CAMR performs even worse as it skips this important information.

To summarize this section, we conducted experiments of AMR parsing on the legal dataset JCivilCode and news domain dataset LDC2017T10 with different ranges of sentence length to observe the abilities of five different models. Experimental results showed the domain adaptation of five models for the legal domain and the performance decreased by approximately 0.2 on the Smatch score. This result shows difficulties in applying AMR parsing for analyzing legal documents.

5 Legal text generation from AMR

5.1 The baseline model

Dealing with the graph structure from AMR, we adopt the graph encoder in Song et al. (2018). For a graph $G = \{V, E\}$, We represent each node $v_i \in V$ by a hidden state vector h^i. The state of the graph can thus be represented as $g = \{h^i\}$. Information exchange between a current node v_i and all nodes connected to it are captured by a sequence of state transitions $\{g_0, g_1, ..., g_k\}$. In particular, 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 consists of a set of initial node states $h^j_0 = z_0$, where z_0 is a vector of all zeros.

A LSTM network is used to model the state transition process. In particular, the transition from g_{t-1} to g_t consists of a hidden state transition for each node. At each state transition step t, the model conducts direct communication between a node and all nodes that are directly connected to the node. 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.

Table 5 We highlight some common errors: $< \boldsymbol{i} > \boldsymbol{-Incorrectconcept}$; $< \boldsymbol{ii} > \boldsymbol{-Incorrectrelation}$; $< \boldsymbol{iii} > \boldsymbol{-Missingconcept}$; $< \boldsymbol{iv} > \boldsymbol{-Missingattribute}$

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 / $\boldsymbol{right} < \boldsymbol{i} >$:ARG1-of ($p2/\boldsymbol{private} - \mathbf{03} < \boldsymbol{i} >$)) :ARG2 (w / welfare $\boldsymbol{: domain - of} < \boldsymbol{ii} >$ (p / $\boldsymbol{public} < \boldsymbol{i} >$))) (missing concept }}$\boldsymbol{obligate} - \mathbf{01}'' < \boldsymbol{iii} >$)	(p / permit-01 :ARG1 (a / abuse-01 :ARG1 (r / $\boldsymbol{right} < \boldsymbol{i} >$)) :polarity -)
LtGraph	(o2 / obligate-01 :ARG1 (r1 / right-05 :ARG1-of (p0 / $\boldsymbol{private} - \mathbf{03} < \boldsymbol{i} >$) $\boldsymbol{: ARG1 - ofc3}$) $< \boldsymbol{ii} >$: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 / $\boldsymbol{private} - \mathbf{03} < \boldsymbol{i} >$) :ARG1-of (x4 / conform-01 :ARG2 (x8 / welfare $\boldsymbol{: mod} < \boldsymbol{ii} >$ (x7 / $\boldsymbol{public} < \boldsymbol{i} >$)))) (missing concept }}$\boldsymbol{obligate} - \mathbf{01}'' < \boldsymbol{iii} >$)	(x6 / permit-01 :ARG1 (x2 / abuse-01 :ARG1 (x4 / $\boldsymbol{right} < \boldsymbol{i} >$))) (missing attribute }} $\boldsymbol{: polarity-}'' < \boldsymbol{iv}$)
AMR-Eager	(v3 / conform-01 :ARG1 (v2 / $\boldsymbol{right} < \boldsymbol{i} >$:ARG1-of (v1 / $\boldsymbol{private} - \mathbf{03} < \boldsymbol{i} >$)) :ARG2 (v5 / welfare $\boldsymbol{: mod} < \boldsymbol{ii} >$ (v4 / $\boldsymbol{public} < \boldsymbol{i} >$))) (missing concept }}$\boldsymbol{obligate} - \mathbf{01}'' < \boldsymbol{iii} >$)	(v3 / permit-01 :ARG1 (v1 / abuse-01 polarity - :ARG1 (v2 / $\boldsymbol{right} < \boldsymbol{i} >$)))
Neural-AMR	(o / obligate-01 :arg2 (r / $\boldsymbol{rule - out} - \mathbf{02} < \boldsymbol{i} >$:arg0 (r2 / $\boldsymbol{right} < \boldsymbol{i} >$:arg1-of (p / $\boldsymbol{private} - \mathbf{03} < \boldsymbol{i} >$)) :arg1 (w / welfare :mod (p2 / public))))	(p / permit-01 :polarity - :arg1 (a / $\boldsymbol{abuse} - \mathbf{02} < \boldsymbol{i} >$:arg1 (r / $\boldsymbol{right} < \boldsymbol{i} >$)))
Ch-AMR	(vv1conform-01 / conform-01 :ARG1 (vv1person / $\boldsymbol{person} < \boldsymbol{i} >$:ARG1-of (vv1private-03 / $\boldsymbol{private} - \mathbf{03} < \boldsymbol{i} >$)) :ARG2 (vv1welfare / welfare :ARG1-of $\boldsymbol{vv1} < \boldsymbol{i} >$)) (missing concept }}$\boldsymbol{right} - \mathbf{05}''$, }}$\boldsymbol{public} - \mathbf{02}''$, }}$\boldsymbol{obligate} - \mathbf{01}'' < \boldsymbol{iii} >$)	(vv3permit-01 / permit-01 :ARG1 (vv3no-abuse / $\boldsymbol{no - abuse} < \boldsymbol{i} >$)) (missing concept }}$\boldsymbol{right} - \mathbf{05}'' < \boldsymbol{iii} >$)

