Can AMR Assist Legal and Logical Reasoning?

Schrack, Nikolaus ; Cui, Ruixiang ; López-Acosta, Hugo-Andrés; Hershcovich, Daniel

Published in:
Findings of the Association for Computational Linguistics

Publication date:
2022

Document Version
Publisher's PDF, also known as Version of record

Can AMR Assist Legal and Logical Reasoning?

Nikolaus Schrack
Department of Computer Science
University of Copenhagen
nikolaus.schrack@di.ku.dk

Ruixiang Cui
Department of Computer Science
University of Copenhagen
rc@di.ku.dk

Hugo A. López
DTU Compute
Technical University of Denmark
hulo@dtu.dk

Daniel Hershcovich
Department of Computer Science
University of Copenhagen
dh@di.ku.dk

Abstract

Abstract Meaning Representation (AMR) has been shown to be useful for many downstream tasks. In this work, we explore the use of AMR for legal and logical reasoning. Specifically, we investigate if AMR can help capture logical relationships on multiple choice question answering (MCQA) tasks. We propose neural architectures that utilize linearised AMR graphs in combination with pre-trained language models. While these models are not able to outperform text-only baselines, they correctly solve different instances than the text models, suggesting complementary abilities. Error analysis further reveals that AMR parsing quality is the most prominent challenge, especially regarding inputs with multiple sentences. We conduct a theoretical analysis of how logical relations are represented in AMR and conclude it might be helpful in some logical statements but not for others.[1]

1 Introduction

Legal NLP has become a highly researched topic in recent times because of its many real-world applications (Zhong et al., 2020). The field has been concerned with developing tools to help legal practitioners with time-consuming and repetitive tasks such as finding similar court cases. Many of these tasks require reading huge amounts of legal documents, which can take a long time and therefore benefit from automation. Legal NLP aims to build systems that can help legal experts as well as people without legal knowledge. Examples of this could be a QA system that allows consumers to ask questions about their data privacy rights or a system that can tell citizens if they are eligible for a certain social service program by stating their case.

Regardless of the task, a system needs to be able to capture the semantics of the relevant legal documents. This can be done implicitly by using the text directly, or explicitly with semantic representation frameworks. Semantic parsing is the process of converting natural text into a graph-structured representation of sentence meaning (Abend and Rappoport, 2017; Žabokrtský et al., 2020). The idea is to utilise the semantic graphs instead of or in addition to the textual input, which allows the system to better encode the document semantics.

Abstract Meaning Representation (AMR; Banarescu et al., 2013) represents the semantics of a sentence as a rooted, directed acyclic graph, where nodes represent concepts and edges encode relations. The advances in AMR parsing have been significant in recent years (Bevilacqua et al., 2021; Bai et al., 2022), with state-of-the-art (SOTA) AMR parsers achieving Smatch scores (Cai and Knight, 2013) higher than 84 on the latest AMR 3.0 dataset (Knight et al., 2021).[2] This creates the possibility of using AMR for downstream tasks including Commonsense Reasoning (Lim et al., 2020), Information Extraction (Zhang and Ji, 2021) and Question Answering (Kapanipathi et al., 2021).

Contributions. This paper investigates whether AMR can help legal and logical reasoning on MCQA tasks. Specifically, we investigate if AMR can help capture logical relationships, since understanding the logic in law is a major challenge in legal NLP and AMR facilitates the representation of some logical structure in sentences. Different models utilising AMR are tested and compared with text-only baseline systems on a MCQA task targeting logical reasoning. Lastly, we provide an error analysis to identify issues with the proposed

[1] Code and models are available on https://github.com/nschrack/fusion.

[2] https://paperswithcode.com/sota/amr-parsing-on-ldc2020t02

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 1555 - 1568
December 7-11, 2022

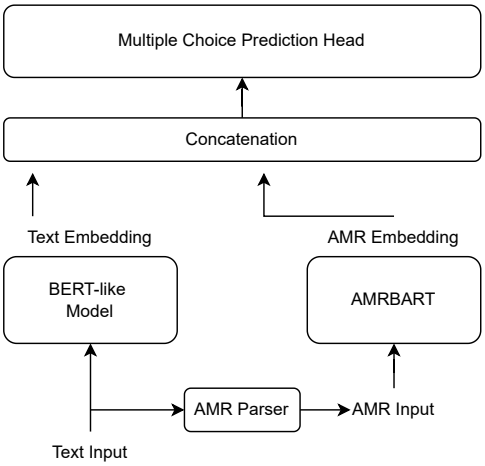

Figure 1: Fusion model (§4.4) using both AMR and text input. Concatenated PLMs representations are input to the prediction head. For text, we use LegalBERT$_{\text{BASE}}$ for CaseHOLD and BERT$_{\text{BASE}}$ for LogiQA.

architectures, concluding that AMR parsing quality is a major bottleneck.

2 Logical Relations in AMR

To reason about whether AMR can help capture logic in law, consider quantifiers, negation, conjunction, disjunction, implication and equivalence—logical operators used in propositional logic (Hurley, 2014). Some logical statements and their corresponding logical connectives are represented consistently in AMR, regardless of the specific English expression (surface form). This includes conditional statements with *if*, *unless* and *in case of*, etc., represented using the `:condition` role. The core concept of the consequence is the root node and the antecedent has the role `:condition`. For example, "no major traffic accidents will occur if the highway is not closed" is represented as:

```
(a / accident
    :polarity -
    :mod (t / traffic)
    :ARG1-of (n / major-02)
    :condition (c / close-01
        :polarity -
        :ARG1 (h / highway)))
```

In this case "no major accident will occur" is the consequence and "the highway is not closed" is the antecedent. Negation is represented in a logical sense with the `:polarity` role. Furthermore, AMR aims to represent the semantics of a sentence independently of syntax, meaning that the same graph corresponds to multiple sentences. The AMR would not change if the consequence and antecedent were reversed, i.e., "if the highway is not closed, no major traffic accidents will occur."

