Modal Dependency Parsing via Biaffine Attention with Self-Loop

Jayeol Chun
Brandeis University
415 South Street, Waltham, MA 02453
jchun@brandeis.edu

Nianwen Xue
Brandeis University
415 South Street, Waltham, MA 02453
xuen@brandeis.edu

Abstract

A modal dependency structure represents a web of connections between events and sources of information in a document that allows for tracing of who-said-what with what levels of certainty, thereby establishing factuality in an event-centric approach. Obtaining such graphs defines the task of modal dependency parsing, which involves event and source identification along with the modal relations between them. In this paper, we propose a simple yet effective solution based on biaffine attention that specifically optimizes against the domain-specific challenges of modal dependency parsing by integrating self-loop. We show that our approach, when coupled with data augmentation by leveraging the Large Language Models to translate annotations from one language to another, outperforms the previous state-of-the-art on English and Chinese datasets by 2% and 4% respectively.

1 Introduction

At a time when we find ourselves inundated with endless streams of new information and knowledge, being able to identify a source of information and the confidence level with which it is conveyed is often helpful—if not sometimes critical—for better understanding the context behind a text or discourse. Modal dependency structure (MDS) (Vigus et al., 2019) is designed with such representation in mind, where the events and the sources (also known as *concievers*[1]) take the center stage as the vertices of the graph, while the edges denote (1) source of factualiy via its direction and (2) level of certainty via its label, which is a combination of 3 modal strengths (*Full*, *Partial*, and *Neutral*) and 2 polarities (*Affirmative* and *Negative*) based on the annotation scheme from FactBank (Saurí and Pustejovsky, 2009).

[1]In what follows, 'conceivers' are preferred over 'sources.'

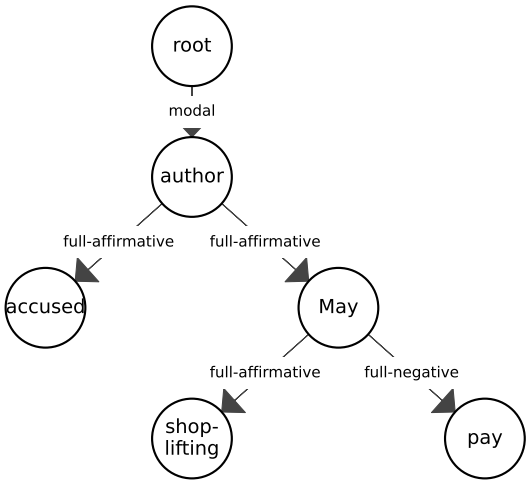


Figure 1: Example of Modal Dependency Graph for the document: "May accused the couple of shoplifting. According to her, they had not paid for their ring."

Figure 1 shows an example of a modal dependency graph for a sample document of two sentences:

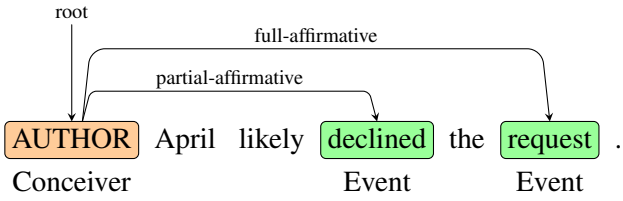
(1) a. May accused the couple of shoplifting.

b. According to her, they had not paid for their ring.

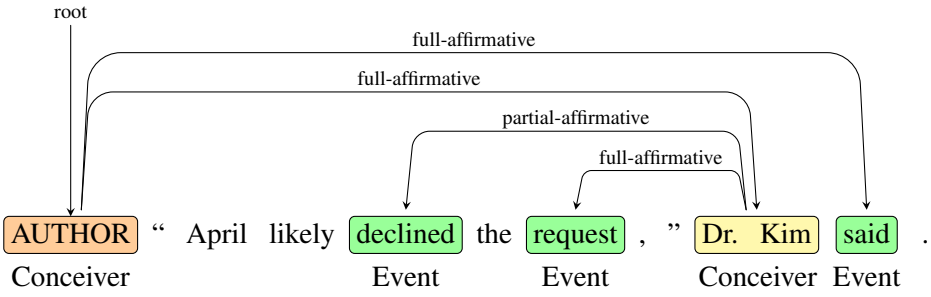
An abstract root node at the top ensures that the structure is single-rooted. Immediately below is an abstract author node, whose presence is implicitly presumed for every document as its creator. In general, an MDS typically shows heavy traffic through the author node as a principal conceiver of various events in the document.

In the first sentence, the author states that May's accusation did take place without any hesitation. This is represented in the MDS with a '`:full-affirmative`' edge between the author and the '`accused`' event node. If it were later revealed that May in fact *never accused* the couple

Findings of the Association for Computational Linguistics: ACL 2025, pages 21226–21238



(a) Second sentence: April likely declined the request.



(b) First sentence: “April likely declined the request,” Dr. Kim said.

Figure 2: Example of Modal Dependency Tree visualization for two sentences. Orange node represents the abstract author node of the document. Green and yellow nodes indicate the events and conceivers respectively.

of shoplifting, we may conclude that the author is responsible for providing the false information.

The author then relays a comment made by May in the second sentence, indicating that the cited statement does not necessarily reflect the author's point of view. This form of reporting, wherein the author (or some conceiver) simply reiterates information from another source, is an important phenomenon in the MDS that is represented with a conceiver-to-conceiver triple (‘author’ :full-affirmative ‘May’). Since it is May who claims that shoplifting by the couple *did happen* (because it is her allegation) and their paying for the ring *did not*, the event nodes ‘shoplifting’ and ‘pay’ are subsequently linked to ‘May’ conceiver node with ‘:full-affirmative’ and ‘:full-negative’ edge labels respectively.

