# Dependency Patterns of Complex Sentences and Semantic Disambiguation for Abstract Meaning Representation Parsing

**Yuki Yamamoto*** **Yuji Matsumoto**** **Taro Watanabe***

*Nara Institute of Science and Technology
{yamamoto.yuki.yt0, taro}@is.naist.jp
**RIKEN Center for Advanced Intelligence Project
{yuji.matsumoto}@riken.jp

## Abstract

Abstract Meaning Representation (AMR) is a sentence-level meaning representation based on predicate argument structure. One of the challenges we find in AMR parsing is to capture the structure of complex sentences which expresses the relation between predicates. Knowing the core part of the sentence structure in advance may be beneficial in such a task. In this paper, we present a list of dependency patterns for English complex sentence constructions designed for AMR parsing. With a dedicated pattern matcher, all occurrences of complex sentence constructions are retrieved from an input sentence. While some of the subordinators have semantic ambiguities, we deal with this problem through training classification models on data derived from AMR and Wikipedia corpus, establishing a new baseline for future works. The developed complex sentence patterns and the corresponding AMR descriptions will be made public[1].

## 1 Introduction

Abstract Meaning Representation (AMR) is a sentence-level meaning representation based on predicate argument structure (Banarescu et al., 2013). AMR Parsing is the task of transforming a sentence into an AMR graph with nodes and edges, each representing a concept or relation. While early studies (Flanigan et al., 2014; Wang et al., 2015; Artzi et al., 2015; Pust et al., 2015) used dependency parsers to integrate syntactic features to their models, recent deep neural network-based approaches (Konstas et al., 2017; Peng et al., 2017; Zhang et al., 2019; Cai and Lam, 2020) tend to encode the input sentence as a sequence without considering its syntactic structure.

Generally speaking, syntactic and semantic structures share much in common. It is assumed that dependency trees and semantic role labeling structures have a strong correlation in that *nsubj* and *dobj* can be used interchangeably for ARG0 and ARG1 role (Xia et al., 2019). Since AMR is annotated based on PropBank frames (Palmer et al., 2005), the same could be said for AMR structures.

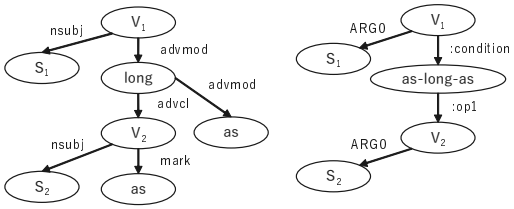



Figure 1: Representation of *as long as*-construction in dependency tree (left) and AMR graph (right).

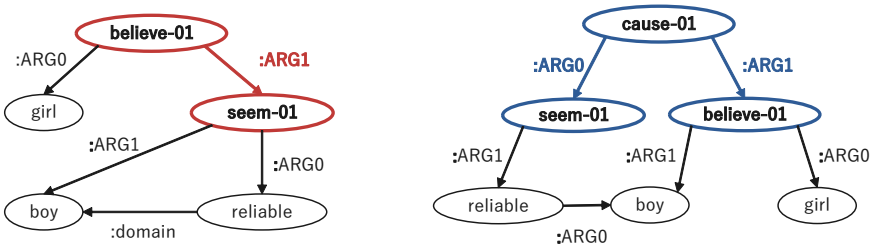
This holds to be true for a simple sentence, which is basically a matrix clause, comprised of a predicate and its arguments. However, it is not always the case with complex sentence constructions, each of which consists of a matrix clause and one or more subordinate clause(s). Consider Figure 1 which shows both dependency and AMR representation of a complex sentence with a subordinator *as long as*. While variables `S`'s and `V`'s are interchangeable between the representations, predicative relations and subordinator itself are expressed quite differently. Compared to uniform structures of simple sentences, various types of complex sentence are used in human language. This characteristics makes it challenging for existing AMR parsers to capture its structure correctly.

Among AMR parsers which are aware of syntactic structures, CAMR (Wang et al., 2015) directly transforms the result of dependency parsing into an AMR graph with transition-based algorithm. As Figure 2(a) shows an example parse with CAMR, existing parsers have trouble capturing the relation

[1]Code and resource are available at https://github.com/yama-yuki/skeletal-amr.