In other cases the representation is closer to the surface form. For example, the `and` concept is used to represent both conjunctive statements and conjunction of entities and therefore does not always represent logical conjunction. It uses the `:op`*N* roles for the operands. The sentence "all musicians are capable of reading music and some musicians are capable of improvising" is represented as:

```
(a / and
    :op1 (c / capable-01
        :ARG1 (m / musician
            :mod (a1 / all))
        :ARG2 (r / read-01
            :ARG0 m
            :ARG1 (m1 / music)))
    :op2 (c2 / capable-01
        :ARG1 (m3 / musician
            :quant (s / some))
        :ARG2 (i / improvise-01
            :ARG0 m3)))
```

Besides conjunction between statements, `and` can represent a list. "Ms.Cai, Ms.Zhu and Ms.Sun are newly recruited by a school" is represented as:

```
(r / recruit-01
    :ARG0 (s / school)
    :ARG1 (a / and
        :op1 (p / person
            :name (n / name
                :op1 "Ms.Cai"))
        :op2 (p1 / person
            :name (n1 / name
                :op1 "Ms.Zhu"))
        :op3 (p2 / person
            :name (n2 / name
                :op1 "Ms.Sun")))
    :ARG1-of (n3 / new-01))
```

The `and` concept does not always represent a conjunctive statement, similar to how *and* in a sentence is not only used to connect two statements. This holds also for disjunctive statements and *or*.

Other conjunction words, such as *moreover*, use different roles or concepts. The conjunctions *but* and *however* are represented by the concepts `contrast-01` or `instead-of-91` or the role `:concession-of`. This is an example of how, besides logical relationships, certain AMR concepts and roles correspond to discourse connectives

(Prasad et al., 2008; Das et al., 2018).[3]

To conclude, AMR helps capture some logical statements but not others.[4]

3 Related Work

Song et al. (2019) incorporated structured semantic information from AMRs for Machine Translation. They use a graph recurrent network (GRN) to encode AMRs and a sequential LSTM (Hochreiter and Schmidhuber, 1997) to encode the source input text. For the decoder model, they use a doubly attentive LSTM architecture, taking both the graph and text encoding as attention memory. They show on an English-to-German translation task that using AMR as complementary to the source text input improves performance.

There have been attempts of using linearised AMRs with Pretrained Language Models (PLM). For example, Mager et al. (2020) fine-tuned a Transformer language model with linearised AMR graphs for the AMR-to-text task. Various methods for graph linearisation and simplifications have been tried such as depth-first traversal through the graph (Konstas et al., 2017). Linearised AMRs have also been used in combination with CNNs (Viet et al., 2017) and Phrase-Based models (Pourdamghani et al., 2016).

The introduction of large PLMs such as BERT (Devlin et al., 2018) has led to new SOTA performance in many NLP domains in recent years. In the legal domain, using domain-specific PLMs for simpler tasks such as text classification has only shown small improvements (Clavi'e and Alphonsus, 2021; Chalkidis et al., 2020). However, bigger gains were achieved for more complex tasks (Zheng et al., 2021). Other efforts have addressed the issue that legal documents are oftentimes much longer than the input size of standard Transformer models such as BERT. PLMs for long sequences such as Longformer (Beltagy et al., 2020) have shown to be beneficial for such tasks (Xiao et al., 2021; Limsopatham, 2021). To the best of our knowledge, no one has tried to leverage structured semantic information for a reading comprehension task in the legal domain.

Similar to the legal domain, large PLMs such as BERT have struggled with reading comprehension tasks that require logical reasoning (Yu et al., 2020; Liu et al., 2020). Huang et al. (2021) proposed a QA model which constructs logical graphs from a set of elementary discourse units (EDUs), where the edges are discourse relations. Li et al. (2022) improved this method by introducing logical relations mapped from rhetorical relations using Graphene (Cetto et al., 2018). In our work, we try to use AMR graphs to capture logical relations.

Other work has explored the use of AMR for MCQA tasks. Xu et al. utilises AMR graphs to fuse semantic concepts between the hypothesis and retrieved evidence facts to find reasoning chains (Xu et al., 2021a) and create active fact-level connection graphs (Xu et al., 2021b) for multi-hop Science Question Answering. The reasoning chains/connection graphs are used to select relevant facts and to guide the reasoning process. In comparison, we use AMR graphs from the context and answer directly with a pre-trained AMR language model to extract embeddings.

Motivated by a similar research question, Glavaš and Vulić (2021) investigated whether supervised syntactic parsing is beneficial for natural language understanding (NLU) tasks. Rather than meaning representation, they focused on syntactic representation with Universal Dependencies (de Marneffe et al., 2021) and their methodology is different from ours in that they use fine-tuning on parsing (with a biaffine decoder) as a way to infuse the symbolic representation into the model, whereas we incorporate linearised AMR graphs directly into the architecture. Furthermore, we focus on specific NLU tasks that require logical or legal reasoning. Nevertheless, we reach a similar conclusion, namely that explicit symbolic representation provides negligible impact, if any.

4 AMR for MCQA

In MCQA, each instance consists of a context paragraph, a question and several answer options, among which only one is correct. A model is evaluated by accuracy, that is, frequency of selecting the correct answer. We propose a system for MCQA by utilising semantic encoding with AMR, to hopefully better capture semantics than a text-only system. Our system is composed of an AMR parser for converting the source text into AMR graphs, an encoding component for AMR based on graph linearisation in combination with a PLM (a domain-

[3] Capturing discourse relations is in fact often useful for legal reasoning even if they do not correspond to logical operators (Walker et al., 2017; Huang et al., 2021). Nevertheless, here we focus on logical reasoning and assume the text is semantically self-contained.

[4] Our full analysis of logical expressions is in Appendix A.

specific model for legal text) and a feedforward layer that takes the text and graph encoding as inputs and makes predictions.

4.1 Similarity-based Baseline

Our first baseline model is a rule-based model based solely on the AMR graphs. Similar to the methodology of Bonial et al. (2020) we employ Smatch (Cai and Knight, 2013) as the similarity metric. Smatch is a measure of the degree of overlap between two AMR graphs. The model calculates the Smatch score for each context statement and answer option and then chooses the answer option with the highest Smatch score. This is based on the idea that the context and the appropriate answer option have similar semantics. To calculate the Smatch score the `amrlib`[5] library is used.