Figure 2 illustrates this point further by comparing a simple statement (Figure 2a) against a quotation (Figure 2b). In Figure 2a, the author does not rely on any external sources; hence, the author is the conceiver of the events ‘declined’ and ‘request’. In addition, the use of the term ‘likely’ indicates that the author is only partially certain about the ‘declined’ event, justifying the assignment of the modal edge label ‘:partial-affirmative.’

In contrast, Figure 2b shows the same sentence as a quotation made by Dr. Kim. It follows that the events embedded within the quotation ‘declined’ and ‘request’ should be connected to ‘Dr. Kim’ who is the author of the quoted statement and hence the conceiver of the said events. It is important to note that, while April serves as the protagonist in both instances, she is not regarded as a conceiver because she is not cited in any capacity.

In order to obtain such a modal dependency tree[2] from text, modal dependency parsing (MDP) needs to perform a few different tasks. First, spans of events and conceivers must be identified and labeled accordingly. Second, the modal relations must be established for the identified spans by predicting the correct modal arc and edge label.

To tackle this problem, we present a simple yet effective solution in the form of a biaffine attention with added support for self-loop. The merits of our approach are as follows:

- The context scope is global.
- The introduction of self-loop allows the events and conceivers to be discovered by the same biaffine module that also generates modal arcs and edge labels, leading to a highly efficient multi-tasking setup that requires a single forward pass over the entire document.
- The model closely follows the logical order of the annotation of the modal structure during

[2]In general, MDS forms a tree not a graph.

decoding, where the conceiver identification depends on the identification of child events.

We further experiment with data augmentation by leveraging the Large Language Models (LLMs) to translate annotations in English or Chinese into the other language while preserving the annotated spans of modal nodes and edges, resulting in a significantly increased number of training samples. Our experiments show that the proposed approach significantly outperforms the previous state of the art by 2% for English and 4% for Chinese. The code is available at https://github.com/umr4nlp/mdp_biaffine.

2 Related Work

Traditionally, event factuality prediction (EFP) was seen as a classification or regression problem that involved rule-based (Nairn et al., 2006; Lotan et al., 2013) or statistical approaches (Diab et al., 2009; Saurí and Pustejovsky, 2012; Lee et al., 2015; Stanovsky et al., 2017). With widespread adoption of deep learning came a surge of neural models tackling this problem, for instance based on LSTMs (Rudinger et al., 2018), GANs (Qian et al., 2018) or GNNs (Pouran Ben Veyseh et al., 2019).

Yao et al. (2021) is the first work that casted EFP as modal dependency parsing and reported baseline results on English, while releasing the crowdsourced dataset which is publicly available[3]. This was followed up by a prompt-based model (Yao et al., 2022) with the first reported results on Chinese MDP trained on annotations from Liu and Xue (2023), along with an incremental improvement for English. With the recent integration of MDS into Uniform Meaning Representation (UMR) (Van Gysel et al., 2021), the prompt-based model has also been used as part of the UMR parsing pipeline (Chun and Xue, 2024). This implies that improved modal dependency parsing performance can have beneficial downstream impact for UMR parsing.

Given the status of the prompt-based model as the current state-of-the-art in MDP, we briefly summarize its core setup. Here an event and its sentence—known as *prompt*—is paired with some local *context* as defined by the number of sentences before, including, and after the prompt sentence. The parser is then trained to predict the event's parent and grand-parent conceivers from the context sentences, based on the simplifying assumption that an event has a chain of one or two conceivers 96% of the time (Yao et al., 2021)[4].

While this is a prudent approach that alleviates the multi-tasking complexity of the parsing process by first focusing on events whose definition is more widely accepted and less context-dependent than conceivers, it is susceptible to error propagation during decoding due to its pipelined setup. Furthermore, since each event requires an individual forward pass over the local context, both training and decoding can be slow, particularly when processing lengthy documents with numerous event candidates. Finally, the simplifying assumption that enables the approach also restricts the model from generating certain forms of modal structures, reducing its generalization capabilities. In contrast, our proposed biaffine parser has a simpler end-to-end setup that requires just a single forward pass over the entire document and does not rely on any simplifying assumptions.

This line of approach based on the deep biaffine scoring mainly traces its roots to dependency parsing (Dozat and Manning, 2017, 2018; Zhang et al., 2020) but has also been explored in other areas such as NER tagging (Yu et al., 2020) and constituency parsing (Bai et al., 2021; Chen and Komachi, 2023). To the best of our knowledge, however, no existing research has examined the utility of modeling the self-loop within the biaffine-based parsing framework.

3 Approach

Motivation The primary challenge in modal dependency parsing lies in its inherently multi-tasking nature that consists of 4 different sub-tasks. First step is to (1) identify spans of modal nodes, because not all tokens participate in the modal dependency structure (for instance, 'likely' in Figure 2). Once located, these spans must be (2) labeled as either an event or a conceiver. This is followed by (3) arc generation for each node and (4) label assignment for the newly created edges.

As a result, previous approaches have primarily relied on a pipeline framework (Yao et al., 2021, 2022). Although these efforts successfully established a strong baseline performance, they fall short of fully capturing the complexity of modal dependency parsing due to the simplifying assumptions

[3] https://github.com/jryao/modal_dependency

[4] This also ignores the small possibility of event-to-event modal relations, which occurs for 2.7% of child events from the training dataset in constructions such as "He '*decided*' to '*eat*'," where '*eat*' is a child of '*decided*.'