*Proceedings of the 10th Conference on Lexical and Computational Semantics*, pages 212–221
August 5–6, 2021, Bangkok, Thailand (online) 




(a) Parser's Output AMR (b) Correct AMR

Figure 2: Result of parsing "As the boy seemed reliable, the girl believed him." with CAMR.

between two clauses. In this case, CAMR predicts the relation between `believe-01` and `seem-01` as ARG1, though it should be represented as a causal relation with `cause-01` as shown in Figure 2(b). This is a crucial error as it incorrectly determines the very core structure of the output graph. We assume that the solution could be achieved either by retraining the parser on larger annotated data or providing the parser with the core structure of the input sentence in advance. The latter seems more reasonable in terms of the cost of annotation.

With the motivation to aid AMR parsing task, we present a method to retrieve all occurrences of complex sentence constructions from an input sentence using a dedicated pattern matcher. At the moment, there is no comprehensive resource that provides structural information about the relations between two clauses, particularly in AMR framework. Therefore, as our first step, we attempt to develop a pattern dictionary of English complex sentences together with the corresponding AMRs which represent the skeleton structure of the sentence (hereinafter referred to as "skeletal AMRs"). Then, we provide a pattern matcher which captures clausal relations between a superordinate and subordinate clauses in a complex sentence.

Our pattern matching approach faces the problem of syntactic and semantic ambiguities. When a complex sentence has more than one subordinate clause, we need to determine which pair of clauses are related. Consider the following example where two subordinate clauses appear in a single sentence.

(1) ... [if you wish to look at the comparative risks]$_{SUB1}$ [if we do not confront terrorist organizations in their staging areas]$_{SUB2}$, [how many people could die as a result of weapons of mass destruction at some point in the not too distant future]$_{MAT}$? (AMR: bolt-eng-DF-199-192783-6849434_0102.3)

While there has been studies regarding the syntactic scope of a subordinate clause such as Utsuro et al. (2000), this problem is beyond the scope of this paper. We rely on the output of the dependency parser, which we employ in our pattern matching system, to decide which pair of clauses are syntactically related.

Meanwhile, when a subordinator itself is ambiguous between several senses, we need to select the correct type of coherence relation between the clauses. Sentences in (2) show usages of a subordinator *since*, which is semantically ambiguous between causal and temporal senses.

(2) a. Since there is responsibility, we are not afraid. (AMR: bolt12_6455_6561.15)

b. Also since he turned 80, people had been paying more and more attention to Mao Zedong's birthday. (AMR: bolt12_10511_7302.5)

In order to resolve the semantic ambiguities of clause-level coherence relation, inspired by the work of Shi and Demberg (2019), we take finetuning-based approach with data augmentation method. With the support of weakly supervised data derived from Wikipedia, we achieve the scores of 75.65% and 83.94% on macro and micro F1 respectively, establishing a new baseline for coherence relation classification of complex sentence constructions in AMR framework.

To sum up, the contributions of this paper are the followings:

- We create a comprehensive resource of complex sentence constructions.
- We develop a pattern matching system which takes a sentence as an input and returns a corresponding skeletal AMR.

- We establish a new baseline for semantic disambiguation task of complex sentence constructions in AMR framework.

## 2 Related Works

While our focus is on clause-level relation of complex sentence constructions, not much study has been done specifically on this topic in AMR framework. Rather, the topic is dealt with in the field of discourse structures, where coherence relations between any text segments are the main focus.

In the studies of discourse parsing, various attempts have been made to capture coherence relations between pairs of sentences or clauses (Pitler et al., 2008; Rutherford et al., 2017; Qin et al., 2017; Bai and Zhao, 2018). These works basically rely on discourse frameworks such as Rhetorical Structure Theory (RST; Thompson and Mann 1987) or Penn Discourse Tree Bank (PDTB; Prasad et al. 2008).

Most recently, Shi and Demberg (2019) has presented a finetuning-based approach using the bidirectional encoder representation from transformers (BERT; Devlin et al. 2019). They designed their model to learn 11 classes to achieve the state-of-the-art performance on implicit discourse relation classification task in PDTB framework.

Their work was motivated by the method taken by Devlin et al. (2019) to pretrain BERT, which is called "next sentence prediction task" (NSP). In the process of pretraining using NSP, the model is presented with pairs of sentences. The model predicts whether the second sentence is the actual subsequent sentence. NSP enables BERT to represent a pair of sentences by packing them together as a single sequence.

Some studies have focused on discourse structure in AMR framework. Donatelli et al. (2018) enhances AMR by annontating tense and aspect phenomena at discourse-level. The work by O'Gorman et al. (2018) targets relations of sentences and provides annotation of coreference in multi-sentence AMR corpus. Yet, neither the structure of the complex sentence constructions nor the coherence relations between subordinate and matrix clauses have been much of a concern in this framework.