4.2 Encoding Linearised AMR with a PLM

The challenge of using semantic graphs as inputs to a PLM, which was trained on text, has to be addressed. The most basic approach is to fine-tune the model using linearised AMR graphs directly (Mager et al., 2020). There are different techniques on how to linearise and simplify AMR graphs. This includes using the Penman representation (Bateman and Matthiessen Licheng, 1999; Goodman, 2020) directly, using only nodes in a breadth-first search approach and other simplification methods such as removing redundant brackets and variables (Mager et al., 2020; Konstas et al., 2017). For this model, the linearisation and simplification technique introduced by Konstas et al. (2017) was used for preprocessing the graphs. In addition, AMR roles are kept in their original form with a leading colon. Even if they resemble a word such as e.g. `:location`, optimally they are understood by the model as representing an edge in the graph. To facilitate this, all roles from the training dataset were accumulated and added to the tokenizer as special tokens.

We use LegalBERT (Chalkidis et al., 2020), a model pre-trained on English legal text including legislation, court cases and contracts, for this architecture. It achieve SOTA results on the LexGLUE benchmark (Chalkidis et al., 2022).

We also experiment with adapters since they allow for effective transfer learning (Ribeiro et al., 2021). For this model, the intention is that the adapter parameters can learn the linearised graph representation, while the parameters of the model that hold the distributed knowledge of pre-training are not changed. Adapter training is done using the `adapter-transformers` library.[6] The default configuration for the adapter is used.

4.3 AMRBART Model

Another approach is to use a PLM that was pre-trained on AMR graphs. The idea is to overcome the problem associated with using linearised graph input on a PLM that was pre-trained on text. AMRBART (Bai et al., 2022) is based on BART (Lewis et al., 2020) and further pre-trained on linearised AMR graphs. AMR graphs are preprocessed using Spring (Bevilacqua et al., 2021) and linearised with a DFS approach, where variables are replaced by special tokens, e.g., `<pointer:X>`. To deal with AMR symbols, the vocabulary is expanded by adding all relations and frames.

Since BART is typically not used for MCQA tasks, the Huggingface library[7] does not have a model class implementation for this task. A common way to implement the MCQA task is to process each sequence independently and then use a softmax layer to create an output distribution over all possible answers (Radford et al., 2018). In this case, a sequence is a concatenation of context and answer, separated by a delimiter token. The transformer uses a linear layer on top of the pooled output to facilitate classification. The pooled output is used as a sentence representation.

4.4 Fusion Model

Finally, we combine AMR and text input into a single architecture. We use both the original input text and the linearised predicted AMR graph, assuming it is possible to capture the semantics of the input data better. The architecture (see Figure 1) consists of a pre-trained model for each data modality, which is used to extract embeddings. The embeddings are then fused and sent into the prediction head. This type of jointly fine-tuning two pre-trained models for different data modalities was to be successful in multimodal speech emotion recognition (Siriwardhana et al., 2020).

The pre-trained models used for text encoding are $\text{LegalBERT}_{\text{BASE}}$ for CaseHOLD and

[5] https://amrlib.readthedocs.io

[6] https://docs.adapterhub.ml/

[7] https://github.com/huggingface/transformers/blob/v4.17.0/src/transformers/models/bart/modeling_bart.py

	Split Size			Text Info. (average)	
	Train	Validation	Test	# words	# sent.
CaseHOLD	45000	3900	3600	134	6[8]
LogiQA	7376	651	651	66	3.3

Table 1: Statistics of CaseHOLD and LogiQA datasets.

$\text{BERT}_{\text{BASE}}$ for LogiQA. The pre-trained model for AMR input is AMRBART. In regards to the fusion technique, we choose to use the simple method of concatenating the two embeddings. Siriwardhana et al. (2020) showed that a shallow fusion approach such as concatenation can give good results. All pre-trained models used have an embeddings size of 768, meaning that the fully connected layer of the prediction head has an input size of 1536. The embeddings are retrieved from AMRBART by using the EOS token. For the BERT models, the pooled output embeddings are used.

5 Experiments

We experiment with the model architectures proposed in §4: BERT, LegalBERT and BART, which use text-only input; AMRBART and the Smatch-based similarity model, which uses AMR-only input, and the Fusion model, which uses both inputs.

5.1 Data

Two MCQA datasets are chosen for the experiments: CaseHOLD, a legal reasoning task, and LogiQA, a logical reasoning task. Their statistics are in Table 1.

The CaseHOLD dataset (Zheng et al., 2021) presents a common task for lawyers, which is to identify the legal holding of a case. A holding is the court's application of the governing legal rule in a particular case. Holdings are an important part of the common law system since they are used as precedence by courts and litigants. The task prompts a court decision statement and gives five candidate holding statements from which one of them is correct. The data was sourced from legal citations in judicial rulings of U.S. case law. It is part of LexGLUE (Chalkidis et al., 2021), a multi-task benchmark for legal understanding in English.

LogiQA (Liu et al., 2020) is an MCQA dataset targeting logical reasoning. The data is sourced from publicly available questions from the National Civil Servants Examination of China and was professionally translated from Chinese into English. The exam is aimed at testing the participants' critical thinking and problem-solving skills. Each instance consists of a context statement, a question and four answer options. The authors identify that around 31% of the questions require categorical reasoning, 28% sufficient conditional reasoning, 25% necessary conditional reasoning, 19% disjunctive reasoning and 21% conjunctive reasoning.

5.2 Experimental Setup

We use Spring as the text-to-AMR parser (Bevilacqua et al., 2021), which is one of the SOTA parsers on AMR 3.0 and is publicly available.[9]

For CaseHOLD, the small version of Legal-BERT[10] is used, as it shows competitive results compared to the big models while being three times smaller than the base model. As a text-only model similar to AMRBART (see §4.3), we also run the experiment with $\text{BART}_{\text{BASE}}$ using text input.

For LogiQA, we experiment with $\text{BERT}_{\text{BASE}}$ as a text-only encoder. We also present the theoretical random baseline, which selects a random answer.

The models can be categorized by their input data. The first type describes models that use the source text as input. The second type relates to models that use the parsed AMR graphs as input and additionally, there is a model using both text and AMR input. There are various preprocessing techniques applied to the AMR graphs: linearisation and simplification (Konstas et al., 2017), Spring preprocessing (Bevilacqua et al., 2021) or using the original Penman notation (Bateman and Matthiessen Licheng, 1999).

