MWE as WSD: Solving Multiword Expression Identification with Word Sense Disambiguation

Anonymous ACL submission

Abstract

Recent approaches to word sense disambiguation (WSD) utilize encodings of the sense gloss (definition text), in addition to the input words and context, to improve performance. In this work we demonstrate that this approach can be adapted for use in multiword expression (MWE) identification by training a Bi-encoder model which uses gloss and context information to filter MWE candidates produced from a simple rule-based extraction pipeline. Our approach substantially improves precision, outperforming the state-of-the-art in MWE identification on the DiMSUM dataset by 0.9 F1 points and achieving competitive results on the PARSEME 1.1 English dataset. Our model also retains most of its ability to perform WSD, demonstrating that a single model can successfully be applied to both of these tasks. Additionally, we experiment with applying Poly-encoder models to MWE identification and WSD, introducing a modified Poly-encoder architecture which outperforms the standard Poly-encoder on these tasks and improves MWE identification performance.

1 Introduction

Word sense disambiguation (WSD), the task of predicting the appropriate sense for a word in context, and multiword expression (MWE) identification, the task of identifying multiword expressions in a body of text, are both tasks that deal with determining the meaning of words in context (Maru et al., 2022; Constant et al., 2017). Despite their commonalities, and the fact that both share a place in the NLP pipeline as preprocessing tasks, they have traditionally been treated as separate tasks. This is potentially disadvantageous as WSD performed on words which are part of unrecognized MWEs cannot produce correct meanings, and the meanings of polysemous MWEs are ambiguous even after identification.

In order to correctly identify the meanings of all words in a sentence we must solve both of these tasks in an integrated way. WSD can give us the appropriate sense for both single words and MWEs, but we must first identify which words in the sentence are part of MWEs. Consider the sentence “She inherited a fortune after her grandfather kicked the bucket.”, which tell us that someone’s grandfather has died, but we would not expect to find meanings associated with death in the sense inventories of either *kick* or *bucket*. However, like many other MWEs, *kick the bucket* can also have a literal, non-compositional meaning as in “He kicked the bucket down the hill”, so we cannot indiscriminately mark all combinations of words in known MWEs as those MWEs. Finally, note that MWEs can have multiple possible senses in the same way words can: for example, *break up* can refer both to objects physically breaking apart and romantic relationships ending.

In this paper, we propose a system that can tackle word sense disambiguation and multiword expression identification together, using an MWE lexicon and rule-based pipeline to identify MWE candidates and a Bi-encoder to both perform WSD and filter MWE candidates. Similar to prior work in WSD, Our Bi-encoder consists of two BERT (Devlin et al., 2019) models which encode the words in context along and the possible sense glosses, respectively, into the same embedding space. By utilizing gloss information to filter out MWE candidates whose meanings don’t make sense in context, we improve precision and achieve state-of-the-art F1 for MWE identification on the DiMSUM dataset (Schneider et al., 2016) and competitive performance on the PARSEME 1.1 English data (Ramisch et al., 2018). To the best of our knowledge, this work is the first to use glosses as a resource for multiword expression identification.

Additionally, we experiment with Poly-encoders (Humeau et al., 2020) for the same set of tasks, proposing a novel architecture that helps the Poly-encoder focus on specific words. Our contributions are summarized as follows:

- We show that it is viable to solve MWE identification and WSD together, presenting an approach which uses a rule-based system to generate MWE candidates and a Bi-Encoder to filter them and perform WSD
- We demonstrate that our approach produces models capable of both tasks, achieving state-of-the-art results for MWE identification on DiMSUM and only 6% less F1 for WSD than an equivalent single-task model
- We propose a Poly-encoder architecture which outperforms the standard Poly-encoder on our tasks and improves MWE identification performance on PARSEME

We make all of our code, models and data public.

2 Related Work

2.1 Word Sense Disambiguation

The task of word sense disambiguation has a long history in NLP, first introduced as a necessary step for machine translation by Weaver (1949). In fact, WSD has shown to be useful for improving downstream performance not just in machine translation, but in other tasks such as Information Extraction as well (Barba et al., 2021; Song et al., 2021).

Until the last few years, most approaches to WSD treated senses simply as labels from a large vocabulary of possible labels in a classification task. This formulation risks limiting the information available to the model about each sense to only what is learnable from the training data, and can lead to poor performance on rare or unseen senses. In order to mitigate these problems, recent approaches to WSD have improved performance by incorporating glosses (Blevins and Zettlemoyer, 2020; Barba et al., 2021; Zhang et al., 2022).

Our work is inspired by this methodology and utilizes gloss information to improve MWE identification. In particular, Blevins and Zettlemoyer (2020) demonstrate that a simple Bi-encoder model consisting of two BERT models can achieve competitive WSD performance, with Kohli (2021) improving Bi-encoder training for WSD and Song et al. (2021) achieving further performance gains through improved sense representations. Bi-encoder models also have the advantage of being efficient at inference time due to the fact that document representations (for WSD, gloss representations) can be computed in advance and cached, which lead us to choose this architecture for our experiments.