## 3 Pattern Matching System

In this section, we will give a description of our pattern matching system. The entire workflow is illustrated in Figure 3, where the system takes a sentence as an input and returns a skeletal AMR if it is a type of complex sentence construction. The disambiguation module will be described in the later section. We will use the sentence given as a running example throughout this paper.

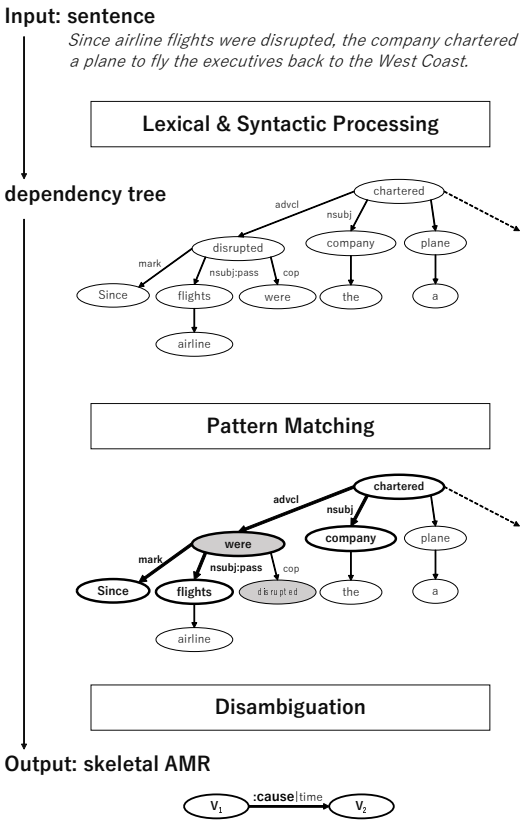



Figure 3: Pattern matching system workflow for a sentence "Since airline flights were disrupted, the company chartered a plane to fly the executives back to the West Coast.".

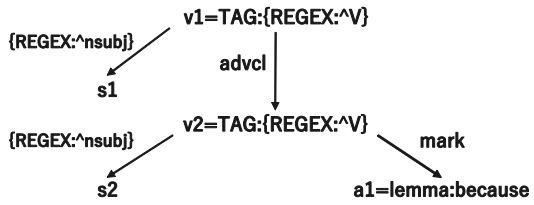
### 3.1 Dictionary of Dependency Patterns

Complex sentence constructions in English grammar can be distinguished by their consisting subordinator. When creating a pattern dictionary, we refer to comprehensive studies on grammars (Quirk et al., 1985; Yamaguchi, 2013), which provide a typology of subordinators. To cover various types of constructions, we include simple (e.g. *if, because*), complex (e.g. *as if, so that*) and correlate (*e.g. no sooner ... than*) subordinators classified in Quirk et al. (1985). In addition to that, we also include the type of constructions involving degree and quantity which are introduced to AMR in Bonial et al. (2018).

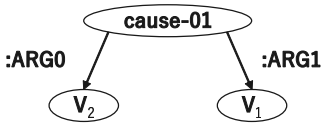
Our pattern matching method depends not only

on lexical processing but also on syntactic processing. For syntactic framework, we follow the annotation of dependency structure in Universal Dependencies v2 format (UD; Nivre et al. 2020), which makes our patterns a form of dependency trees. The nodes in trees are represented either by a lemma form of a lexical entry or an abstract one defined by POS tags, where every edge has a dependency relation label. Regular expression is employed to place any form of a specific element. We provide these dependency patterns with a corresponding skeletal AMR that describes the core structure of an input sentence. Depending on the type of construction, a set of variables are used to take alignments between patterns and skeletons. At the moment, our dictionary includes 70 distinct patterns[2].

Figure 4 illustrates an example of a paired entry of a dependency pattern and skeletal AMR for *because*-construction:




(a) Dependency Pattern

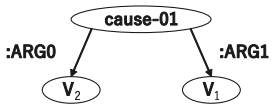



(b) Skeletal AMR

Figure 4: Pattern dictionary entry of complex sentence construction with simple subordinator *because*.