The model implementations were done with Pytorch. The AdamW optimiser (Loshchilov and Hutter, 2017) was used with a learning rate of $3e^{-5}$. The dropout rate was set to 0.1. The effective batch size for the Fusion model was 4 for CaseHOLD and 8 for LogiQA (due to memory constraints). The other models had an effective batch size of 16.

6 Results

We proceed to present the results of our experiments, measuring performance by accuracy.[11]

[8] Estimated by manually counting the number of sentences for 10 instances and averaging.

[9] https://github.com/SapienzaNLP/spring

[10] https://huggingface.co/nlpaueb/legal-bert-small-uncased

[11] The *micro-F1* (μ-F_1) score, which is often used for CaseHOLD, is equivalent to accuracy for multi-class classification where exactly one class is correct.

Model	Input	Model Size	Accuracy
LegalBERT$_{\text{SMALL}}$	Text	35M	0.72[12]
LegalBERT$_{\text{SMALL}}$ + adapter	Text	35M	0.73
BART$_{\text{BASE}}$	Text	139M	0.74
LegalBERT$_{\text{SMALL}}$ + adapter	AMR (linearised and simplified)	35M	0.53
Smatch Model	AMR (Penman)	-	0.34
AMRBART$_{\text{BASE}}$	AMR (Spring prepr.)	142M	0.51
Fusion Model	Text and AMR (Spring prepr.)	252M	0.74

Table 2: CaseHOLD performance of different models with text and AMR input. The Fusion Model uses LegalBERT$_{\text{BASE}}$ to encode the text and AMRBART$_{\text{BASE}}$ to encode the linearised AMR.

6.1 CaseHOLD

Table 2 shows the performance on CaseHOLD. The baseline model LegalBERT$_{\text{SMALL}}$ with text input is slightly improved when using adapters for fine-tuning. The BART$_{\text{BASE}}$ model received an accuracy of 0.74, which is similar to the other text models.

The Smatch model performs poorly with an accuracy of 0.34. This shows that the simple approach of using the highest Smatch score to predict the holding statement does not yield good results. LegalBERT$_{\text{SMALL}}$ with linearised and simplified AMR input using adaptor training gets an accuracy of 0.53. The results are worse compared to 0.74 accuracy of the same model with text input.

AMRBART achieves 0.51 accuracy. The model was pre-trained on AMR graphs and therefore is expected to capture semantics better than models trained on text input. Still, compared to LegalBERT$_{\text{SMALL}}$ with AMR input the performance is slightly worse. The Fusion model combines text and AMR input. The model archives an accuracy of 0.74. This is, tied with the BART$_{\text{BASE}}$ model, the highest score out of all conducted experiments. There is no notable performance increase compared to the text models.

6.2 LogiQA

Table 3 shows the performance of various models with different input data types on the LogiQA dataset. The baseline BERT$_{\text{BASE}}$ model achieves the highest accuracy of 0.28[13]. AMRBART$_{\text{BASE}}$ and the Fusion model both have the accuracy of 0.27. The results show that BERT with text input outperforms both AMRBART and the Fusion model.

7 Error Analysis

Models using only AMR input perform overall worse than models with text input. To check if they solve different instances than the text models, Table 4 shows the number of correctly predicted instances for CaseHOLD and LogiQA, including the intersection between the models. For this analysis, we choose AMRBART as the AMR model and LegalBERT/BERT as the text model for CaseHOLD and LogiQA, respectively.

For CaseHOLD, the percentage of intersecting instances is 86%. This means that most of AMRBART's correct predictions were also correctly predicted by the text model. On the other hand, the percentage of intersecting instances for LogiQA is 29%. This means that less than a third of the correctly predicted instances by the AMR model were also correctly predicted by the text model.

To see if the AMR model can consistently solve different instances than the text model, the experiments were run three times for the LogiQA dataset. The results (see Table 5) show that BERT solves 65 instances and AMRBART solves 76 instances consistently over three runs. The theoretical number of consistently correctly predicted instances of a random model is around 10.17 elements.[14] The number of overlapping instances between text and AMR model is 13, showing that over 80% of the consistently correctly predicted instances of AMRBART were not solved by BERT. This indicates that the AMR model has learnt different knowledge about logical relations comparing to the text-only models and hence the prediction difference.

[12]The official LexGLUE benchmark ranking has reported an accuracy of 0.747 for LegalBERT$_{\text{SMALL}}$.

[13]Previous work by (Liu et al., 2020) has reported an accuracy of 0.32. We receive an avg. accuracy of 0.3 over 3 runs. Since the Fusion model has only been run 1 time, we report the results for all models for seed 1.

[14]Given there are four answer options, the number of times a random model predicts an instance correctly three times in a

Model	Input	Model Size	Accuracy
Random			0.25
$\text{BERT}_{\text{BASE}}$	Text	139M	0.28
$\text{AMRBART}_{\text{BASE}}$	AMR (Spring prepr.)	142M	0.27
Fusion Model	Text and AMR (Spring prepr.)	252M	0.27

Table 3: LogiQA performance of different models with text and AMR input. The Fusion Model uses $\text{BERT}_{\text{BASE}}$ to encode the text and $\text{AMRBART}_{\text{BASE}}$ to encode the linearised AMR.

Dataset	Size	BERT-like.	AMRBART	Intersection
CaseHOLD	3600	2598	1841	1598 (86%)
LogiQA	651	182	173	50 (29%)

Table 4: The numbers of correctly predicted instances of a text-only BERT-like model and AMRBART for LogiQA and CaseHOLD dataset. BERT-like. is LegalBERT for CaseHOLD and BERT for LogiQA. The Intersection column contains the number of correctly predicted instances by both models. Since the test dataset size differs, we show the percentage of intersecting instances compared to the total size of the correctly predicted instances of the AMRBART model.

Random	BERT	AMRBART	Inters.
10.17	65	75	13

Table 5: The number of consistently correctly predicted instances over 3 runs of LogiQA for BERT and AMRBART. Intersection shows the number of overlapping instances between BERT and AMRBART.