2.2 Poly-encoders

The Poly-encoder architecture was proposed by Humeau et al. (2020) as a middle ground between Bi-encoders and Cross-encoders (which jointly encode all possible input pairs), retaining the speed advantage of the Bi-encoder, but allowing some information to flow between the two encoder outputs like the Cross-encoder. It can be used in place of Bi-encoder models in tasks such as information retrieval (Li et al., 2022) and text reranking (Kim et al., 2022), or in our case, word sense disambiguation and MWE identification.

2.3 Multiword Expression Identification

Precisely defining what constitutes a multiword expression has proven to be difficult (Maziarz et al., 2015), but they can be broadly defined as groupings of words whose meaning is not entirely composed of the meanings of included words (Sag et al., 2002; Baldwin and Kim, 2010). This includes idioms such as *a taste of one's own medicine*, verb-particle constructions such as *break up* or *run down*, idiomatic compound nouns such as *bus stop*, and potentially any other grouping of words with non-compositional semantics. In fact, a significant portion of noun MWEs are named entities, such that there is some overlap between MWE identification and NER (Savary et al., 2019).

The task of MWE identification is locating these MWEs in a given body of text. The two main approaches to solving MWE identification have been rule-based systems (Foufi et al., 2017; Pasquer et al., 2020) and neural token tagging systems (Rohanian et al., 2019; Liu et al., 2021). Rule-based systems remain competitive with neural models in this task, and many systems use MWE lexicons in order to identify multiword expressions in text, which Savary et al. (2019) argue are critical to making progress in MWE identification. This applies to our system as well, which relies on a lexicon in order to be able to find candidate MWEs. Kurfalı and Östling (2020) and Kanclerz and Piasecki (2022) are similar to our work in that they frame the task of MWE identification as a classification problem, although neither use gloss information.

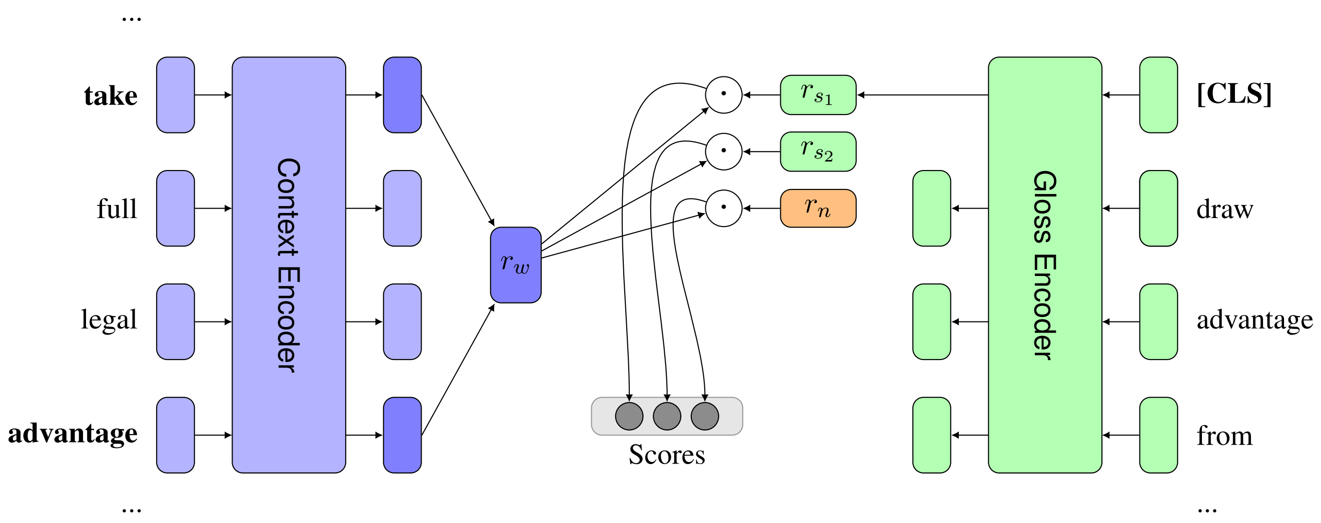

Figure 1: A diagram illustrating how the Bi-encoder computes scores for an arbitrary MWE. Representations of each MWE are computed as an average of its constituents, and sense representations are taken from the [CLS] token encodings. The dot product of these representations then becomes the scores for each sense.

Among all the types of MWEs, verbal MWEs are particularly difficult to identify due to their surface variability — constituents can be conjugated or separated so that they become discontinuous (Pasquer et al., 2020). Much work on verbal MWE identification, especially in languages other than English, has been done as part of recent iterations of the PARSEME shared task (Ramisch et al., 2018) which focused on identifying verbal MWEs across a wide variety of languages.

3 Methodology

In this section, we explain our approach for solving multiword expression identification and word sense disambiguation.