The dependency pattern in Figure 4(a) describes the simplest type of structure in our dictionary. The nodes that are represented as variables `s1` and `v1` capture a predicate and its argument in a matrix clause, whereas `s2` and `v2` capture a subordinate clause. Subordinators are basically described as `a`'s, where their lemma forms are given as cues. `REGEX` operators are used on both nodes and edges to flexibly match possible elements. For example, the dependency relation of `{REGEX:^nsubj}` between `s`'s and `v`'s enables us to handle subjects of both the active (`nsubj`) and passive (`nsubj:pass`) voice. In the case of `TAG:{REGEX:^V}`, `REGEX` is used to represent any POS tag that starts from "`V`", meaning that it captures any form of a verb. Figure 4(b) shows the corresponding skeletal AMR, which represents a core relation between the two predicates. In all patterns in the dictionary, $\texttt{V}_1$ and $\texttt{V}_2$ act as slots for the predicates of the matrix and subordinate clauses.

Some subordinators could indicate more than one coherence relations. For example, Figure 5 shows the case of our running example with *since*, where the subordinator presents either (a) causal or (b) temporal relation:

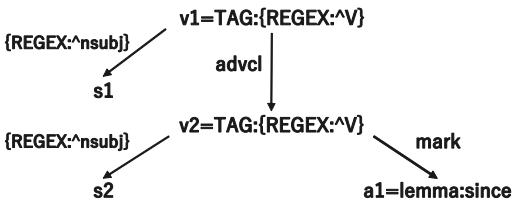



(a) Causal


(b) Temporal

Figure 5: Skeletal AMRs of ambiguous *since*.

In the pattern dictionary, rather than enumerating entries for every possible structure, we describe a primitive structure of skeletal AMR to dynamically generate the actual relation, which is in accordance with the Generative Lexicon approach (Pustejovsky, 1995). More specifically, we accomodate all possible relations separating them with a vertical bar on an edge of a skeletal AMR in the way described in Figure 6(b), which we call a "primitive skeleton":




(a) Dependency Pattern


(b) Skeletal AMR

Figure 6: Pattern dictionary entry of complex sentence construction with simple subordinator *since*.

[2]We make it available for users to add new patterns to the dictionary for further expansion.

The disambiguation step will follow after the pattern matcher returns the primitive skeleton. For simplicity and consistency, we use *shortcut*s to represent the relations if available, such as using a relational role `:cause` instead of a predicative frame `cause-01` that can be substitutionally used.

Aside the semantic ambiguity of relation mentioned above, structural ambiguity could be seen when several types of structure exists within the same relation. Compare Figure 5(b) and 7, where both skeletal AMRs represent temporal relation but show different structures:

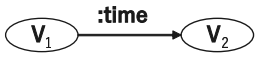



Figure 7: Skeletal AMR of *as*, *once*, etc. representing temporal relation.

Since the primitive skeleton only describes the simplest form, recovery step needs be taken to generate the final skeletal AMR for ambiguous subordinators. Therefore, we define a possible interpretation (such as Figure 5(b) and 7) for primitive skeleton of each ambiguous subordinator, which will be referred to after the skeletal AMR is disambiguated. This idea share similarity with the notion of *meaning postulates* (Carnap, 1952; Dowty, 1979). By following the definition, the structural ambiguity of skeletal AMRs will be resolved.

### 3.2 Pattern Matching Method

Our pattern matching method builds on the dependency matching module introduced by Honnibal et al. (2020) that matches subtrees within a dependency tree. The matcher works in naive manner, searching from the top to the bottom of the pattern dictionary. It is originally capable of handling recursive nature of clauses. Namely, even when an input sentence has more than two clauses, the matcher searches all possible patterns for all clause pairs in a single run.

Meanwhile, it suffers from the following situations: "overlap" where multiple patterns accidentally match with a pair of clauses and "copula" where the matcher fails to capture copular constructions due to the limitation of expressiveness of patterns. To account for these cases, we extend the matching method by incorporating additional functions.

**Overlap** Duplicated matching may occur within a single pair of clauses since some patterns share their forms.

(3) As if he were still in his old job, Mr.Wright enjoys a $120,000 annual office expense allowance.

When the target sentence is (3), the dependency matcher returns the following overlapping patterns:

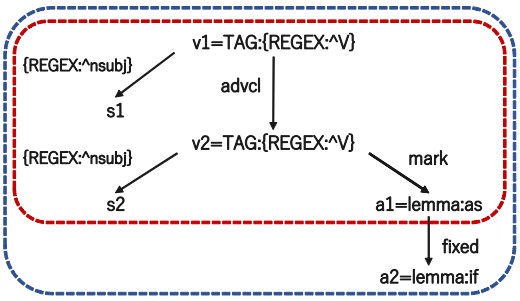



Figure 8: Overlap of patterns for *as* and *as if*.