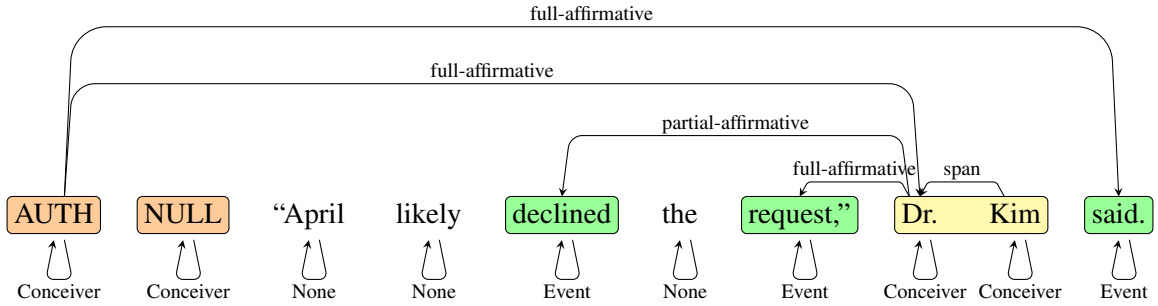


Figure 3: Example of Modal Dependency Parsing for the sentence from Figure 2b: “April likely declined the request,” Dr. Kim said. Orange nodes indicate abstract nodes for (1) the author of the document, (2) the null-conceiver which is reserved for when the conceiver is not specified.

that constrain their structural expressiveness, such as constraining any subtree to a depth of 2.

This study is an attempt to develop a streamlined yet comprehensive framework capable of fully representing the modal dependency relations without relying on any simplifying assumptions. Although a biaffine parser is seen as an effective solution for dependency parsing in various contexts, its ability to fully address the complexities of modal dependency parsing is only realized when support for self-loop is incorporated. This is because while the biaffine parser has been conventionally used to generating arcs *between* different tokens, the addition of a self-loop enables it to additionally behave as a *tagger*, which enables more effective handling of conceiver node identification while eliminating the need for a pipelined setup. As illustrated in Figure 2, conceiver identification is a highly context-dependent sub-task that significantly benefits from access to the overall structural information. Our approach of training the biaffine module to be both a parser *and* a tagger provides the flexibility to leverage partially constructed parse graphs in the decoding process that is not afforded when using a separate tagger.

In what follows, we provide details on model setup and architecture.

Setup Building on the fundamental assumption that a modal dependency structure is inherently a tree and therefore *single-headed* (Yao et al., 2021), our parser employs the biaffine attention to locate the most suitable head for each token and to label the newly formed edges. This is a natural way of modeling the modal arcs and relations between different nodes—sub-tasks (3) and (4)—which is similar to that of a traditional dependency parsing configuration. These arcs and relation labels are drawn on top of the tokens in Figure 3.

However, not all phrases act as events or conceivers within the modal dependency structure. This is why MDP can be understood as a form of *sparse* dependency parsing, and the non-modal tokens must be pruned first.

To this end, we introduce self-loop for every token. Since self-loops are structurally self-evident, it is only the predicted edge label that is of interest, chosen from three possible options: ‘`Event`’, ‘`Conceiver`’ or ‘`None`’. When a token is labeled ‘`None`’, it is neither an event or a conceiver and is hence subject to be be pruned. Otherwise, the self-loop label is used to classify between an event and a conceiver—sub-task (2)—as seen with the edges below each token in Figure 3.

Meanwhile, dependency structure is arguably not the most intuitive method of representing multi-word spans, because some arbitrary token must be raised as a head to ensure structural integrity. This is also an issue for MDP as an event or a concept may extend across multiple words but must be treated as a single node in the MDS.

We address this by assuming that the leftmost token is the representative head in our modeling where there are multiple words in a node[5]. Any token to the right within a multi-word span should hence be headed by the leftmost token with a special label of ‘`:span`,’ which triggers a special interpretation during decoding for a flat structure.

The key strength of this solution lies in its consistency with the single-headedness of MDS, while facilitating the use of the existing biaffine arc generator for span identification—sub-task (1)—without any modifications. For instance, ‘`Kim`’ points to ‘`Dr.`’ in Figure 3 with ‘`:span`’ edge label. Therefore, any relation to and from ‘Dr.’ should be

[5]Our experiments indicate that raising the right-most token instead does not produce significantly different results.

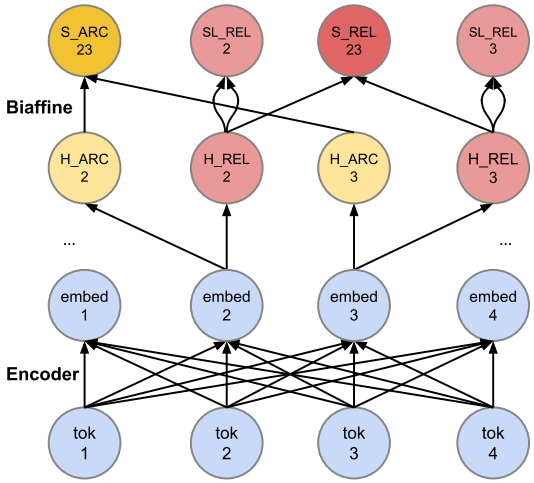


Figure 4: The network architecture diagram of our approach. 'H' stands for the hidden state, 'S' stands for the score. 'S_ARC' and 'S_REL' represent arc and relation prediction scores between different tokens. 'SL_REL' is the self-loop label score.

structurally interpreted as involving the span of 'Dr. Kim' as a whole.