3.1 Bi-encoder

We use a Bi-encoder identical to that of Blevins and Zettlemoyer (2020) for WSD, consisting of a **context encoder** T_c and **gloss encoder** T_g, both of which are BERT (Devlin et al., 2019) models. Given an input context sentence $c = (w_0, ...w_n)$ containing the target words to disambiguate, we first tokenize it and use the context encoder to produce representations for each token. Because tokenization may break words up into multiple subwords — and because as described below, we also use this model for multiword expressions — representations are computed as an average of all subwords in a word or MWE.

$$T_c(c) = t_0, ...t_n$$

$$r_w = \frac{1}{|w|} \sum_{t \in w} t$$

Then, for each target word, the **gloss encoder** produces representations for each of the word's senses. We pool the encoder output by taking the representation of the [CLS] token for each sense.

$$r_s = T_g(g_s)[0]$$

Scores corresponding to possible senses for each target word are computed as the dot product of the word and sense representations.

$$\phi(w, s_i) = r_w \cdot r_{s_i}$$

Finally, the model predicts the sense with the highest score.

$$pred(w) = \arg\max_{s_i} \phi(w, s_i) : s_i \in S_w$$

An overview of the model architecture can be seen in Figure 1.

3.2 Poly-encoder

We also experiment with Poly-encoders as an alternative to the Bi-encoder model. The Poly-encoder still has two encoders, a **context encoder** T_c for target word contexts and a **gloss encoder** T_g for gloss definitions. There is also a new set of parameters that Humeau et al. (2020) refer to as **code**

embeddings, Q. These codes are used to extract information from context representation produced by the **context encoder**. The inputs to the Poly-encoder are the same as the Bi-encoder, sense representations r_s are computed in the same way, and predictions are still the highest scoring sense. However, senses are scored differently, as described below.

We take the last hidden state of the **context encoder** as the as the context representation $r_c = T_c(c)$, which we use along with the code embeddings $Q = (q_1, ..., q_m)$ in the first dot-product attention step (*code context attention*) of the Poly-encoder. We use a different set of embeddings for single words and MWEs. The number of embeddings, m, is a hyperparameter and their dimensionality is the same as the encoders' hidden sizes. The context representation r_c is used as both keys and values in this dot-product attention module, yielding a *code attended context* Y_{ctxt}. The representation a code q_i extracts is as follows:

$$(w_0^{q_i}, ..., w_n^{q_i}) = softmax(q_i \cdot r_{c_1}, ...q_i \cdot r_{c_n})$$

$$y_{ctxt}^i = \sum_{j=1}^{n} w_j^{q_i} r_{c_j}$$

The sense representations r_s are then used as queries and the code-attended context representations Y_{ctxt} are used as keys and values in a final dot-product attention module, which yields a *gloss attended code-context*. For a word or MWE w with $|S_w| = k$ possible senses:

$$(w_1, ..., w_m) = softmax(r_{s_i} \cdot y_{ctxt}^1, ..., r_{s_i} \cdot y_{ctxt}^m)$$

$$y_{final} = \sum_{i=1}^{k} w_i y_{ctxt}^i$$

We then take the dot product of the gloss attended code-context y_{final} and each gloss embedding $r_{s_0}, ...r_{s_k}$, yielding a score for each definition: $\phi(w, s_i) = y_{final} \cdot r_{s_i}$.

3.3 Distinct Codes Attention

Since the Poly-encoder was originally designed to compute *sentence* representations, it contains no mechanism for explicitly focusing on a specific set of target words/subword tokens. To address this problem, we propose a variation of the Poly-encoder which we call "distinct codes attention" (DCA). We change the *code context attention* step of the Poly-encoder to use two sets of code embeddings: one set of code embeddings for target words and one set for non-target words. Since we also maintain different code embeddings for single words and MWEs, this gives us a total of four sets of code embeddings.

In the first attention module, *code-context attention*, we now construct a query matrix using the target-word codes only at the indices of subwords in a target word or MWE, and the nontarget code embeddings elsewhere. We do this by using two masks: the target mask, M_t, which is 1 at the indices of target subwords and 0 otherwise. The nontarget mask M_{nt} is the opposite: 0 at target indices, 1 elsewhere. We then multiply each mask by its respective code embeddings Q and then add the products:

$$QK^T = (M_t * Q_t) + (M_n * Q_{nt})$$

Finally, we softmax and multiply QK^T by the encoded context r_c to yield the *code attended context*, $Y_{ctxt} = softmax(QK^T)(r_c)$. The *gloss attended context* and final scores are then computed identically to the standard Poly-encoder.

3.4 MWE Identification Pipeline

Our system for MWE identification is a three-stage pipeline inspired by Kulkarni and Finlayson (2011), consisting of one or more **detector** functions which generate possible MWEs from an input sentence, zero or more **filter** functions which filter these candidates, and up to one **resolver** which chooses between two MWE candidates in case of overlap.

Our **detector** is a simple exhaustive search which returns all combinations of words in a sentence which correspond to MWEs in our lexicon. That is, our initial set of candidates before filtering is every combination of words in the input sentence that correspond to any MWEs in our lexicon. Our **filter**s include *OrderedOnly* which discards MWE candidates where the constituent words are out of order and *MaxGappiness* which discards candidates with too many tokens in between constituents, but the most critical is the *BiEncoderFilter*, which discards MWE candidates judged to be incorrect by our Bi-encoder (or Poly-encoder) model. We use the term "rule-based pipeline" to describe variations of the pipeline without the *BiEncoderFilter* in later sections.