This mostly occurs between simple and complex subordinators in Quirkian term (e.g. *as* and *as far as*, *if* and *even if*, etc.). To address this issue, we assign an ID for each entry in the pattern dictionary to refer to the type of subordinators. For example, *as* is assigned "#1.6.1", whereas *as if* is "#2.2.1". The first number in an ID represents the number of words consisting the subordinator. The second describes its sorted order while the third is organized by the structure of the pattern. When the matcher detects an overlap, it looks up the IDs to select the most desirable output. In the case of overlap between simple and complex subordinators, the matcher compares the first number in ID, selecting the pattern with more words. In this case, *as if* is regarded as the desirable pattern.

**Copula** Verbs that show a relationship between subjects and objects are referred to as copulas (e.g. "is" for "John is tall."). Since UD format refrains from using a copula as a head of its complement[3], the matcher cannot find a copular clause with complex sentence patterns alone. To avoid redundancy of creating additional patterns substituting `V`'s with copulas for each entry in dictionary, we make an extra pattern outside that describes a copula-complement structure:

[3]For "John is tall.", UD treats "tall" as a head of "is".

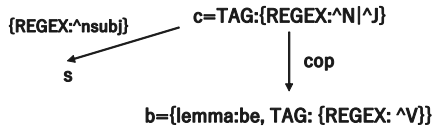



Figure 9: Dependency pattern of copular construction.

With the implementation of a converter, our system can detect copular clauses in complex sentences and convert it to a copular-headed structure when matched, in the way illustrated in the gray nodes in Figure 3.

## 4 Semantic Disambiguation Experiment

In this section, we describe details of our study on semantic disambiguation. We observe that the coherence relation between subordinate and superordinate clauses are commonly ambiguous among causal, conditional, concessional, and temporal senses. Considering that we are dealing with relations between clauses, which basically can be regarded as pairs of simple sentences, we can cast the problem of semantic ambiguity as a multi-class sentence-pair classification task. To be more specific, we assign the most typical class labels including CAUSE, COND, CONC, and TIME to make it a 4-class classification setting.

Throughout the experiment, we use the pretrained "bert-base-uncased" model for finetuning. BERT is pretrained under next sentence prediction as well as masked language modeling task. During pretraining, the model predicts the actual next sentence from a pair of candidate sentences. Thus, the model is expected to learn what the next sentence should look like. By finetuning BERT on a set of pairs of clauses, we can further expect the model to capture coherence relations between them.

Our input will be a separate pair of clauses with special tokens "[CLS]" and "[SEP]". While BERT takes the input in sequential manner, we would like the model to take advantage of dependency parsing, which we apply in the process of pattern matching. Therefore, we add artificial tags of " < " and " > " to indicate the head verb of each clause. The input format of our running example would look like: "[CLS] airline flights < were > disrupted [SEP] the company < chartered > a plane to fly the executives back to the West Coast [SEP]".

### 4.1 Experimental Setup

For creating the dataset, we use the lateset release of AMR corpus (LDC2020T02), which provides 59.2k pairs of sentences and AMR graphs. To extract complex sentence constructions from the corpus, we use the STANZA pipeline (Qi et al., 2020) for lexical and syntactic processing of the sentences and employ our pattern matcher with all patterns in the dictionary. In order to check whether the corresponding AMR graph describes the relation we want for each class, we look for alignments between sentence tokens and AMR graphs[4][5]. Finally, we split the sentence to obtain a pair of clauses and a subordinator.

While the data derived from AMR corpus can be regarded as "supervised", the amount is relatively small with the total of 1,933 pairs of subordinate and matrix clauses. As it consumes time and money to create more supervised data, we take weak supervision approach to augment training data. In other words, using specific subordinators that are known to be unambiguous in AMR corpus as linguistic cues, we seek to obtain complex sentences from larger corpus. We use raw data from Wikipedia for data augmentation to generate "weakly supervised" data. The list of subordinators used in this method is shown in Table 1. The distribution of each data, which we will refer to as AMR and WIKI data, is illustrated in Table 2. The data comprises *(subordinate clause, superordinate clause, subordinator, class label)* quadruplets.

| Label | Subordinators |
|---|---|
| CAUSE | *because* |
| COND | *if, unless* |
| CONC | *although, though, even if* |
| TIME | *once, whenever* |

Table 1: Subordinators used to create weakly supervised data (WIKI data).

| Label | AMR | WIKI |
|---|---|---|
| CAUSE | 442 | 27,264 |
| COND | 1,002 | 26,756 |
| CONC | 75 | 46,213 |
| TIME | 414 | 19,558 |
| Total | 1,933 | 119,791 |

Table 2: Distributions of labels in AMR and WIKI data.