The inputs include representations of edges that are connected to v_j, where v_j can be either the source or the target of the edge. Each edge is defined as a triple $(i; j; l)$, where i and j are indices of the source and target nodes, respectively, and l is the edge label. $x^l_{i;j}$ 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 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 edge inputs, the model also takes the hidden states of the incoming and outgoing neighbors of each node during a state transition. Taking v_j as an example, the states of its incoming and outgoing neighbors are summed up before being passed to the cell and 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_{t-1} to g_t, as represented by h^j_t, can be defined as:

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

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.

Figure 9 shows a demonstration of graph state transition. After k iterations, we obtain the last hidden state of the graph, containing all the hidden vectors of nodes in it, where k is the maximum graph diameter in the dataset (we choose $k = 9$ in our experiments).

5.2 The proposed method

Conditional training (CT)

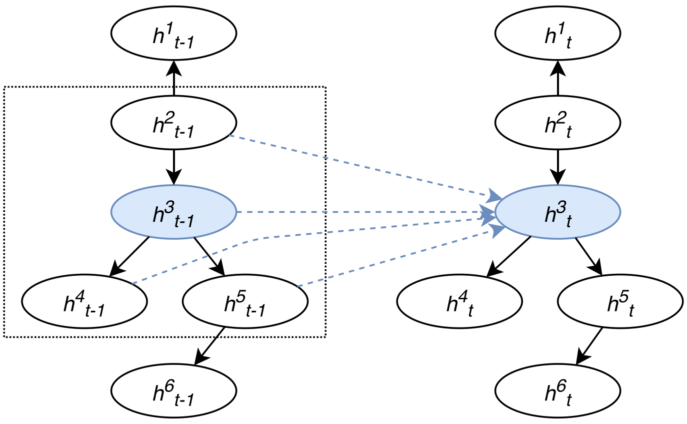


Fig. 9 Transition from graph state g_{t-1} to g_t, where information from the current node h^3_{t-1}, its incoming node h^2_{t-1} and outgoing nodes h^4_{t-1}, h^5_{t-1} are captured and transferred to h^3_t

This training method (Fan et al. 2018) aims to learn an encoder-decoder model $P(y|g, z)$, where z is a discrete control variable and g is the AMR graph. We design 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:

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

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 $loss_{CT}$.

Legal Decoding (LD)

To enhance the probability of generating words with certain features, we adopt Weighted Decoding (WD) that was introduced by Ghazvininejad et al. (2017). On the t_{th} step of decoding, the generated hypothesis $y_{<t} = y_1, \dots, y_{t-1}$ is expanded by computing the score for each possible next word w in the vocabulary by the formula:

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

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

Table 6 Generation results in BLEU score, METEOR score and number of OOV generated. The baseline Graph2Seq is trained on benchmark dataset only. Our proposed models are presented in the next row, with and without finetuning data. The last row shows 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

hypothesis $y_{<t}$ and $f_i(w;y_{<t}, g)$ are decoding features with the corresponding weights δ_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 (Pennington et al. 2014).