3.4.1 Bi-encoder Filter

Because all of our MWE candidates correpond to words (and consequently subwords) in the input sentence, we can produce a representation r_w for each MWE candidate, along with scores for each of their possible senses, the same way we do for words. However, since no MWE will have a sense corresponding to the case where that candidate is a false positive, we define a special sense n to represent the case where none of the other senses are correct (I.E. where this candidate is not actually an MWE, or at least not one in our lexicon). Since n has no gloss, we cannot use the **gloss encoder** to compute a representation for it, and instead make this representation a learnable parameter matrix r_n, with the same dimensionality as the model's hidden size. This representation can then be used in the scoring functions for the Bi-encoder or Poly-encoder to compute a score for the candidate not being an MWE, which we use in our *BiEncoderFilter*. This filter excludes any MWE candidates where the "not an MWE" score is higher than any of the scores for other senses, retaining only candidates for which the below is true:

$$\exists s_i \in S_w \; \phi(w, s_i) > \phi(w, n)$$

Although uncommon after filtering, in cases of overlap between candidates, our **resolver** chooses the MWE with the largest difference between its highest scoring sense and the "not an MWE" sense. Note that since this filtering process involves computing scores for all possible senses, it also effectively performs WSD on any polysemous MWEs.

3.4.2 Limitations

The output of our MWE pipeline can only ever be a subset of the original candidates generated, which are by definition a subset of MWEs present in our lexicon. Furthermore, because our *BiEncoderFilter* uses the gloss text as input, it requires that definitions be present for all MWE lexicon entries. Consequently, our approach depends on the presence of a high-quality lexicon which includes both MWE lemmas and possible definitions, making it ill-suited for scenarios where data like this may not be publicly available yet, such as in low resource languages. However, we are optimistic that work in MWE discovery (Ramisch et al., 2010) and definition generation (Bevilacqua et al., 2020) will help to mitigate this problem by automating parts of the data creation process.

4 Experiments

4.1 Lexicon

We use WordNet (Miller, 1995) as our MWE lexicon for all experiments, treating every entry including the character "_" as an MWE. All sense glosses are taken from WordNet 3.0.

4.2 Training Data

We train our models on SemCor (Miller et al., 1993), a word sense disambiguation dataset containing a total of 226,036 examples annotated with senses from WordNet. In order to make the data usable for MWE identification in addition to WSD, we preprocess it in the following ways. First, since MWEs in SemCor are not distinguished from normal words, we explicitly mark any words whose lemma includes the character "_" as MWEs such that during training the possible labels for these MWEs include the "not an MWE" sense as well as their normally available senses. We also attach stranded constituents to their parent MWE, since some discontiguous MWEs in SemCor are labeled only on a subset of the included words[1]. Finally, because SemCor contains no examples of negative MWEs — instances where the constituent elements of an MWE are all present but their meaning in context does not match any of the MWE senses — we must add these ourselves. We primarily do this by automatically generating synthetic negative examples, using the rule-based pipeline with its filters inverted. That is, we mark out-of-order and/or extremely gappy MWEs as training examples whose gold label is the "not an MWE" sense. We randomly add negative training examples in this fashion until they account for approximately 50% of the MWE examples in the training data.

While this approach is effective in generating a large number of negative examples, it risks encouraging the model to learn the heuristics used to generate these synthetic negative examples instead of learning how to judge whether a MWE candidate is correct using context and the information in its gloss(es). In order to combat this, we manually annotate a small number of examples which are neither out of order nor excessively gappy. Annotation is done by using a variation of the rule-based pipeline to extract candidates, which we mark either with the appropriate sense from WordNet or as a negative example if none of the available senses are appropriate for the constituent words in the given sentence.

[1] For example, in the sentence "Are they encouraged to take full legal advantage of these benefits?" (ID d000.s015), the verb *take* is correctly labeled as the MWE *take_advantage*, but *advantage* is not labeled as being part of any MWE.

	Pos MWE	Neg MWE
SemCor	12409	0
+Annotation	12907	658
+AutoNeg	12907	14688

Table 1: SemCor data after each processing step

4.3 Training

Like Blevins and Zettlemoyer (2020), we train with cross-entropy loss. The difference is that for MWEs, there is one additional possible label representing the "not an MWE" case. Given a word or MWE w, its gold sense g_s, and $|S_w| = j$ possible senses in the lexicon, this formalizes to:

$$\mathcal{L}(w, g_s) = -\phi(w, g_s) + \log \sum_{x \in X} \exp(\phi(w, x))$$

$$X = \begin{cases} \{s_0, ...s_j, n\} & \text{if MWE} \\ \{s_0, ...s_j\} & \text{otherwise} \end{cases}$$

We define batch size by the number of training examples (words or MWEs to be labeled) in each batch, and keep this number constant by adjusting the number of sentences and/or masking out examples to save them for the next batch. We train for 15 epochs, computing F1 on the WSD and MWE identification dev sets once per epoch and using the best performing model as our final model. Batch size and other hyperparameters such as learning rate were determined by hyperparameter search. Further implementation and training details can be found in Appendix A.