[4]We use the alignments provided in the corpus that are automatically generated.

[5]We only target the simple structure such as the one presented in Figure 7, which we assume represent typical relation for each class.

Cosidering the size of AMR data, we make 5 splits[6] of the data to perform 5-fold cross validation. When training on WIKI data, we average the scores of 5 runs with different random seeds to ensure stability. In addition, we perform grid search for hyperparameter tuning over the options in Table 3. All models will be evaluated using both Macro and micro F1 scores ($F_M$ and $F_m$, respectively). Variances will be given in parentheses.

| Hyperparameter | Values |
|---|---|
| batch size | 16, 32, 64 |
| learning rate | 2e-05, 3e-05, 5e-05 |
| epochs | 3, 5, 10 |

Table 3: Hyperparameter options for grid search.

### 4.2 Baseline Model

As our baseline, we finetuned BERT solely on AMR data. To make best use of the data, we compared BERT→AMR with BERT→AMRs (with subordinators as a feature). One of BERT's "unused tags" is placed to specify the position of a subordinator in an input sequence: "[CLS] [unused_0] *subordinator* [unused_0] *subordinate clause* [SEP] *matrix clause* [SEP]". As presented in Table 4, it turned out that explicit information of subordinators was not effective in our experiment, as opposed to the general tendency in *explicit vs implicit* connective settings (Pitler et al., 2008). Therefore, we choose BERT→AMR as our baseline and seek for other ways to make use of subordinators.

| Models | $F_M$ | $F_m$ |
|---|---|---|
| BERT→AMR | 64.06(±.06) | 74.29(±.03) |
| BERT→AMR*s* | 22.43(±.02) | 54.77(±.03) |

Table 4: Performance of baseline models.

## 5 Use of Weakly Supervised Data

### 5.1 Approaches

In the experiments, we take several approaches to examine the effect of augmentated data on classification performance. For direct comparison of the training data, BERT→WIKI is solely finetuned on WIKI data to see whether the model would benefit from the larger weakly supervised data. BERT→MIX is finetuned on a combined set of data which consists of AMR data and certain amount of additional WIKI data. The amount of WIKI data used ranges from 2∼20k, where we add 2k sentences[7] at a time . We take this approach with an expectation that WIKI data would complement the inbalanced distribution of AMR data to perform better than the baseline or BERT→WIKI. Finally, we evaluate the model marked as BERT→WIKI→AMR which is first finetuned on WIKI data, then further finetuned on AMR data. This is conducted under our hypothesis that "prefinetuning" on WIKI data would make BERT model fit to our task than the original model, which is just pretrained on next sentence prediction task.

### 5.2 Results and Analyses

| Models | $F_M$ | $F_m$ |
|---|---|---|
| BERT→AMR (baseline) | 64.06(±.06) | 74.29(±.03) |
| BERT→WIKI | 47.67(±.00) | 61.72(±.00) |
| BERT→MIX$_{8k}$ | 67.12(±.01) | 77.50(±.00) |
| **BERT→WIKI→AMR** | **72.43**(±.**02**) | **81.22**(±.**00**) |

Table 5: Performance of each approach. Only the best performing model is presented for BERT→MIX.

The scores of all approaches are presented together in Table 5. The results show that training on weakly supervised data by itself does not improve the baseline, with BERT→WIKI harming the performance by 16.39% points on $F_M$ and 12.57% points on $F_m$. This may be attributed to the gap of construction types between WIKI and AMR data. Meanwhile, we see improvements of 3.06% and 3.21% when we add 8k amount of WIKI data to AMR data. The transition of scores by the amount of added data is presented in Figure 10. Both $F_M$ and $F_m$ show an increase as we combine AMR and WIKI data until they reach the peak at 8k. Further addition only lead to drop the model's performance. This suggests that too much amount of WIKI data dilutes the presence of supervised AMR data. We could predict that training on full addition of WIKI data would deteriorate its performance near to that of BERT→WIKI. Among all our approaches, BERT→WIKI→AMR achieved the best results with 8.37% and 6.93% increase as expected. This proves the effectiveness of prefinetuning on weakly supervised data when you have supervised data of a small size.