Model Architecture Our proposed model network consists of two modules: (1) a document encoder based on a pre-trained language model (PLM), followed by (2) a biaffine module consisting of two biaffine layers that predict arc presence and edge labels respectively. Figure 4 visualizes the network architecture of our approach.

One of the major bottlenecks faced by the document encoder is the input length limitation imposed by the choice of the PLMs. This is especially relevant as the typical MDP text input is generally a multi-sentence document.

We cope with this challenge by splitting a long input sequence into smaller segments that are embedded independently, before being merged together (Yao et al., 2021). As such, we find that it is advantageous to choose a PLM with a long context window which can thereby reduce the frequency of such sentence fragmentation during encoding.

The contextualized embeddings from the document encoder are first projected into arc and relation hidden states by the biaffine parser, shown as 'H_ARC' and 'H_REL' in Figure 4 respectively. Then arc and relation scores 'S_ARC' and 'S_REL' are produced by the biaffine classifier that considers the arc and relation hidden states of any two positional tokens. While typical biaffine mechanism only considers two *different* positions, our model also considers self-loop. Since self-loops are self-evident edges, only the relation label needs to be predicted. These self-loop label scores are denoted as 'SL_REL' in the figure.

Formal Definition

Formally, a document $\boldsymbol{d}$ is represented as a sequence of tokens $(\boldsymbol{t_0}, ..., \boldsymbol{t_{-1}}, \text{AUTH}, \text{NULL})$, where the surface tokens are followed by two special tokens denoting the author and the Null Conceiver. A Null Conceiver is an abstract node introduced in cases where the conceiver is unspecified. Structurally, it is linked to the abstract root node but not to the author.

Let $\boldsymbol{H} = (\boldsymbol{h_0}, ..., \boldsymbol{h_{-1}}, \boldsymbol{h}_{\text{AUTH}}, \boldsymbol{h}_{\text{NULL}})$ be the contextualized embedding output from the document encoder for the document $\boldsymbol{d}$. Arc and relation scores for $\boldsymbol{i}$-th token and $\boldsymbol{j}$-th parent candidate token is obtained by two independent biaffine scorers:

$$\hat{\boldsymbol{y}}^{\boldsymbol{arc}}_{\boldsymbol{i,j}} = \text{Biaffine}_{\boldsymbol{1}}(\boldsymbol{h_i}, \boldsymbol{h_j})$$
$$\hat{\boldsymbol{y}}^{\boldsymbol{rel}}_{\boldsymbol{i,j}} = \text{Biaffine}_{\boldsymbol{2}}(\boldsymbol{h_i}, \boldsymbol{h_j})$$

During decoding, the final predictions are obtained by taking the argmax over the constrained search space, as discussed in greater detail in the following section:

$$\hat{\boldsymbol{j}}_{\boldsymbol{i}} = \arg\max_{\boldsymbol{j}} \hat{\boldsymbol{y}}^{\boldsymbol{arc}}_{\boldsymbol{i,j}}$$
$$\hat{\boldsymbol{r}}_{\boldsymbol{i}} = \arg\max_{\boldsymbol{r}} \hat{\boldsymbol{y}}^{\boldsymbol{rel}}_{\boldsymbol{i},\hat{\boldsymbol{j}}_{\boldsymbol{i}},\boldsymbol{r}}$$

The complete set of relation labels can be found in Table 1.

Our model attempts to minimize the negative log likelihood which is the sum of cross entropy losses:

$$\boldsymbol{\mathcal{L}} = \boldsymbol{\mathcal{L}_{arc}} + \boldsymbol{\mathcal{L}_{rel}}$$

Inference

While typical applications of dependency parsing utilize the first-order Eisner algorithm (Gormley et al., 2015) during decoding, it cannot be immediately applied in MDP for a couple of reasons. First, not all tokens participate in decoding and should be ignored. Second, the presence of a conceiver implies presence of some child node, and therefore only events can become the terminal nodes due to the structural constraint of MDS.

For this reason, we design a customized bottom-up decoding that begins by first pruning the non-modal tokens as predicted by the label of the self-loop. Although the self-loop label also allows

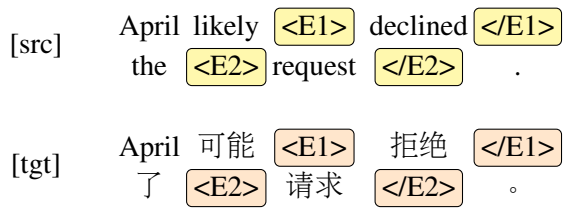


Figure 5: Sentence ("April likely declined the request.") augmented with special markers for events, translated and tagged from English (src) to Chinese (tgt).

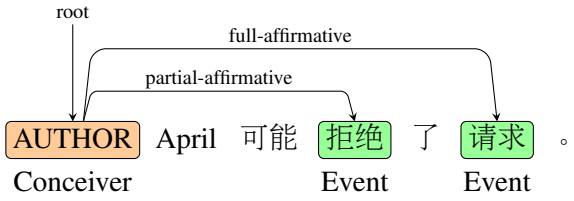


Figure 6: Silver annotation in Chinese translated from the English sentence: "April likely declined the request."

for classification of the remaining tokens as either event or conceiver, only the event nodes are generated at this phase, because conceivers cannot not serve as terminal nodes in MDS. It must be noted that whenever a new node is created, we immediately look to its right position(s) to detect any incoming :span edges. If such an edge is found, the two tokens are consequently merged into a single node.