4.4 Evaluation

We evaluate our model on two MWE detection datasets: The English section of the PARSEME 1.1 Shared Task (Ramisch et al., 2018) and the DiMSUM dataset (Schneider et al., 2016). We do not evaluate on STREUSLE (Schneider et al., 2018) as it requires predicting lexical categories and supersenses[2], while our system predicts only the presence or absence of MWEs. For performance on the WSD task, we use the unified evaluation framework established by Raganato et al. (2017), and evaluate on the English all-words task.

We report scores on four variations of our system. The first is an entirely rule-based pipeline with no *BiEncoderFilter*, and the remainder are the same pipeline with variations of the *BiEncoderFilter* with different models: one Bi-Encoder (abbreviated as BiEnc) trained on the modified SemCor data, one trained on the SemCor data and then fine-tuned on the MWE identification datasets, and our DCA Poly-encoder trained and fine-tuned on the same data. We fine tune using any positive labeled examples of MWEs which are in our lexicon (as we cannot identify MWEs missing from our lexicon regardless), and take any incorrect outputs of our pipeline on the fine tuning data as negative examples. This means that all the negative training examples the model sees when fine tuning are false positives from the model itself, allowing the model to learn from its mistakes. Because PARSEME and DiMSUM are not annotated with sense information (only a binary labeling of MWE or not), we pick the first sense from WordNet as the gold label for positive examples when fine-tuning.

4.4.1 PARSEME 1.1

The PARSEME 1.1 shared task focuses on the identification of verbal multiword expressions, with its English data containing 3471 sentences in the training set and 3965 in test. We use 10% of the train data for our dev set when fine tuning. Because the dataset contains only verbal MWEs, when evaluating on PARSEME we add a filter that limits the output of our pipeline to verbal MWEs.

4.4.2 DiMSUM

The DiMSUM test set consists of a mixture of online reviews, tweets and TED Talks which have been annotated with MWEs and other information. There are 4799 sentences in the training set, and 1000 in the test set. As with the PARSEME data, we use 10% of the train data for our dev set. Because most noun phrases are marked as MWEs in DiMSUM, when evaluating on DiMSUM we also add a rule-based detector which marks consecutive nouns as MWEs.

4.4.3 WSD Evaluation

Following standard practice, we use the SemEval-2007 dataset (Pradhan et al., 2007) as our dev set, holding out the remaining Senseval-02, Senseval-03, SemEval-2013, and SemEval-2015, as test sets (Palmer et al., 2001; Snyder and Palmer, 2004; Navigli et al., 2013; Moro and Navigli, 2015).

5 Results

Table 2 shows results on MWE identification for PARSEME 1.1 English and DiMSUM, as well as

[2] The STREUSLE evaluation script rejects input without appropriate lexical categories/supersenses

Evaluation Results	PARSEME 1.1						DiMSUM			WSD
System	**MWE-Based**			**Token-based**			**MWEs**			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**F1**
Taslimipoor+ (2019)	-	-	36.0	-	-	40.2	-	-	-	-
Rohanian+ (2019)	-	-	**41.9**	-	-	-	-	-	-	-
Kirilin+ (2016)	-	-	-	-	-	-	73.3	48.4	58.4	-
Williams (2017)	-	-	-	-	-	-	65.4	**56.0**	60.4	-
Liu+ (2021)	36.1	**45.5**	40.3	40.2	**52.0**	**45.4**	47.9	52.2	50.0	-
BEM (2020)	-	-	-	-	-	-	-	-	-	**79.0**
Rule-based	16.3	39.9	23.1	19.2	43.9	26.7	57.7	55.5	56.6	-
BiEnc (SemCor)	27.5	38.8	32.2±0.8	30.0	39.43	34.1±0.3	70.7	52.57	60.0±0.4	77.4±0.6
BiEnc (fine-tuned)	44.5	31.0	36.5±0.9	46.4	30.5	36.2±0.8	**80.9**	49.3	**61.3**±0.4	74.2±1.0
DCA (fine-tuned)	**45.4**	33.2	38.3±0.1	**46.9**	31.9	38.0±0.2	80.4	49.5	**61.3**±0.4	74.4±0.6

Table 2: Test set results on PARSEME 1.1 English and DiMSUM for MWE identification, and the English all-words WSD task. For trainable models we report the mean (± standard deviation for the F1 score) of three runs with random seeds. Because our system uses gold POS tags/lemmas to look up sense glosses, we compare against systems using gold information where available, such as for Liu et al. (2021) and Kirilin et al. (2016).

the English all-words WSD task. We focus primarily on work in MWE identification, but include BEM (Blevins and Zettlemoyer, 2020) as a point of reference for WSD performance, since it is a Bi-encoder trained exclusively for WSD.

Our system achieves moderate performance on PARSEME and competitive performance on the DiMSUM trained only on the modified SemCor data. When fine-tuned on the training data from both MWE identification datasets its performance on PARSEME improves and it achieves state-of-the-art performance on DiMSUM. Additionally, its F1 for the WSD task is only 6% lower than BEM, showing that a single model can perform both tasks.