7.1 Parser Quality

The AMR graphs are predicted with the Spring AMR parser. Since the accuracy of the parser is not perfect, the parser will introduce noise into the generated AMRs. This can have a negative impact on downstream tasks. Studies have shown that using parsed AMRs compared to gold annotations can hurt downstream task performance (Song et al., 2019). Intuitively, long sentences and inputs with multiple sentences will be especially challenging for the parser. Most of the context data of CaseHOLD and LogiQA are multiple sentences long.

One major problem we observe when looking at samples of parsed AMR graphs is that entire sentences are missing compared to the input text, which can greatly impact performance. To see how common this phenomenon is, we investigate LogiQA and calculate the average number of sentences in the context and the parsed AMRs. The number of sentences in the AMR graphs is calculated by counting `:snt` roles in cases where the AMR had a `multi-sentences` tag and otherwise it is counted as a single sentence. The average number of sentences in the original text is 3.27, while the number for parsed AMR is 1.73. Nearly 50% of the sentences are therefore missing in the generated AMR graphs, confirming the problem.

AMRs can be represented as triples consisting of a relationship of either two variables or a variable and a concept. The number of triples is used as a measure for the size of an AMR. In Figure 2, for CaseHOLD, we draw the prediction distribution w.r.t. the ratio between the number of triples of generated AMR graphs and number of words in the original text, roughly reflecting the completeness of information the parsed AMR graphs contain (in general, lower ratio indicates less information has been parsed). In addition, we plot the accuracy of three models per triples/words ratio range. The graph shows that the accuracy increases with the triples/words ratio for AMRBART, indicating that parser quality has a great impact on performance. Text-only LegalBERT has a relatively stable performance for the same instances, showing that the difficulty of the instances does not play a role. This verifies that the loss of information during the parsing process hurts downstream task performance of AMRBART. However, the Fusion model has a nearly consistent performance, which suggests that an improvement in parser quality does increase the accuracy. This indicates that the AMR information does not contribute much to the overall model.

8 Discussion

AMR annotation limitation. The error analysis has shown that the Spring parser has difficulties parsing inputs with multiple sentences. Around half of the sentences were missing from the AMR annotations of the context of the LogiQA dataset. It was also shown that missing information from

row in a set of 651 instances $= 0.25 \cdot 0.25 \cdot 0.25 \cdot 651 \approx 10.17$

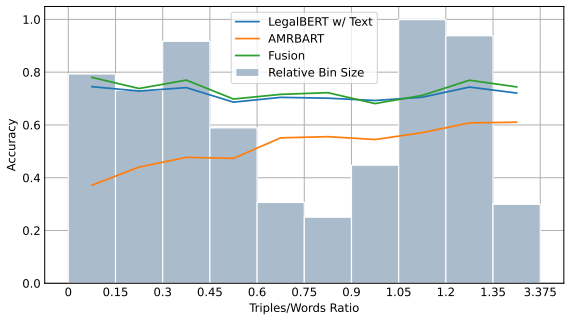

Figure 2: Ratio between triples in parsed AMR and words in the text and their average accuracy for AMRBART. As comparison the performance of the same instances for LegalBERT w/Text and Fusion model is shown. The data is taken from CaseHOLD test results. The AMRs are from the court decision statements.

parsed AMR graphs hurt downstream task performance. The absence of entire sentences from the AMR graphs is therefore one explanation for the unsatisfactory performance of the AMR models. A possible way to improve the accuracy of the AMR graphs could be to parse each sentence of the input separately and combine them manually by using the `multi-sentence` role. Since there are no coreferences annotated between sentences in the current version of AMR, there is no apparent drawback to this procedure. Another problem with parsing could be that the AMR parser has not seen domain-specific text during pre-training. When looking at examples of CaseHOLD, it is noticed that the court decision statement has a very specific writing style, e.g., frequent and abbreviated references to the law or other court cases, which could impact parser quality. The fact that the current AMR guidelines do not attend to professional domains limits its downstream application, in our case to the legal tasks.

Difficulty to encode AMR graphs. The AMR models were not able to outperform text-only models on the CaseHOLD task. As already mentioned, parser quality has most likely a big impact. Besides that, the poor performance of the Smatch model might be due to a lack of semantic similarity between the court decision statements and their holdings. For the neural models, a reason for unsatisfactory performance could be that the models were not able to encode the AMR graphs well enough. For LegalBERT, experiments indicate that even with the linearisation and simplification of the graphs, the model still struggled to understand/learn the graph structure. In this case, the AMR-specific graph elements might act as noise for the model. This issue has been the reason for using AMRBART, which was pre-trained on AMR graphs. AMRBART, however, was not able to perform better than LegalBERT. An explanation for this could be that it has never seen input with multiple AMR graphs (context and answer) in pre-training. One way to combat this is to only provide the model with one graph e.g. the answer AMR. This could be promising in the Fusion architecture, where the model still sees the entire input through the text encoding. In general, the Fusion model was performing similar to text-only models on the CaseHOLD task. Further investigation is necessary to determine what impact the AMR encoding had on task performance.

The promise of leveraging AMR. The error analysis has shown that AMRBART was able to consistently solve different instances than a text-only BERT model on the LogiQA dataset. This indicates that a model using the explicit semantic encoding of AMR can solve instances that a text-only model cannot. Furthermore, a model that uses both representations therefore might be able to outperform text-only models. The Fusion model, which uses both AMR and text, however, is underperforming on the LogiQA dataset. We conjecture that the fusion mechanism in form of concatenation does not allow the overall model to effectively make use of the semantic meaning of text and AMR representation. A potential way to solve this could be to use a co-attention layer (Siriwardhana et al., 2020) which would allow for embedding-level interaction between the encodings. Lastly, it is also possible that low performance is due to inherent limitations of the architecture.

9 Conclusion

We investigated whether AMR can help capture logical relations by conducting experiments on legal and logical reasoning datasets with model architectures that utilize AMR. In addition, a theoretical analysis was performed to see how logical statements are represented in AMR.

Specifically, we proposed four AMR model architectures for CaseHOLD, which requires legal reasoning. Using only AMR input performs worse than using text input. Using both AMR and text input showed similar performance to the text-only models. AMR models have therefore not been able

to outperform baselines.