Additionally, we checked whether our preprocessing step of adding artificial tags ("<" and ">") in section 4 helped the model's performance. With

[6]Train:Dev:Test=3:1:1 for each split.

[7]Balanced data with 0.5k for each class label.

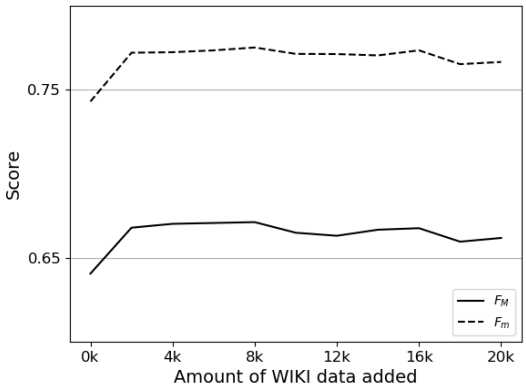



Figure 10: Score transitions of BERT→MIX by amount of WIKI data added. "k" stands for 1,000.

untagged version of BERT→WIKI→AMR scoring 71.21% for $F_M$ and 80.47% for $F_m$, we see 1.22% and 0.75% increase for tagged version. We find that the improvements achieved are not as much as utilizing weakly supervised data, but still beneficial to some extent considering it is a by-product of dependency matching.

While it was not effective to use subordinators as features, it remains reasonable to take advantage of its information. Therefore, we make modifications to the trained models in Table 5. We first create a list of ambiguous subordinators and their possible labels (e.g. *since* is ambiguous between CAUSE and TIME). When we feed a complex sentence, its subordinator will be searched in the list to check whether it is potentially ambiguous. If it turns out to be true, a restriction will be applied to the softmax layer by lowering the probability of irrelevant class labels to 0. In other words, the models are modified to only look at possible labels defined in the list.

| Models | $F_M$ | $F_m$ |
|---|---|---|
| BERT→AMR+$r$ | 67.11(±.05) | 77.18(±.03) |
| BERT→MIX$_{8k}$+$r$ | 70.76(±.01) | 80.52(±.00) |
| **BERT→WIKI→AMR+$r$** | **75.65**(±.**03**) | **83.94**(±.**01**) |

Table 6: Performance with subordinator as restrictions on softmax layer.

The results after applying the restriction are shown in Table 6 with +$r$ in model names. Compared to the performance in Table 5, we achieve overall improvements of 3.05∼3.64% on $F_M$ and 2.72∼3.02% on $F_m$ for all models. Table 7(a) and 7(b) present precision, recall, and F1 of BERT→AMR and BERT→AMR+$r$ by labels.

The approach seems mostly effective except for CONC where the precision decreases by 5.75%. With further analysis on confusion matrices on the first split of data for cross validation in Table 8, we find the number of false positive errors of CONC increased. This is due to an error predicting sentences with *while*, which we regard ambiguous between CONC and TIME. When the correct label is TIME, the model first predicted CAUSE or COND. Even after the restriction was applied, the model predicted CONC. Overall, the restriction seems to help the models reduce false positive errors.

(a) BERT→AMR

| Labels | $P$ | $R$ | $F$ |
|---|---|---|---|
| CAUSE | 60.51(±.07) | 68.93(±.04) | 64.34(±.04) |
| COND | 84.00(±.02) | 84.91(±.03) | 84.43(±.02) |
| CONC | 48.93(±.46) | 43.16(±.19) | 45.44(±.25) |
| TIME | 72.96(±.21) | 54.17(±.05) | 62.01(±.08) |

(b) BERT→AMR+$r$

| Labels | $P$ | $R$ | $F$ |
|---|---|---|---|
| CAUSE | 63.71(±.07) | 73.19(±.03) | 68.00(±.03) |
| COND | 89.17(±.02) | 84.91(±.03) | 86.97(±.02) |
| CONC | 43.18(±.21) | 43.16(±.19) | 43.07(±.19) |
| TIME | 73.25(±.18) | 67.92(±.06) | 70.40(±.10) |

Table 7: Performance by labels.