High precision stands out as a clear strength of our approach to MWE identification, but it suffers from low recall — even the entirely rule-based pipeline with minimal filtering still falls behind other systems in recall. We attribute this primarily to the issue of lexicon dependence described in Section 3.4.2; multiword expressions missing from our lexicon simply cannot be identified, and this accounts for a large portion of our false negatives as we show in our error analysis below (Section 6). These findings echo those of Savary et al. (2019) in terms of the importance of lexicons for MWE identification, and suggest that there is room to improve performance simply by expanding the lexicon.

5.1 Poly-encoder Performance

The standard Poly-encoder architecture performed worse than the Bi-encoder systems for all tasks, likely because it was designed to improve sentence representations and has no mechanism to focus on target words. Our proposed distinct codes attention architecture remedies this weakness and outperforms the Bi-encoder on the PARSEME data while matching its performance for other datasets, such that overall our best-performing model overall is the fine-tuned DCA Poly-encoder. However, we did not find that the DCA architecture meaningfully improved performance on WSD in our experiments, and leave Poly-encoder architectures for WSD to future work.

5.2 Transfer Learning Ablation

In order to assess whether the model's performance on MWE identification benefits from training on SemCor data, we also train models using just the PARSEME and/or DiMSUM data. We find that the presence or absence of this SemCor pretraining makes a substantial difference; systems using models trained on only a single dataset barely outperform the rule-based pipeline. A model trained on both datasets produces slightly better results (likely just due to having more training data), but still scores worse than even the SemCor only model, achieving 30.0 and 59.0 F1 on PARSEME and DiMSUM respectively.

6 Error Analysis

In order to better understand the output of our system and its performance on the DiMSUM and PARSEME data, we perform an error analysis of the output of our base and fine-tuned models on both test sets, taking 50 false positives and 50 false

Dataset	Type	Sentence	Note
PARSEME	FP	*...were **propped up** on a foot-warmer, ...*	**prop up** never marked as MWE in dataset
PARSEME	FN	***Never mind**, Mrs. Bray will join you later.*	**never mind** missing from lexicon
PARSEME	FP	*...his mind **drifted off** to the accounts...*	Lexicon definition of “fall asleep” does not apply
DiMSUM	FP	*Aww, **thank you**.*	**thank you** marked as MWE in 4 other sentences
DiMSUM	FN	*All our dreams can **come true**,...*	**come true** missing from lexicon
DiMSUM	FN	*...this was a **breathe of fresh air**.*	Present in lexicon; Bi-encoder false negative

Table 3: Error Examples

negatives from each combination of model and dataset (for a total of 400 examples). Select examples can be seen in Table 3, and detailed statistics about the outcome of our analysis can be found in Appendix B.

For false positives, we find that more than 80%[3] of the time a definition found in our lexicon was appropriate for the combination of words marked as an MWE in context of that sentence, meaning that these are cases where the model is successfully identifying a MWE candidate with a valid definition in our lexicon but the output still disagrees with the gold label annotations. Many of these MWEs are present in our lexicon but nowhere in the test set, suggesting discrepancies between the scope of what WordNet and these datasets respectively define as multiword expressions. Furthermore, there are also a significant number of false positives for MWEs that *are* marked as MWEs in other places in the dataset, but not in that specific sentence. In some cases this may be because these combinations of words were only marked as MWEs when they had specific meanings or particularly non-compositional semantics, but this did not seem to be the case for many examples we examined.

For false negatives, more than 85% were cases where the target MWE was missing from our lexicon, so the bottleneck for recall appears to be our system's lexicon. However, for the majority of false negatives where the the MWE was present in our lexicon it was also associated with a definition appropriate for that combination of words in that sentence, meaning that these represent failures of our MWE identification system and not the lexicon. In conclusion, the results of our analysis speak to the difficult and potentially subjective nature of defining and annotating MWEs, and we hope to see further work exploring this area in the future.

7 Conclusion

In this work, we present an approach to MWE identification and WSD using rule-based candidate extraction with a Bi-encoder filter, achieving strong results on the PARSEME 1.1 English data and state-of-the-art results for MWE identification on the DiMSUM dataset. Our system uses the same model for both word sense disambiguation and MWE identification, demonstrating that these tasks can be tackled together. We also experiment with applying Poly-encoders to the same tasks, introducing a modified Poly-encoder architecture better suited to MWE identification.

Our system's strength is its high precision for MWE identification, but it remains limited by its low recall. We show this to be a function of lexicon size, so one possible direction for future work could be expanding the lexicon by mining MWEs and generating their definitions, which has the potential to substantially increase recall for lexicon-based systems.

Future work in better approaches for multitask training of MWE identification/WSD models could also be valuable; the ideal preprocessing pipeline would be competitive with state-of-the-art systems in both tasks, and not just MWE identification.

References

Timothy Baldwin and Su Nam Kim. 2010. Multiword expressions. In *Handbook of Natural Language Processing*.

Edoardo Barba, Tommaso Pasini, and Roberto Navigli. 2021. ESC: Redesigning WSD with extractive sense comprehension. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4661–4672, Online. Association for Computational Linguistics.

Michele Bevilacqua, Marco Maru, and Roberto Navigli. 2020. Generationary or “how we went beyond word

[3] Computed excluding false-positives from the DiMSUM noun phrase detector, which does not use the lexicon

sense inventories and learned to gloss". In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7207–7221, Online. Association for Computational Linguistics.