We further analysed the performance for LogiQA, which requires logical reasoning. Again, AMR models did not outperform text-only baselines. Since only some types of logical statements are represented consistently in AMR and certain concepts and roles are not only used to represent logical relationships but are also used to annotate other semantics, we can expect this kind of mixed results. AMR might be useful to capture logical relations in some statements but not for others.

Our Fusion model takes the text encoding and graph encoding as inputs and makes predictions. Ideally, the model would leverage AMR to better capture the semantics than a text-only system. This can be a separate model or simply a task-specific head of a transformer model. Future work will further investigate how AMR can help understand the logic in law. The challenge of creating accurate document-level AMRs remains and alternative graph encoding components (Xu et al., 2018; Song et al., 2019) may be more appropriate. By addressing this, the utility of AMR on downstream tasks in the legal domain, which oftentimes requires the understanding of long documents, could be manifold. Besides automated reasoning, NLP can help humans understand logical relationships by e.g., creating simplified versions of legal text. This could in turn be used to enable semi-automatic legal process discovery (López, 2021).

10 Limitations

In §4, we make simplifying assumptions to make the modeling feasible. An idealistic model architecture would feature a state-of-the-art AMR parser incorporating document-level AMR graphs (O'Gorman et al., 2018) and overcoming the standard sentence-level representation of AMR. To accurately create document-level AMRs, it is necessary to resolve coreferences between sentences (Fu et al., 2021), which current parsers neglect.

Furthermore, the model should have access to the entire legal corpus necessary for solving the given task. This can contain legislation, court decision statement and legal contracts. The model should resolve references to, e.g., specific paragraphs of the law, which are very common in legal text. This would require multiple aspects such as retrieving the relevant documents and extracting the important information, or using generation-augmented retrieval (Mao et al., 2021).

Climate performance model card	
1. Is the resulting model publicly available?	Yes
2. How much time does the training of the final model take?	9h
3. How much time did all experiments take (incl. hyperparameter search)?	55h
4. What was the energy consumption (GPU/CPU)?	280W
5. At which geo location were the computations performed?	*Denmark*

Table 6: Climate performance model card of the models used in the experiments.

11 Broader Impact

In Table 6 we report the climate performance of this work using the climate performance model card introduced by Hershcovich et al. (2022). Note that we cannot foresee clear positive environmental impact from our work, besides the research insights that will enable more efficient modeling in the future.

In terms of societal impact, improved legal reasoning can assist humans in case handling or compliance verification, contributing to welfare and administrative efficiency. However, our contributions are limited in that respect.

References

Omri Abend and Ari Rappoport. 2017. The state of the art in semantic representation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 77–89, Vancouver, Canada. Association for Computational Linguistics.

Xuefeng Bai, Yulong Chen, and Yue Zhang. 2022. Graph pre-training for AMR parsing and generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6001–6015, Dublin, Ireland. Association for Computational Linguistics.

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

John A Bateman and Christian MIM Matthiessen Licheng. 1999. Multilingual natural language generation for multilingual software: a functional linguistic approach. *Applied Artificial Intelligence*, 13(6):607–639.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *CoRR*, abs/2004.05150.

Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. 2021. One spring to rule them both: Symmetric amr semantic parsing and generation without a complex pipeline. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):12564–12573.

Claire Bonial, Stephanie M. Lukin, David Doughty, Steven Hill, and Clare Voss. 2020. InfoForager: Leveraging semantic search with AMR for COVID-19 research. In *Proceedings of the Second International Workshop on Designing Meaning Representations*, pages 67–77, Barcelona Spain (online). Association for Computational Linguistics.

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

Matthias Cetto, Christina Niklaus, André Freitas, and Siegfried Handschuh. 2018. Graphene: Semantically-linked propositions in open information extraction. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2300–2311, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. LEGAL-BERT: The muppets straight out of law school. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904, Online. Association for Computational Linguistics.

Ilias Chalkidis, Abhik Jana, Dirk Hartung, Michael Bommarito, Ion Androutsopoulos, Daniel Katz, and Nikolaos Aletras. 2022. LexGLUE: A benchmark dataset for legal language understanding in English. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4310–4330, Dublin, Ireland. Association for Computational Linguistics.

Ilias Chalkidis, Abhik Jana, Dirk Hartung, Michael J. Bommarito II, Ion Androutsopoulos, Daniel Martin Katz, and Nikolaos Aletras. 2021. Lexglue: A benchmark dataset for legal language understanding in english. *CoRR*, abs/2110.00976.

Benjamin Clavi'e and Marc Alphonsus. 2021. The unreasonable effectiveness of the baseline: Discussing svms in legal text classification. In *JURIX*.

Debopam Das, Tatjana Scheffler, Peter Bourgonje, and Manfred Stede. 2018. Constructing a lexicon of English discourse connectives. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 360–365, Melbourne, Australia. Association for Computational Linguistics.

Marie-Catherine de Marneffe, Christopher D. Manning, Joakim Nivre, and Daniel Zeman. 2021. Universal Dependencies. *Computational Linguistics*, 47(2):255–308.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Qiankun Fu, Linfeng Song, Wenyu Du, and Yue Zhang. 2021. End-to-end AMR coreference resolution. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4204–4214, Online. Association for Computational Linguistics.

Goran Glavaš and Ivan Vulić. 2021. Is supervised syntactic parsing beneficial for language understanding tasks? an empirical investigation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3090–3104, Online. Association for Computational Linguistics.

Michael Wayne Goodman. 2020. Penman: An open-source library and tool for AMR graphs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 312–319, Online. Association for Computational Linguistics.

Daniel Hershcovich, Nicolas Webersinke, Mathias Kraus, Julia Anna Bingler, and Markus Leippold. 2022. Towards climate awareness in NLP research. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Yinya Huang, Meng Fang, Yu Cao, Liwei Wang, and Xiaodan Liang. 2021. DAGN: Discourse-aware graph network for logical reasoning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5848–5855, Online. Association for Computational Linguistics.

Patrick J. Hurley. 2014. *A concise introduction to logic*. Nelson Education.

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

Gaetano Rossiello, Udit Sharma, G P Shrivatsa Bhargav, and Mo Yu. 2021. Leveraging Abstract Meaning Representation for knowledge base question answering. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3884–3894, Online. Association for Computational Linguistics.