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.

5.3 Experiments and analysis

We evaluate our models mainly by BLEU score (Papineni et al. 2002) and METEOR score (Denkowski and Lavie 2014). We also report the number of OOV words generated from each model.

From Table 6, it can be observed that our both proposed modifications improve the performance of text generation. Compared to the baseline model, Conditional 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

Table 7 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 .

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 (Song et al. 2018) and NeuralAMR (Konstas et al. 2017). 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.

To have a closer look, we provide some output examples for each model in Table 7. 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.

6 Conclusion

To summarize this article, we study the problem of both AMR Parsing and Generation for texts in the legal domain. We provide an overview of different methods in AMR parsing and their performances when analyzing legal documents. We conduct experiments of different AMR parsers on our annotated legal dataset JCivilCode to figure out the challenges in applying to this domain. 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. To tackle this problem, we propose two modifications to an encoder-decoder AMR-to-text generation model. Specifically, we add constraints to the training phase (conditional training) and modifying the decoding algorithm with weighted decoding and legal vocabulary (legal decoding). In which, the legal vocabulary is obtained through the finetuning process, extracted from the VNCivilCode dataset. Our proposed method is tested on JCivilCode, showing an improvement compared to the baseline model.

7 Future works

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

- 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 (Xie et al. 2020; Shaw et al. 2020), 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 SP*k* Languages to explore the characteristics of long-distance dependencies (Mahalunkar and Kelleher 2019). 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 previous sections, 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 (Shaw et al. 2019; Dong and Lapata 2016). We plan to explore these works to build a logical attention mechanism to capture these information more effetively.

Acknowledgments This work was supported by JST CREST Grant Number JPMJCR1513 Japan, JSPS Kakenhi Grant Number 20H04295, 20K2046, and 20K20625. The research also was supported in part by the Asian Office of Aerospace R\&D (AOARD), Air Force Office of Scientific Research (Grant no. FA2386-19-1-4041).

References

Abend O, Rappoport A (2013) Universal conceptual cognitive annotation (UCCA). In: Proceedings of the 51st annual meeting of the association for computational linguistics (Volume 1: Long Papers), pp. 228–238. Association for Computational Linguistics, Sofia, Bulgaria. https://www.aclweb.org/anthology/P13-1023

Ballesteros M, Al-Onaizan Y (2017) AMR parsing using stack-LSTMs. In: Proceedings of the 2017 conference on empirical methods in natural language processing, pp. 1269–1275. Association for Computational Linguistics, Copenhagen, Denmark. https://doi.org/10.18653/v1/D17-1130

Banarescu L, Bonial C, Cai S, Georgescu M, Griffitt K, Hermjakob U, Knight K, Koehn P, Palmer M, Schneider N (2013) Abstract meaning representation for sembanking. In: Proceedings of the 7th linguistic annotation workshop and interoperability with discourse, pp. 178–186. Association for Computational Linguistics, Sofia, Bulgaria

Basile V, Bos J, Evang K, Venhuizen N (2012) Developing a large semantically annotated corpus. In: Proceedings of the eighth international conference on language resources and evaluation (LREC'12), pp. 3196–3200. European Language Resources Association (ELRA), Istanbul, Turkey. http://www.lrec-conf.org/proceedings/lrec2012/pdf/534_Paper.pdf

Brandt L, Grimm D, Zhou M, Versley Y (2016) 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), pp. 1160–1166. Association for Computational Linguistics, San Diego, California. https://doi.org/10.18653/v1/S16-1179

Cai S, Knight K (2013) Smatch: an evaluation metric for semantic feature structures. In: Proceedings of the 51st annual meeting of the association for computational linguistics (Volume 2: Short Papers), pp. 748–752. Association for Computational Linguistics, Sofia, Bulgaria

Cao K, Clark S (2019) 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), pp. 2157–2163. Association for Computational Linguistics, Minneapolis, Minnesota. https://doi.org/10.18653/v1/N19-1223

Damonte M, Cohen SB (2019) Structural neural encoders for AMR-to-text generation. In: Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, Volume 1 (Long and Short Papers), pp. 3649–3658. Association for Computational Linguistics, Minneapolis, Minnesota. https://doi.org/10.18653/v1/N19-1366