Then we enter a loop where every node without a head predicts its modal head and the corresponding relation label, with the pruned non-modal token positions being ignored throughout. It is only at this point that a conceiver node is created, provided that some node attempts to generate an arc to its position. This is in alignment with the constraint that a conceiver node requires existence of some child node.

This loop may be executed multiple times, since the initial set of newly created nodes has to find their respective heads in the next iteration. The loop terminates when there are no remaining event and conceiver nodes without a head. Since there is a limited number of tokens in a document, this decoding loop is guaranteed to terminate. Due to the wide rather than vertical characteristics of a typical modal dependency tree, in practice the decoding loop generally terminates rather quickly.

In the event of unexpected errors or inconsistencies, the system defaults to attaching to the author node with '`:full-affirmative`' edge label to ensure connectedness. The pseudo-code is provided in Algorithm 1 of Appendix E.

4 Data Augmentation with LLM

Due to the challenging and costly nature of the MDS annotation process via crowd-sourcing (Yao et al., 2021), the number of annotated documents has remained the same since the initial release. In an effort to artificially increase the number of training samples, we obtain silver data by leveraging the LLMs to translate annotated examples across languages. Throughout this process, we strive to maintain the spans of annotated modal events and conceivers, because the edges are automatically retained as long as the source and target vertices exist in the translation.

This is fundamentally a two-step process.

1. **Translation**: an LLM translates the raw document text from one language to another.

2. **Tagging**: With the translation as reference, the LLM attempts to copy over a set of special markers from the source sentence that indicate spans of events and conceivers.

For instance, consider a sample sentence from Figure 2b: "April likely declined the request." First, the LLM translates the raw English sentence to Chinese ("April可能拒绝了请求。"). Then, in a separate conversation context, the LLM is given an augmented English sentence as shown in Figure 5 as 'src', where the two events '`declined`' and '`request`' are surrounded by the special markers 'E1' and 'E2' respectively[6]. The LLM is then instructed to insert the same set of special markers 'E1' and 'E2' in the Chinese translation.

Figure 5 shows a successful completion of our data augmentation pipeline, where 'tgt' contains not only the correct Chinese translation but also the special markers that denote the same events in English. Hence, any edges in the English annotation that use these events as endpoints will be preserved. For instance, the modal edge ('`author`' `:full-affirmative` '`request`') from Figure 2b can now be mapped to ('`author`' :full-affirmative '请求'), since both '`request`' and '请求' are

[6]'E' is a shorthand for an event and the integer is a unique id assigned to each annotated span. Although not present in the sample sentence, 'C' is another possibility for conceivers.

English	Train	Train+Silver	Dev	Test
#Documents	289	586	32	32
#Sentences	6,825	10,276	740	759
#Tokens	151,487	293,914	17,308	17,177
#Conceivers	2,344	4,056	298	296
#Events	19,541	33,025	2,307	2,168
`:full-affirmative`	18,425	31,101	2,205	2,077
`:full-negative`	800	1,172	99	89
`:partial-affirmative`	1,292	2,351	165	158
`:neutral-affirmative`	1,368	1,871	136	140
Chinese	Train	Train+Silver	Dev	Test
#Documents	237	590	30	30
#Sentences	3,187	10,996	398	366
#Tokens	79,809	284,224	10,352	10,053
#Conceivers	879	4,349	136	116
#Events	11,679	34,284	1,464	1,318
`:full-affirmative`	10,879	32,339	1,383	1,257
`:full-negative`	331 (298*)	1,242	50 (45*)	31
`:partial-affirmative`	919	2,435	103	101
`:partial-negative`	0 (26*)	26	0 (5*)	0
`:neutral-affirmative`	429	1,994	64	45
`:neutral-negative`	0 (7*)	7	0	0

Table 1: Summary statistics of English and Chinese modal dependency datasets. Conceivers does not include Author which occurs once per document. Labels does not include `Depends-on` which occurs once per document. *Numbers in parenthesis in Chinese statistics denote counts of fine-grained negative values in a 6-way fine-grained version of the corpus.

tagged by the same special marker 'E2' in Figure 5. Figure 6 visualizes the final annotation in Chinese.

In practice, the LLM is instructed to translate a document sentence-by-sentence in a series of back-and-forths that keeps track of its previous translations. We hypothesize that access to the entire document text, along with its prior translations, can enhance the translation quality by ensuring consistent mapping of named entities across sentences. Moreover, sentence-level translation allows the model to focus on only a limited number of tags at a time during the tagging phase, potentially increasing the likelihood of preserving special markers. See Figure 8 in Appendix D for an illustration.

5 Experiments

5.1 Corpora

The parser is trained and evaluated using the English (Yao et al., 2021) and Chinese (Liu and Xue, 2023) modal dependency corpora, whose statistics are shown in Table 1. We follow previous work on the train/eval/test splits for both languages.

Unlike the English dataset where all of the negative polarity labels are merged into a single '`:full-negative`' label (Yao et al., 2021), Chinese dataset additionally offers a fine-grained version with '`:partial-negative`' and '`:neutral-negative`' annotations, albeit only a few in number. It is not explicitly stated which version is used in the experiments of Yao et al. (2022). We report results using the fine-grained version.