Kevin Knight, Bianca Badarau, Laura Baranescu, Claire Bonial, Madalina Bardocz, Kira Griffitt, Ulf Hermjakob, Daniel Marcu, Martha Palmer, Tim O'Gorman, and Nathan Schneider. 2021. Abstract Meaning Representation (AMR) Annotation Release 3.0. Abacus Data Network.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: sequence-to-sequence models for parsing and generation. *CoRR*, abs/1704.08381.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Xiao Li, Gong Cheng, Ziheng Chen, Yawei Sun, and Yuzhong Qu. 2022. AdaLoGN: Adaptive logic graph network for reasoning-based machine reading comprehension. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7147–7161, Dublin, Ireland. Association for Computational Linguistics.

Jungwoo Lim, Dongsuk Oh, Yoonna Jang, Kisu Yang, and Heuiseok Lim. 2020. I know what you asked: Graph path learning using AMR for commonsense reasoning. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2459–2471, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Nut Limsopatham. 2021. Effectively leveraging BERT for legal document classification. In *Proceedings of the Natural Legal Language Processing Workshop 2021*, pages 210–216, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3622–3628. International Joint Conferences on Artificial Intelligence Organization. Main track.

Hugo A. López. 2021. Challenges in legal process discovery. In *Proceedings of the 1st Italian Forum on Business Process Management (ITBPM 2021), Rome, Italy, September 10th, 2021.*, CEUR Workshop Proceedings, pages 68–73. CEUR. Publisher Copyright: © 2021 CEUR-WS. All rights reserved.; 1st Italian Forum on Business Process Management, ITBPM 2021 ; Conference date: 10-09-2021.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization.

Manuel Mager, Ramón Fernandez Astudillo, Tahira Naseem, Md. Arafat Sultan, Young-Suk Lee, Radu Florian, and Salim Roukos. 2020. Gpt-too: A language-model-first approach for amr-to-text generation. *CoRR*, abs/2005.09123.

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. 2021. Generation-augmented retrieval for open-domain question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4089–4100, Online. Association for Computational Linguistics.

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

Nima Pourdamghani, Kevin Knight, and Ulf Hermjakob. 2016. Generating English from Abstract Meaning Representations. In *Proceedings of the 9th International Natural Language Generation conference*, pages 21–25, Edinburgh, UK. Association for Computational Linguistics.

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

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *OpenAI Blog*.

Leonardo F. R. Ribeiro, Yue Zhang, and Iryna Gurevych. 2021. Structural adapters in pretrained language models for amr-to-text generation. *CoRR*, abs/2103.09120.

Shamane Siriwardhana, Andrew Reis, Rivindu Weerasekera, and Suranga Nanayakkara. 2020. Jointly fine-tuning "bert-like" self supervised models to improve multimodal speech emotion recognition. *arXiv preprint arXiv:2008.06682*.

Linfeng Song, Daniel Gildea, Yue Zhang, Zhiguo Wang, and Jinsong Su. 2019. Semantic Neural Machine Translation Using AMR. *Transactions of the Association for Computational Linguistics*, 7:19–31.

Lai Dac Viet, Vu Trong Sinh, Le Minh Nguyen, and Ken Satoh. 2017. Convamr: Abstract meaning representation parsing for legal document. *CoRR*, abs/1711.06141.

Vern R Walker, Ji Hae Han, Xiang Ni, and Kaneyasu Yoseda. 2017. Semantic types for computational legal reasoning: propositional connectives and sentence roles in the veterans' claims dataset. In *Proceedings of the 16th edition of the International Conference on Articial Intelligence and Law*, pages 217–226.

Chaojun Xiao, Xueyu Hu, Zhiyuan Liu, Cunchao Tu, and Maosong Sun. 2021. Lawformer: A pre-trained language model for chinese legal long documents. *CoRR*, abs/2105.03887.

Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, and Vadim Sheinin. 2018. Graph2seq: Graph to sequence learning with attention-based neural networks. *CoRR*, abs/1804.00823.

Weiwen Xu, Yang Deng, Huihui Zhang, Deng Cai, and Wai Lam. 2021a. Exploiting reasoning chains for multi-hop science question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1143–1156, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Weiwen Xu, Huihui Zhang, Deng Cai, and Wai Lam. 2021b. Dynamic semantic graph construction and reasoning for explainable multi-hop science question answering. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1044–1056, Online. Association for Computational Linguistics.

Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. 2020. Reclor: A reading comprehension dataset requiring logical reasoning. In *International Conference on Learning Representations (ICLR)*.

Zdeněk Žabokrtský, Daniel Zeman, and Magda Ševčíková. 2020. Sentence meaning representations across languages: What can we learn from existing frameworks? *Computational Linguistics*, 46(3):605–665.

Zixuan Zhang and Heng Ji. 2021. Abstract Meaning Representation guided graph encoding and decoding for joint information extraction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 39–49, Online. Association for Computational Linguistics.

Lucia Zheng, Neel Guha, Brandon R. Anderson, Peter Henderson, and Daniel E. Ho. 2021. When does pretraining help? assessing self-supervised learning for law and the casehold dataset. *CoRR*, abs/2104.08671.

Haoxi Zhong, Chaojun Xiao, Cunchao Tu, Tianyang Zhang, Zhiyuan Liu, and Maosong Sun. 2020. How does NLP benefit legal system: A summary of legal artificial intelligence. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5218–5230, Online. Association for Computational Linguistics.

Logical Operator	English Expression
quantifier	all, any, nobody, some
negation	not, it is not the case that
conjunction	and, also, moreover, but
disjunction	or, unless
implication	if ... then, only if
equivalence	if and only if

Table 7: List of logical operators and some of their respective English expressions.

A Logic in AMR

The theoretical analysis covered examples of each logical operator from Table 7. Specifically, we went through examples of logical statements from LogiQA (Liu et al., 2020) and ReClor (Yu et al., 2020) that use the expressions from the list and see how they are represented in AMR. We choose examples form LogiQA and ReClor since they are logical reasoning tasks. The examples are in some cases slightly modified by e.g. removing subclauses to reduce some complexity from the AMR graphs. The AMRs have been created with the help of the Spring parser and validated by using resources such as the AMR guidelines[15] and the AMR Annotation Dictionary.[16] Especially the AMR annotation Dictionary has been used to check crucial parts of the graphs as it covers most of the expressions from the list. Additionally, we used the gold annotation from the AMR Annotation Release 3.0 (Knight et al., 2021) to get reference annotations of similar sentences.