Damonte M, Cohen SB, Satta G (2017) An incremental parser for abstract meaning representation. In: Proceedings of European chapter of the ACL (EACL)

Denkowski M, Lavie A (2014) Meteor universal: Language specific translation evaluation for any target language. In: Proceedings of the EACL 2014 workshop on statistical machine translation

Dohare S, Karnick H, Gupta V (2017) Text summarization using abstract meaning representation. arXiv preprint arXiv:1706.01678

Dong L, Lapata M (2016) Language to logical form with neural attention. In: Proceedings of the 54th annual meeting of the association for computational linguistics (Volume 1: Long Papers), pp. 33–43. Association for Computational Linguistics, Berlin, Germany. https://doi.org/10.18653/v1/P16-1004. https://www.aclweb.org/anthology/P16-1004

Fan A, Grangier D, Auli M (2018) Controllable abstractive summarization. In: Proceedings of the 2nd Workshop on Neural Machine Translation and Generation, pp. 45–54. Association for Computational Linguistics, Melbourne, Australia. https://doi.org/10.18653/v1/W18-2706. https://www.aclweb.org/anthology/W18-2706

Flanigan J, Thomson S, Carbonell J, Dyer C, Smith NA (2014) A discriminative graph-based parser for the abstract meaning representation. In: Proceedings of the 52nd annual meeting of the association for computational linguistics (Volume 1: Long Papers), pp. 1426–1436. Association for Computational Linguistics, Baltimore, Maryland. https://doi.org/10.3115/v1/P14-1134

Foland W, Martin JH (2017) Abstract meaning representation parsing using LSTM recurrent neural networks. In: Proceedings of the 55th annual meeting of the association for computational linguistics (Volume 1: Long Papers), pp. 463–472. Association for Computational Linguistics, Vancouver, Canada. https://doi.org/10.18653/v1/P17-1043

Ge D, Li J, Zhu M, Li S (2019) Modeling source syntax and semantics for neural amr parsing. In: Proceedings of the twenty-eighth international joint conference on artificial intelligence, IJCAI-19, pp.

4975–4981. International joint conferences on artificial intelligence organization. https://doi.org/10.24963/ijcai.2019/691. https://doi.org/10.24963/ijcai.2019/691

Ghazvininejad M, Shi X, Priyadarshi J, Knight K (2017) Hafez: an interactive poetry generation system. In: Proceedings of ACL 2017, system demonstrations, pp. 43–48. Association for computational linguistics, Vancouver, Canada. https://www.aclweb.org/anthology/P17-4008

Goodman J, Vlachos A, Naradowsky J (2016) Noise reduction and targeted exploration in imitation learning for abstract meaning representation parsing. In: Proceedings of the 54th annual meeting of the association for computational linguistics (Volume 1: Long Papers), pp. 1–11. Association for Computational Linguistics, Berlin, Germany. https://doi.org/10.18653/v1/P16-1001

Gu J, Lu Z, Li H, Li VO (2016) Incorporating copying mechanism in sequence-to-sequence learning. In: Proceedings of the 54th annual meeting of the association for computational linguistics (Volume 1: Long Papers), pp. 1631–1640. Association for Computational Linguistics. https://doi.org/10.18653/v1/P16-1154. http://aclweb.org/anthology/P16-1154

Hardy H, Vlachos A (2018) Guided neural language generation for abstractive summarization using abstract meaning representation. In: Proceedings of the 2018 conference on empirical methods in natural language processing, pp. 768–773. Association for Computational Linguistics, Brussels, Belgium. https://doi.org/10.18653/v1/D18-1086

Jones B, Andreas J, Bauer D, Hermann KM, Knight K, (2012) Semantics-based machine translation with hyperedge replacement grammars. In: Proceedings of COLING 2012, pp. 1359–1376. The COLING, (2012) Organizing Committee. Mumbai, India

Katayama T (2007) Legal engineering-an engineering approach to laws in e-society age. In: Proceedings of the 1st international workshop on JURISIN