(a) BERT→AMR

| | Predicted Labels | | | |
|---|---|---|---|---|
| True Labels | CAUSE | COND | CONC | TIME |
| CAUSE | 55 | 15 | 4 | 6 |
| COND | 10 | 179 | 1 | 6 |
| CONC | 4 | 5 | 12 | 1 |
| TIME | 10 | 12 | 0 | 26 |

(b) BERT→AMR+$r$

| | Predicted Labels | | | |
|---|---|---|---|---|
| True Labels | CAUSE | COND | CONC | TIME |
| CAUSE | 57 | 11 | 3 | 9 |
| COND | 10 | 179 | 1 | 6 |
| CONC | 4 | 5 | 12 | 1 |
| TIME | 7 | 3 | 5 | 33 |

Table 8: Confusion matrices of models trained on the first split of data for cross validation.

## 6 Conclusion

With the intention to capture the structure of complex sentence constructions in AMR framework, we proposed a pattern matching method using a list of dependency patterns and its corresponding

skeletal AMRs. In the course of creating a comprehensive pattern dictionary, we observed semantically ambiguous entries. In order to resolve the semantic ambiguities, we framed the problem as a sentence-pair classification task and finetuned pretrained BERT models on data derived from AMR and Wikipedia corpus. Through the experiments, we found that the supplemental usage of weakly supervised data generated from Wikipedia effectively improves performance of the models compared to the one trained solely on small-sized supervised data. A natural next step will be to incorporate the presented method in AMR parsing task, which we leave for future work.

## References


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

Hongxiao Bai and Hai Zhao. 2018. Deep enhanced representation for implicit discourse relation recognition. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 571–583, Santa Fe, New Mexico, USA. 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.

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

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

Rudolf Carnap. 1952. Meaning postulates. *Philosophical Studies: An International Journal for Philosophy in the Analytic Tradition*, 3(5):65–73.

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

Lucia Donatelli, Michael Regan, William Croft, and Nathan Schneider. 2018. Annotation of tense and aspect semantics for sentential AMR. In *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, pages 96–108, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

David Dowty. 1979. *Word Meaning and Montague Grammar*. Reidel, Dordrecht.

Jeffrey Flanigan, Sam Thomson, Jaime G Carbonell, Chris Dyer, and Noah A Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python. Zenodo.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. *meeting of the association for computational linguistics*.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal dependencies v2: An evergrowing multilingual treebank collection. *arXiv preprint arXiv:2004.10643*.

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.

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

Xiaochang Peng, Chuan Wang, Daniel Gildea, and Nianwen Xue. 2017. Addressing the data sparsity issue in neural AMR parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 366–375, Valencia, Spain. Association for Computational Linguistics.

Emily Pitler, Mridhula Raghupathy, Hena Mehta, Ani Nenkova, Alan Lee, and Aravind Joshi. 2008. Easily identifiable discourse relations. In *Coling 2008: Companion volume: Posters*, pages 87–90, Manchester, UK. Coling 2008 Organizing Committee.

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 2008)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Parsing English into Abstract Meaning Representation using syntax-based machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1143–1154, Lisbon, Portugal. Association for Computational Linguistics.

James Pustejovsky. 1995. *The Generative Lexicon*. MIT Press, Cambridge, MA.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Lianhui Qin, Zhisong Zhang, Hai Zhao, Zhiting Hu, and Eric Xing. 2017. Adversarial connective-exploiting networks for implicit discourse relation classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1006–1017, Vancouver, Canada. Association for Computational Linguistics.

Randolph Quirk, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. 1985. *A Comprehensive Grammar of the English Language*. Longman, London.

Attapol Rutherford, Vera Demberg, and Nianwen Xue. 2017. A systematic study of neural discourse models for implicit discourse relation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 281–291, Valencia, Spain. Association for Computational Linguistics.

Wei Shi and Vera Demberg. 2019. Next sentence prediction helps implicit discourse relation classification within and across domains. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5790–5796, Hong Kong, China. Association for Computational Linguistics.

S. A. Thompson and W. Mann. 1987. Rhetorical structure theory: A framework for the analysis of texts.

Takehito Utsuro, Shigeyuki Nishiokayama, Masakazu Fujio, and Yuji Matsumoto. 2000. Analyzing dependencies of Japanese subordinate clauses based on statistics of scope embedding preference. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics*.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 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*, pages 366–375.

Qingrong Xia, Z. Li, Min Zhang, Meishan Zhang, G. Fu, Rui Wang, and Luo Si. 2019. Syntax-aware neural semantic role labeling. In *AAAI*.

Shunji Yamaguchi. 2013. *Eigo Koubun Zenkaisetsu* [The Perfect Study on English Sentence Structures]. Kenkyusha.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019. Broad-coverage semantic parsing as transduction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3786–3798, Hong Kong, China. Association for Computational Linguistics.