Lukas Biewald. 2020. Experiment tracking with weights and biases. Software available from wandb.com.

Terra Blevins and Luke Zettlemoyer. 2020. Moving down the long tail of word sense disambiguation with gloss informed bi-encoders. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1006–1017, Online. Association for Computational Linguistics.

Mathieu Constant, Gülşen Eryiğit, Johanna Monti, Lonneke van der Plas, Carlos Ramisch, Michael Rosner, and Amalia Todirascu. 2017. Survey: Multiword expression processing: A Survey. *Computational Linguistics*, 43(4):837–892.

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.

William Falcon and The PyTorch Lightning team. 2019. PyTorch Lightning.

Vasiliki Foufi, Luka Nerima, and Éric Wehrli. 2017. Parsing and MWE detection: Fips at the PARSEME shared task. In *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, pages 54–59, Valencia, Spain. Association for Computational Linguistics.

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2020. Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring. In *International Conference on Learning Representations*.

Kamil Kanclerz and Maciej Piasecki. 2022. Deep neural representations for multiword expressions detection. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 444–453, Dublin, Ireland. Association for Computational Linguistics.

Minju Kim, Chaehyeong Kim, Yong Ho Song, Seungwon Hwang, and Jinyoung Yeo. 2022. BotsTalk: Machine-sourced framework for automatic curation of large-scale multi-skill dialogue datasets. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5149–5170, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Angelika Kirilin, Felix Krauss, and Yannick Versley. 2016. ICL-HD at SemEval-2016 task 10: Improving the detection of minimal semantic units and their meanings with an ontology and word embeddings. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 937–945, San Diego, California. Association for Computational Linguistics.

Harsh Kohli. 2021. Training bi-encoders for word sense disambiguation. *CoRR*, abs/2105.10146.

Nidhi Kulkarni and Mark Finlayson. 2011. jMWE: A Java toolkit for detecting multi-word expressions. In *Proceedings of the Workshop on Multiword Expressions: from Parsing and Generation to the Real World*, pages 122–124, Portland, Oregon, USA. Association for Computational Linguistics.

Murathan Kurfalı and Robert Östling. 2020. Disambiguation of potentially idiomatic expressions with contextual embeddings. In *Proceedings of the Joint Workshop on Multiword Expressions and Electronic Lexicons*, pages 85–94, online. Association for Computational Linguistics.

Zichao Li, Prakhar Sharma, Xing Han Lu, Jackie Cheung, and Siva Reddy. 2022. Using interactive feedback to improve the accuracy and explainability of question answering systems post-deployment. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 926–937, Dublin, Ireland. Association for Computational Linguistics.

Nelson F. Liu, Daniel Hershcovich, Michael Kranzlein, and Nathan Schneider. 2021. Lexical semantic recognition. In *Proceedings of the 17th Workshop on Multiword Expressions (MWE 2021)*, pages 49–56, Online. Association for Computational Linguistics.

Marco Maru, Simone Conia, Michele Bevilacqua, and Roberto Navigli. 2022. Nibbling at the hard core of Word Sense Disambiguation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4724–4737, Dublin, Ireland. Association for Computational Linguistics.

Marek Maziarz, Stan Szpakowicz, and Maciej Piasecki. 2015. A procedural definition of multi-word lexical units. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 427–435, Hissar, Bulgaria. INCOMA Ltd. Shoumen, BULGARIA.

George A. Miller. 1995. Wordnet: A lexical database for english. *Communications of the ACM*, 38(1):39–41.

George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. 1993. A semantic concordance. In *Human Language Technology: Proceedings of a Workshop Held at Plainsboro, New Jersey, March 21-24, 1993*.

Andrea Moro and Roberto Navigli. 2015. SemEval-2015 task 13: Multilingual all-words sense disambiguation and entity linking. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 288–297, Denver, Colorado. Association for Computational Linguistics.

Roberto Navigli, David Jurgens, and Daniele Vannella. 2013. SemEval-2013 task 12: Multilingual word sense disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 222–231, Atlanta, Georgia, USA. Association for Computational Linguistics.

Martha Palmer, Christiane Fellbaum, Scott Cotton, Lauren Delfs, and Hoa Trang Dang. 2001. English tasks: All-words and verb lexical sample. In *Proceedings of SENSEVAL-2 Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 21–24, Toulouse, France. Association for Computational Linguistics.

Caroline Pasquer, Agata Savary, Carlos Ramisch, and Jean-Yves Antoine. 2020. Verbal multiword expression identification: Do we need a sledgehammer to crack a nut? In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3333–3345, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Sameer Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007. SemEval-2007 task-17: English lexical sample, SRL and all words. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 87–92, Prague, Czech Republic. Association for Computational Linguistics.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 99–110, Valencia, Spain. Association for Computational Linguistics.