5.2 Data Augmentation

During data augmentation, all train, development, and test examples from one language are combined into a single input. The second column of Table 1 shows the summary statistics for the training dataset that has been augmented with the silver data from another language. The Chinese edge labels of '`:partial-negative`' and '`:neutral-negative`' are mapped to '`:full-negative`' during translation to English for consistency.

5.3 Baselines

We evaluate the performance of our parser against the prompt-based model from Yao et al. (2022). Since the prompt-based parser does not report conceiver identification scores, we attempt to replicate their results with the default set of hyperparameters. We observe slightly lower scores except for the test micro F1 on Chinese. We label this row in Table 2 as 'Prompt-ours'.

Because the choice of the PLM for English is different between our setup and the prompt-based parser Yao et al. (2022), we perform another experiment with the prompt-based model where the

Models	Split	English			Chinese		
		Event	Conceiver	Parsing	Event	Conceiver	Parsing
Prompt-based	Dev	93.2	-	72.7	87.4	-	65.5
	Test	**91.9**	-	71.9	**88.6**	-	63.6
Prompt-ours	Dev	91.1	68.6	71.7	84.6	83.5	64.1
	Test	89.7	72.1	70.8	85.3	85.2	64.7
Prompt-Longformer	Dev	93.0	69.7	72.4	-	-	-
	Test	91.0	72.0	71.3	-	-	-
Biaffine	Dev	93.0	**73.1**	74.5	87.2	**89.1**	68.6
	Test	91.8	**74.7**	73.3	87.5	87.3	66.7
+Silver Data	Dev	**93.3**	72.9	**74.5**	87.5	88.8	**69.0**
	Test	91.5	74.2	**73.5**	**87.9**	**87.4**	**67.3**

Table 2: Experimental results showing Event and Conceiver identification and Parsing micro-F score. The highest values are highlighted in bold. Empty values indicate unreported results.

original `bert-large-cased` (Devlin et al., 2019) is replaced with the `longformer-base`, leading to slightly improved results over our replicated results. These numbers are reflected in Table 2 with 'Prompt-Longformer' model name.

5.4 Results

Table 2 shows overall parsing results on English and Chinese MDP in micro F-score as average across 3 different seeds. Our approach outperforms the prompt-based model by about 2% in English, while the gain is even more significant with Chinese at around 4%, with more noticeable gain from data augmentation. The implementation details as well as experimental settings are described in Appendix B and C.

6 Analysis

For English MDP, the conceiver identification still remains a major bottleneck as a highly context-dependent task that may span across multiple sentences. Nevertheless, the improvement in conceiver identification by our proposed approach is evident with at least 2% gain across the board. Since both the prompt-based model and ours share the generally same approach of first prioritizing event detection followed by conceiver identification, we attribute the increase to the biaffine classifier as well as the customized decoding process.

To test this hypothesis, we perform an experiment with a simple decoding that does not use the bottom-up decoding described in Section E. Rather, we accept the self-loop label predictions as-is and generate the nodes in a single step, which is more akin to the prompt-based setup. We observe a slight dip in Conceiver Identification by 0.7 and 0.3 in dev and test sets respectively, which suggests that our customized decoding is indeed beneficial to the overall parsing performance.

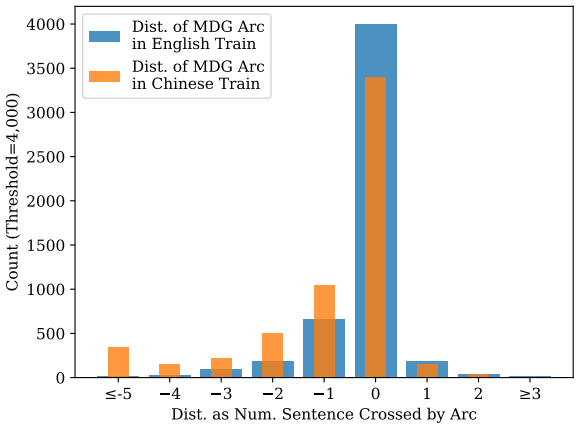


Figure 7: Number of sentences crossed by modal arcs in English and Chinese training corpora, with upper threshold at 4,000. Negative values mean the head is located in some previous sentence, while positive values imply otherwise.

The experimental results also show higher improvement in Chinese compared to English. Since the prompt-based model consistently shows lower Conceiver Identification scores for Chinese as well, the earlier discussion of the efficacy of customized decoding process is again relevant.

In fact, it is important to emphasize that our approach is fundamentally language-agnostic aside from the choice of pre-trained language models. This is in contrast to the prompt-based model, where a language-specific parameter for determining the size of the context window needs to be manually adjusted (Yao et al., 2021) based on the statistics shown in Figure 7, showing a bigger portion of modal heads in prior sentences for Chinese. Compared to English which is more clustered around the current sentence, it follows that the Chinese

modal dependency parser should be able to generate comparatively long-distance edges. Indeed, on the development set, the average number of sentences crossed by the biaffine classifier is 0.2 for English and 0.78 for Chinese. Although the prompt-based model comes similar for English at 0.18, it also favors generating edges that are significantly shorter for Chinese, at 0.62.

In terms of computational efficiency[7], the prompt-based model takes about 30 minutes per epoch during training and 5 minutes for inference on the development and test set. The biaffine parser in comparison is considerably faster, requiring only 10 minutes per training epoch and 2 minutes for inference. It is also worth noting that the prompt-based model further requires a separate training phase for the event tagger.