Konstas I, Iyer S, Yatskar M, Choi Y, Zettlemoyer L (2017) Neural AMR: Sequence-to-sequence models for parsing and generation. In: Proceedings of the 55th annual meeting of the association for computational linguistics (Volume 1: Long Papers), pp. 146–157. Association for Computational Linguistics, Vancouver, Canada. https://doi.org/10.18653/v1/P17-1014

Liao K, Lebanoff L, Liu F (2018) Abstract meaning representation for multi-document summarization. In: Proceedings of the 27th international conference on computational linguistics, pp. 1178–1190. Association for Computational Linguistics, Santa Fe, New Mexico, USA

Lin Z, Xue N (2019) Parsing meaning representations: is easier always better? In: Proceedings of the first international workshop on designing meaning representations, pp. 34–43. Association for Computational Linguistics, Florence, Italy. https://doi.org/10.18653/v1/W19-3304

Liu F, Flanigan J, Thomson S, Sadeh N, Smith NA (2015) Toward abstractive summarization using semantic representations. In: Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: human language technologies, pp. 1077–1086. Association for Computational Linguistics, Denver, Colorado. https://doi.org/10.3115/v1/N15-1114

Liu Y, Che W, Zheng B, Qin B, Liu T (2018) An AMR aligner tuned by transition-based parser. In: Proceedings of the 2018 conference on empirical methods in natural language processing, pp. 2422–2430. Association for Computational Linguistics, Brussels, Belgium. https://doi.org/10.18653/v1/D18-1264

Lyu C, Titov I (2018) AMR parsing as graph prediction with latent alignment. In: Proceedings of the 56th annual meeting of the association for computational linguistics (Volume 1: Long Papers), pp. 397–407. Association for Computational Linguistics, Melbourne, Australia. https://doi.org/10.18653/v1/P18-1037

Mahalunkar A, Kelleher J (2019) 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, pp. 34–43. Association for Computational Linguistics, Florence. https://doi.org/10.18653/v1/W19-3904. https://www.aclweb.org/anthology/W19-3904

Mitra A, Baral C (2016) Addressing a question answering challenge by combining statistical methods with inductive rule learning and reasoning. In: Thirtieth AAAI conference on artificial intelligence

Nakamura M, Nobuoka S, Shimazu A (2007) Towards translation of legal sentences into logical forms. In: Annual conference of the Japanese society for artificial intelligence, pp. 349–362. Springer

Napoles C, Gormley M, Van Durme B (2012) Annotated Gigaword. In: Proceedings of the joint workshop on automatic knowledge base construction and web-scale knowledge extraction (AKBC-WEKEX), pp. 95–100. Association for Computational Linguistics, Montréal, Canada. https://www.aclweb.org/anthology/W12-3018

Naseem T, Shah A, Wan H, Florian R, Roukos S, Ballesteros M (2019) Rewarding Smatch: Transition-based AMR parsing with reinforcement learning. In: Proceedings of the 57th annual meeting of the association for computational linguistics, pp. 4586–4592. Association for Computational Linguistics, Florence, Italy. https://doi.org/10.18653/v1/P19-1451

Navas-Loro M, Satoh K, Rodríguez-Doncel V (2019) Contractframes: bridging the gap between natural language and logics in contract law. In: Kojima K, Sakamoto M, Mineshima K, Satoh K (eds) New frontiers in artificial intelligence. Springer International Publishing, Cham, pp 101–114

van Noord R, Bos J (2017) Neural semantic parsing by character-based translation: experiments with abstract meaning representations. Comput Linguist Netherlands J 7:93–108

Papineni K, Roukos S, Ward T, Zhu WJ (2002) Bleu: a method for automatic evaluation of machine translation. In: Proceedings of 40th annual meeting of the association for computational linguistics, pp. 311–318. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA. https://doi.org/10.3115/1073083.1073135. https://www.aclweb.org/anthology/P02-1040

Peng X, Song L, Gildea D (2015) A synchronous hyperedge replacement grammar based approach for AMR parsing. In: Proceedings of the nineteenth conference on computational natural language learning, pp. 32–41. Association for Computational Linguistics, Beijing, China. https://doi.org/10.18653/v1/K15-1004

Peng X, Wang C, Gildea D, Xue N (2017) Addressing the data sparsity issue in neural AMR parsing. In: Proceedings of the 15th conference of the European chapter of the association for computational linguistics: Volume 1, Long Papers, pp. 366–375. Association for Computational Linguistics, Valencia, Spain