Carlos Ramisch, Silvio Ricardo Cordeiro, Agata Savary, Veronika Vincze, Verginica Barbu Mititelu, Archna Bhatia, Maja Buljan, Marie Candito, Polona Gantar, Voula Giouli, Tunga Güngör, Abdelati Hawwari, Uxoa Iñurrieta, Jolanta Kovalevskaitė, Simon Krek, Timm Lichte, Chaya Liebeskind, Johanna Monti, Carla Parra Escartín, Behrang QasemiZadeh, Renata Ramisch, Nathan Schneider, Ivelina Stoyanova, Ashwini Vaidya, and Abigail Walsh. 2018. Edition 1.1 of the PARSEME shared task on automatic identification of verbal multiword expressions. In *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, pages 222–240, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Carlos Ramisch, Aline Villavicencio, and Christian Boitet. 2010. mwetoolkit: a framework for multiword expression identification. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Omid Rohanian, Shiva Taslimipoor, Samaneh Kouchaki, Le An Ha, and Ruslan Mitkov. 2019. Bridging the gap: Attending to discontinuity in identification of multiword expressions. 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 2692–2698, Minneapolis, Minnesota. Association for Computational Linguistics.

Ivan Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for nlp. pages 1–15.

Agata Savary, Silvio Cordeiro, and Carlos Ramisch. 2019. Without lexicons, multiword expression identification will never fly: A position statement. In *Proceedings of the Joint Workshop on Multiword Expressions and WordNet (MWE-WN 2019)*, pages 79–91, Florence, Italy. Association for Computational Linguistics.

Nathan Schneider, Dirk Hovy, Anders Johannsen, and Marine Carpuat. 2016. SemEval-2016 task 10: Detecting minimal semantic units and their meanings (DiMSUM). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 546–559, San Diego, California. Association for Computational Linguistics.

Nathan Schneider, Jena D. Hwang, Vivek Srikumar, Jakob Prange, Austin Blodgett, Sarah R. Moeller, Aviram Stern, Adi Bitan, and Omri Abend. 2018. Comprehensive supersense disambiguation of English prepositions and possessives. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 185–196, Melbourne, Australia. Association for Computational Linguistics.

Benjamin Snyder and Martha Palmer. 2004. The English all-words task. In *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 41–43, Barcelona, Spain. Association for Computational Linguistics.

Yang Song, Xin Cai Ong, Hwee Tou Ng, and Qian Lin. 2021. Improved word sense disambiguation with enhanced sense representations. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4311–4320, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Shiva Taslimipoor, Omid Rohanian, and Le An Ha. 2019. Cross-lingual transfer learning and multitask learning for capturing multiword expressions. In

Proceedings of the Joint Workshop on Multiword Expressions and WordNet (MWE-WN 2019), pages 155–161, Florence, Italy. Association for Computational Linguistics.

Warren Weaver. 1949. Translation. In William N. Locke and A. Donald Boothe, editors, *Machine Translation of Languages*, pages 15–23. MIT Press, Cambridge, MA. Reprinted from a memorandum written by Weaver in 1949.

Jake Williams. 2017. Boundary-based MWE segmentation with text partitioning. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 1–10, Copenhagen, Denmark. 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.

Guobiao Zhang, Wenpeng Lu, Xueping Peng, Shoujin Wang, Baoshuo Kan, and Rui Yu. 2022. Word sense disambiguation with knowledge-enhanced and local self-attention-based extractive sense comprehension. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4061–4070, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

A Implementation Details

Bi-encoder and Poly-encoder models are implemented and trained with Pytorch Lightning (Falcon and The PyTorch Lightning team, 2019), using pretrained BERT models from the Transformers library (Wolf et al., 2020). All models were trained on a single GeForce GTX TITAN X GPU, with hyperparameters tuned using Weights & Biases (Biewald, 2020) to run random sweeps and track performance. Separate sweeps were run for the Bi-encider and Poly-encoder, each having a maximum of 20 runs and using early stopping to terminate runs with poor performance. Our total compute time was approximately 150 days (though this would have been significantly lower using a newer model of GPU), and our models have 220M parameters. Further detail, including all training hyperparameters and instructions for reproduction, can be found in our published code.

B Error Analysis Details

This appendix contain details about the frequency with which we found various types of false positives or false negatives in our error analysis.

B.1 PARSEME

In the table below, **Def?** represents the % of false positives where a definition appropriate for the predicted MWE was present in our lexicon. **MWE?** represents the % of false positives where the MWE was present in other sentences in the dataset, and the % of false negatives where it was present in our lexicon, respectively.

	False Positives		False Negatives
Model	**Def?**	**MWE?**	**MWE?**
SemCor	90%	16%	6%
fine-tuned	90%	34%	16%

Table 4: PARSEME Error Analysis

B.2 DiMSUM

Our results on DiMSUM are similar to those of PARSEME, except that for the system using the SemCor model 22% of the false positives were from the rule-based consecutive noun tagger, with that number increasing to 56% for the fine-tuned model (the false positive rate drops substantially after fine tuning the filtering model as can be seen in Table 2, which leads to these errors accounting for a higher percentage of total false positives). The **Def?** and **MWE?** percentages for false positives in the below table are computed excluding consecutive noun tagger false positives.

	False Positives		False Negatives
Model	**Def?**	**MWE?**	**MWE?**
SemCor	92%	56%	4%
fine-tuned	81%	63%	12%

Table 5: DiMSUM Error Analysis