7 Conclusion

This work presents a biaffine modal dependency parser that is simple yet effective. By incorporating self-loop, our proposed approach is able to fully and efficiently address the multi-tasking nature of modal dependency parsing. We also show that training on silver data generated by using the LLMs to translate the annotated samples from one language to another leads to improved performance. The model is evaluated on the English and Chinese datasets and in both instances outperform the previous state-of-the-art.

Acknowledgment

This work is supported by grants from the CNS Division of National Science Foundation (Awards no: NSF_2213804) entitled "Building a Broad Infrastructure for Uniform Meaning Representations". Any opinions, findings, conclusions or recommendations expressed in this material do not necessarily reflect the views of NSF.

8 Limitations

MDP experiments remain focused on English and Chinese due to the limited availability of modal dependency annotations in other languages. However, with the adoption of modal dependency structure into Uniform Meaning Representation, more and more annotations for low-resource languages such as Arapaho, Cocama-Cocamilla, Navajo, Sanapaná and potentially additional languages may be prepared and released for future model fitting.

The English parsing results may not be reflective of true parsing performance on extremely long documents. It is by pure chance that none of the documents in the English dataset when tokenized is longer than the maximum context length supported by the longformer which is the PLM of choice for English. This implies that the default encoding method of splitting a long sequence into smaller fragments, each of which is embedded independently and then merged, needs not occur at any point during training and inference on the annotated datasets in our English experiments. It remains to be seen how the proposed approach is able to handle extremely long documents.

The domain of the English and Chinese datasets is limited to newswire only, where in general the sentences are grammatically correct and logically coherent everywhere. The model performance is yet to be tested in other domains.

[7]Using the default hyper-parameters specified in both studies, the models consume approximately same amount of VRAM (around 10GB) on a single RTX A6000.

References

Xinyi Bai, Nan Yin, Xiang Zhang, Xin Wang, and Zhigang Luo. 2021. Entity-aware biaffine attention fornbsp;constituent parsing. In *Artificial Neural Networks and Machine Learning – ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part I*, page 191–203, Berlin, Heidelberg. Springer-Verlag.

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

Zhousi Chen and Mamoru Komachi. 2023. Discontinuous combinatory constituency parsing. *Transactions of the Association for Computational Linguistics*, 11:267–283.

Jayeol Chun and Nianwen Xue. 2024. Uniform meaning representation parsing as a pipelined approach. In *Proceedings of TextGraphs-17: Graph-based Methods for Natural Language Processing*, pages 40–52, Bangkok, Thailand. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

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.

Mona Diab, Lori Levin, Teruko Mitamura, Owen Rambow, Vinodkumar Prabhakaran, and Weiwei Guo. 2009. Committed belief annotation and tagging. In *Proceedings of the Third Linguistic Annotation Workshop (LAW III)*, pages 68–73, Suntec, Singapore. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. *Preprint*, arXiv:1611.01734.

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

Matthew R. Gormley, Mark Dredze, and Jason Eisner. 2015. Approximation-aware dependency parsing by belief propagation. *CoRR*, abs/1508.02375.

Kenton Lee, Yoav Artzi, Yejin Choi, and Luke Zettlemoyer. 2015. Event detection and factuality assessment with non-expert supervision. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1643–1648, Lisbon, Portugal. Association for Computational Linguistics.

Zhifu Liu and Nianwen Xue. 2023. A dependency structure annotation for modality in chinese news articles. In *Chinese Lexical Semantics*, pages 143–157, Cham. Springer Nature Switzerland.

Amnon Lotan, Asher Stern, and Ido Dagan. 2013. TruthTeller: Annotating predicate truth. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 752–757, Atlanta, Georgia. Association for Computational Linguistics.

Rowan Nairn, Cleo Condoravdi, and Lauri Karttunen. 2006. Computing relative polarity for textual inference. In *Proceedings of the Fifth International Workshop on Inference in Computational Semantics (ICoS-5)*.

Amir Pouran Ben Veyseh, Thien Huu Nguyen, and Dejing Dou. 2019. Graph based neural networks for event factuality prediction using syntactic and semantic structures. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4393–4399, Florence, Italy. Association for Computational Linguistics.

Zhong Qian, Peifeng Li, Yue Zhang, Guodong Zhou, and Qiaoming Zhu. 2018. Event factuality identification via generative adversarial networks with auxiliary classification. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4293–4300. International Joint Conferences on Artificial Intelligence Organization.

Rachel Rudinger, Aaron Steven White, and Benjamin Van Durme. 2018. Neural models of factuality. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 731–744, New Orleans, Louisiana. Association for Computational Linguistics.

Roser Saurí and James Pustejovsky. 2012. Are you sure that this happened? assessing the factuality degree of events in text. *Computational Linguistics*, 38(2):261–299.

Roser Saurí and James Pustejovsky. 2009. Factbank: A corpus annotated with event factuality. *Language Resources and Evaluation*, 43:227–268.

Gabriel Stanovsky, Judith Eckle-Kohler, Yevgeniy Puzikov, Ido Dagan, and Iryna Gurevych. 2017. Integrating deep linguistic features in factuality prediction over unified datasets. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 352–357, Vancouver, Canada. Association for Computational Linguistics.

Jens E. L. Van Gysel, Meagan Vigus, Jayeol Chun, Kenneth Lai, Sarah Moeller, Jiarui Yao, Tim O'Gorman, Andrew Cowell, William Croft, Chu-Ren Huang, Jan Hajič, James H. Martin, Stephan Oepen, Martha Palmer, James Pustejovsky, Rosa Vallejos, and Nianwen Xue. 2021. Designing a uniform meaning representation for natural language processing. 35(3):343–360.