Pennington J, Socher R, Manning C (2014) Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1532–1543. Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1162. http://aclweb.org/anthology/D14-1162

Pourdamghani N, Knight K, Hermjakob U (2016) Generating English from abstract meaning representations. In: Proceedings of the 9th international natural language generation conference, pp. 21–25. Association for Computational Linguistics, Edinburgh, UK. https://doi.org/10.18653/v1/W16-6603

Rao S, Marcu D, Knight K, Daumé III H (2017) Biomedical event extraction using abstract meaning representation. In: BioNLP 2017, pp. 126–135. Association for Computational Linguistics, Vancouver, Canada. https://doi.org/10.18653/v1/W17-2315. https://www.aclweb.org/anthology/W17-2315

Ribeiro LFR, Gardent C, Gurevych I (2019) Enhancing AMR-to-text generation with dual graph representations. In: Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP), pp. 3174–3185. Association for Computational Linguistics, Hong Kong, China. https://doi.org/10.18653/v1/D19-1314. https://www.aclweb.org/anthology/D19-1314

Sachan M, Xing E (2016) Machine comprehension using rich semantic representations. In: Proceedings of the 54th annual meeting of the association for computational linguistics (Volume 2: Short Papers), pp. 486–492. Association for Computational Linguistics, Berlin, Germany. https://doi.org/10.18653/v1/P16-2079

Shaw P, Massey P, Chen A, Piccinno F, Altun Y (2019) Generating logical forms from graph representations of text and entities. In: Proceedings of the 57th annual meeting of the association for computational linguistics, pp. 95–106. Association for Computational Linguistics, Florence, Italy. https://doi.org/10.18653/v1/P19-1010. https://www.aclweb.org/anthology/P19-1010

Shaw S, Pajak M, Lisowska A, Tsaftaris SA, O'Neil AQ (2020) Teacher-student chain for efficient semi-supervised histology image classification. arXiv preprint arXiv:2003.08797

Song L, Gildea D, Zhang Y, Wang Z, Su J (2019) Semantic neural machine translation using AMR. Trans Assoc Comput Linguist 7:19–31. https://doi.org/10.1162/tacl_a_00252

Song L, Zhang Y, Wang Z, Gildea D (2018) A graph-to-sequence model for AMR-to-text generation. In: Proceedings of the 56th annual meeting of the association for computational linguistics (Volume 1: Long Papers), pp. 1616–1626. Association for Computational Linguistics, Melbourne, Australia. https://doi.org/10.18653/v1/P18-1150

Wang C, Pradhan S, Pan X, Ji H, Xue N (2016) CAMR at SemEval-2016 task 8: An extended transition-based AMR parser. In: Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016), pp. 1173–1178. Association for Computational Linguistics, San Diego, California. https://doi.org/10.18653/v1/S16-1181

Wang C, Xue N, Pradhan S (2015) 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, pp. 366–375. Association for Computational Linguistics, Denver, Colorado. https://doi.org/10.3115/v1/N15-1040

Xie Q, Luong MT, Hovy E, Le QV (2020) Self-training with noisy student improves imagenet classification. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 10687–10698

Zhang S, Ma X, Duh K, Van Durme B (2019) AMR parsing as sequence-to-graph transduction. In: Proceedings of the 57th annual meeting of the association for computational linguistics, pp. 80–94. Association for Computational Linguistics, Florence, Italy. https://doi.org/10.18653/v1/P19-1009

Zhou J, Xu F, Uszkoreit H, Qu W, Li R, Gu Y (2016) AMR parsing with an incremental joint model. In: Proceedings of the 2016 conference on empirical methods in natural language processing, pp. 680–689. Association for Computational Linguistics, Austin, Texas. https://doi.org/10.18653/v1/D16-1065

Zhu J, Li J, Zhu M, Qian L, Zhang M, Zhou G (2019) Modeling graph structure in transformer for better AMR-to-text generation. In: Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP), pp. 5462–5471. Association for Computational Linguistics, Hong Kong, China. https://doi.org/10.18653/v1/D19-1548. https://www.aclweb.org/anthology/D19-1548