Quantifiers Quantifiers are used in categorical reasoning. The goal of categorical reasoning is to see if a concept is part of a category. The following sentence is a so-called categorical proposition, meaning that it relates classes/categories to each other. In this case, it is stated that the class of people who love sweets is a subset of the class of people who love peppers. We will be looking into how the quantifier *everyone* is represented in AMR.

Everyone who loves sweets loves peppers.

[15] https://github.com/amrisi/amr-guidelines

[16] https://amr.isi.edu/doc/amr-dict.html

```
(l / love-01
    :ARG0 (e / everyone
        :ARG0-of (l1 / love-01
            :ARG1 (s / sweet)))
    :ARG1 (p / pepper))
```

In general, AMR handles pronouns as concepts, similar to how nouns are represented. This holds for indefinite pronouns such as everyone, everybody, nobody, etc. In this case `everyone` is `ARG0` of `love-01`, which annotates the role of the lover. The relative clause "who loves sweets" will be represented using the inverse role `ARG0-of`, putting the focus on *everyone*. Following two examples show the use of the quantifiers *all* and *some*.

All actors are exuberant.

```
(e / exuberant
    :domain (p / person
        :ARG0-of (a / act-01)
        :mod (a1 / all)))
```

Some inventors are American.

```
(p / person
    :ARG0-of (i / invent-01)
    :mod (c / country
        :wiki "United_States"
        :name (n / name
            :op1 "America"))
    :quant (s / some))
```

The quantifiers *all* and *some* are being represented with non-core roles `:mod` and `:quant`, respectively. The non-core roles are connected to the concept they are quantifying, in this case, `people`. When *some* is quantifying a noun it is seen as a non-exact quantity that uses the role `:quant`. When *all* is used as a quantifier it is represented using `:mod`. In general non-core roles are only used when core roles are not sufficient. To summarise, if a quantifier is a pronoun it will be represented in the same way as a regular noun. In case a quantifier is a determiner modifying a noun it will be represented with non-core roles such as `:quant` and `:mod`.

Conditional statements There are sufficient conditional statements that come in the form of "if p then q" or "q in case of p". In these statements, p is the antecedent and q is the consequence. Another type of conditional statement takes the following pattern "p only if q". In this case, q is a necessary condition for p. The following example shows such an instance.

Mark would go visit Tony only if they had an appointment.

```
(v / visit-01
    :ARG0 (p / person
        :name (n / name
            :op1 "Mark"))
    :ARG1 (p1 / person
        :name (n1 / name
            :op1 "Tony"))
    :condition (h / have-03
        :ARG0 (t / they)
        :ARG1 (a / appointment-02
            :ARG0 p
            :ARG1 p1)
        :mod (o / only)))
```

Compared to the first example from 7, the only difference in how AMR represents this conditional statement is the `:mod(o / only)` role under `:condition`. To summarise, conditional statements make use of the `:condition` role, which in the case of "only if" is further specified with the `:mod` role. Furthermore, the examples showed that negations are annotated with the `:polarity` role.

It should be noted that for `:condition` there is a reification in the form of `have-condition-91`. Reification is the transformation of non-core roles into first-class concepts. This can be done to put more focus on certain AMR fragments, but according to the AMR guidelines[17] there are no specific instructions on when to use it.

Disjunctive statements Disjunctive statements can be identified by the use of terms such as *or* and *unless*. They describe a sentence where two or more statements are in a disjunctive relationship. Following is an example of a disjunctive statement.

Mark either went to the gym or visited Tony.

```
(o / or
    :op1 (g / go-02
        :ARG0 (p / person
            :name (n / name
                :op1 "Mark"))
        :ARG4 (g1 / gym))
    :op2 (v / visit-01
        :ARG0 p
        :ARG1 (p1 / person
            :name (n1 / name
```

[17] https://github.com/amrisi/amr-guidelines/blob/master/amr.md

```
                :op1 "Tony"))))
```

In AMR disjunctions are represented with the `or` concept, which uses the `:op1`, `op2`, ... roles for each element. It can be noted that the *either* in this sentence is not explicitly annotated. The next example shows a statement with *unless*.

The crisis will not be overcome unless students spend more time studying.

```
(o / overcome-01
    :polarity -
    :ARG1 (c / crisis)
    :condition (s / spend-02
        :polarity -
        :ARG0 (p / person
            :ARG0-of (s1 /
                study-01))
        :ARG1 (t / time
            :quant (m / more))
        :ARG2 (s2 / study-01
            :ARG0 p)))
```

AMR annotates *unless* as a negative condition, by using `:condition` and `:polarity -`. This means that *unless* is interpreted as "if not" instead of "either ... or". When translating text to logical functions, "A unless B" can both be understood as $A \vee B$ and $\neg B \rightarrow A$ (Hurley, 2014). This means that AMR captures the conditional meaning of "unless", instead of the disjunctive meaning. To summarise, disjunctive statements of the form "p or q" use the `or` concept and statements with "unless" are interpreted as a conditional statement.

Equivalence Equivalence comes in the form of statements such as "p if and only if q". In this case, the statement is true if p and q have the same truth value.

A society should adopt democracy if and only if the people prefer a democracy.

```
(r / recommend-01
    :ARG1 (a / adopt-01
        :ARG0 (s / society)
        :ARG1 (d / democracy))
    :condition (p / prefer-01
        :ARG0 (p1 / person)
        :ARG1 d
        :mod (o / only)))
```

The AMR dictionary and guidelines do not give instructions on how to annotate a statement such as "p if and only if q". Furthermore, no similar example could be found in the AMR annotation release 3.0 (Knight et al., 2021) gold annotations which could be used to verify this annotation. It is therefore unclear how this statement should be annotated. The above annotation shows the result of the Spring AMR parser, which represents this statement in the same way as a necessary conditional statement of the form "q only if p".