Meagan Vigus, Jens E. L. Van Gysel, and William Croft. 2019. A dependency structure annotation for modality. In *Proceedings of the First International Workshop on Designing Meaning Representations*, pages 182–198, Florence, Italy. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Jiarui Yao, Haoling Qiu, Jin Zhao, Bonan Min, and Nianwen Xue. 2021. Factuality assessment as modal dependency parsing. 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 1540–1550, Online. Association for Computational Linguistics.

Jiarui Yao, Nianwen Xue, and Bonan Min. 2022. Modal dependency parsing via language model priming. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2913–2919, Seattle, United States. Association for Computational Linguistics.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6470–6476, Online. Association for Computational Linguistics.

Yu Zhang, Zhenghua Li, and Min Zhang. 2020. Efficient second-order TreeCRF for neural dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3295–3305, Online. Association for Computational Linguistics.

A Corpus Details

The publicly available English dataset (Yao et al., 2021) contains newswire annotations from various news media sources (Yao et al., 2022). The Chinese dataset consists of newswire data from Xinhua news agency (Liu and Xue, 2023).

B Implementation Details

Hyperparameter	English	Chinese
PLM	`longformer-base`	`xlm-roberta-base`
PLM Dropout	0.1	0.1
Max. Seq. Len.	4096	512
Batch Size	4	1
Grad. Acc. Steps	4	4
Epochs	1,000	1,000
Optim.	AdamW	AdamW
LR	5e-5	5e-5
Weigh Decay	0.01	0.01
Warmup Prop.	0.1	0.1
Arc Hidden Dim.	512	400
Arc Dropout	0.33	0.33
Rel. Hidden Dim.	128	100
Rel. Dropout	0.33	0.33

Table 3: Hyperparameters used in experiments.

We use the Huggingface[8] (Wolf et al., 2020) implementation `longformer-base` (Beltagy et al., 2020) and `xlm-roberta-base` (Conneau et al., 2020) for English and Chinese, respectively. The choice of `xlm-roberta-base` is inherited from the prompt-based approach (Yao et al., 2022). The biaffine layer implementation is based on SuPar[9].

C Experimental Details

All experiments are run on a single NVIDIA RTX A6000 GPU and each run requires approximately 3 to 4 hours with the default hyperparameters in Table 3. The number of parameters for English is 149,450,504; that of Chinese is 278,770,441.

D LLM Prompting for Data Augmentation

Figure 8 illustrates the sample prompting method for obtaining silver data. The LLM model being used is `gpt-4o-2024-08-06`.

E Custom Bottom-up Decoding Pseudo-code

[8] https://huggingface.co/docs/transformers

[9] https://github.com/yzhangcs/parser

````
Phase 1: Document Translation

$ System Prompt:
    You are an expert translator. Your mission is to accurately translate the following document
        ↪ sentence-by-sentence from English to Chinese:
    ```
    May accused the couple of shoplifting.
    According to her, they had not paid for their ring.
    ```

$ User Prompt 1:
    Translate sentence 1 to Chinese (write only in Chinese):
    ```
    May accused the couple of shoplifting.
    ```

$ Assistant Response 1:
    ```
    May 指控这对夫妇偷窃。
    ```

$ User Prompt 2:
    Translate sentence 2 to Chinese (write only in Chinese):
    ```
    According to her, they had not paid for their ring.
    ```

$ Assistant Response 2:
    ```
    根据她的说法，他们没有支付戒指的费用。
    ```


Phase 2: Sentence Tagging

$ User Prompt:
    You are given a sentence in English and its translation in Chinese.

    Sentence in English:
    ```
    May accused the couple of shoplifting.
    ```

    Translation in Chinese:
    ```
    May 指控这对夫妇偷窃。
    ```

    You are now given the above sentence in English that has been augmented with special symbols of
        ↪ the format: <E%d>, </E%d>

    <E%d> and </E%d> indicate a span of an Event.

    Insert the same set of special symbols (<E1>, </E1>, <E2>, </E2>) in the Chinese translation that
        ↪  correspond to the same Event(s):
    ```
    May <E1> accused </E1> the couple of <E2> shoplifting </E2> .
    ```

$ Assistant Response:
    ```
    梅<E1> 指控</E1> 这对夫妇<E2> 偷窃</E2> 。
    ```
````

Figure 8: Sample prompts used to generate silver annotated data in Chinese from English gold annotations. Each 'Phase' represents a new conversation context with the LLM. In Phase 1, the document is translated sentence-by-sentence. In Phase 2, each translation gets independently tagged. We avoid redundancy by only showing the tagging process for the first sentence from the sample document.

Algorithm 1 Custom Bottom-up Decoding

```
Input: input_doc
Output: modal_nodes_with_heads
logits ← compute_model_logits(input_doc)
pruned_nodes ← prune_non_modal_tokens(logits) // Each token treated as a node
label_event_nodes(pruned_nodes)
while True do
    headless_nodes ← find_nodes_without_head(pruned_nodes)
    if headless_nodes.is_empty() then
        break
    end if
    for all node ∈ headless_nodes do
        predicted_head ← predict_arc(node, pruned_nodes)
        if NOT predicted_head.is_labeled() then
            label_node_as_conceiver(predicted_head)
        end if
        assign_head(node, predicted_head)
    end for
end while
return pruned_nodes_with_heads
```