Bi-Matching Mechanism to Combat the Long Tail of Word Sense Disambiguation

Anonymous ACL submission

Abstract

The long tail phenomenon of word sense distribution in linguistics causes the Word Sense Disambiguation (WSD) task to face a serious polarization of word sense distribution, that is, Most Frequent Senses (MFSs) with huge sample sizes and Long Tail Senses (LTSs) with small sample sizes. The single matching mechanism model that does not distinguish between the two senses will cause LTSs to be ignored because LTSs are in a weak position. The few-shot learning method that mainly focuses on LTSs is not conducive to grasping the advantage of easy identification of MFSs. This paper proposes a bi-matching mechanism to serve the WSD model to deal with two kinds of senses in a targeted manner, namely definition matching and collocation feature matching. The experiment is carried out under the evaluation framework of English all-words WSD and is better than the baseline models. Moreover, state-of-the-art performance is achieved through data enhancement.

1 Introduction

Word Sense Disambiguation (WSD) occupies an important position in the field of Natural Language Processing (NLP) (Bevilacqua et al., 2021), because the correct recognition of word senses has a direct and far-reaching impact on subsequent semantic understanding tasks, such as natural language understanding (Dewadkar et al., 2010; Mills and Bourbakis, 2014), machine translation (Neale et al., 2016; Rios Gonzales et al., 2017), etc.

WSD is to assign the correct sense to the target word according to the given context (Raganato et al., 2017b; Navigli, 2009). However, due to the long tail phenomenon of word sense distribution in linguistics, WSD models need to face the serious polarization of word sense distribution, that is, Most Frequent Senses (MFSs) with huge sample sizes and Long Tail Senses (LTSs) with small sample sizes (Li et al., 2021; Kumar et al., 2019). For example, the verb form of *Play*[1] has 35 senses in WordNet 3.1 (Miller, 1998), and most of the senses used are "*participate in games or sports*". There are a large number of LTSs that are rarely used, such as "*Princeton plays Yale this weekend*", that is, "*contend against an opponent in a sport, game, or battle*".

Traditional methods employ a single matching mechanism to complete WSD tasks. For example, Blevins and Zettlemoyer (2020) trained the glosses in WordNet as text embeddings to replace the original labels, which employ the definitions of the target word to match the most frequent and long-tail senses consistently. See also (Bevilacqua and Navigli, 2020; Scarlini et al., 2020b; Huang et al., 2019). The methods that pay attention to LTSs rely on MFSs to improve LTSs. For example, Holla et al. (2020) proposed a meta-learning framework for few-shot WSD, where the goal is to learn features from labeled instances for disambiguation of unseen words. See also (Du et al., 2021; Li et al., 2021; Kumar et al., 2019).

The single matching mechanism model that does not distinguish between the two senses will cause LTSs to be ignored, because the sample size of LTSs is at a disadvantage. The few-shot learning method that mainly focuses on LTSs is not conducive to grasping the advantages of easy identification of MFSs, and then affects the final effect. Considering that LTSs often appear in the form of fixed collocations, this paper proposes a **collocation feature matching mechanism for LTSs**; And considering that MFSs have clear definitions and are not easy to cause ambiguity, this paper proposes a **definition matching mechanism for MFSs**. In fact, this judgment is in line with facts:

- MFSs can be widely adopted, not only because of their wide range of applications, but also because of their clear definition and not

[1] http://wordnetweb.princeton.edu/perl/webwn?s=play

easy to cause ambiguity.

- The fundamental reason for the scarcity of LTS samples is their narrow scope of application, that is, they often appear in the form of fixed collocations.

To verify the effectiveness of the bi-matching mechanism, we conduct experiments under the evaluation framework of English all-words WSD, and the result is better than the baseline models. Moreover, to pursue better performance, we obtain state-of-the-art performance through data enhancement methods, that is, expanding multilingual datasets. The contributions of this article are summarized as follows:

- Propose a bi-matching mechanism to improve the recognition method of the WSD model that does not distinguish between most frequent and long-tail senses, and fill the gaps in research in this area;

- Implement extensive experimental verification and obtain state-of-the-art performance.

Codes and pre-trained models are available at https://github.com/yboys0504/wsd.

2 Related Work

2.1 Models with Different Matching Mechanisms

According to traditional classification methods (Bevilacqua et al., 2021; Raganato et al., 2017b), WSD models can be roughly divided into two categories, namely, supervised technology models and knowledge-based models.

Supervised technology models mainly employ a unified network structure to process all senses, and add a classifier at the end to calculate the probability distribution of sense labels. For example, Recurrent Neural Networks (RNN) suitable for sequence processing are often used as text processing networks, and use a normalized function to calculate the probability distribution in the output layer (Le et al., 2018; Kågebäck and Salomonsson, 2016; Yuan et al., 2016; Raganato et al., 2017a). Including the subsequent pre-training models, they all employ a similar architecture when dealing with WSD tasks (Scarlini et al., 2020a; Wiedemann et al., 2019; Hadiwinoto et al., 2019; Du et al., 2019; Huang et al., 2019). The advantage of such structures is that they can rely on the powerful learning ability of neural networks, but the disadvantage is that such data-driven models will seriously underestimate the long tail senses of the scarcity of sample sizes.

Knowledge-based models try to employ external knowledge to improve the recognition rate of WSD models, such as dictionary knowledge (Blevins and Zettlemoyer, 2020; Luo et al., 2018b), semantic network knowledge (Fernandez et al., 2018; Dongsuk et al., 2018), and multilingual knowledge (Pasini, 2020; Scarlini et al., 2020a). One of the most commonly used methods is to train text embeddings from the glosses in the dictionary to replace the labels (Blevins and Zettlemoyer, 2020; Scarlini et al., 2020b; Kumar et al., 2019). Such definition (or gloss) matching methods are good for identifying MFSs, but they are not good for identifying LTSs. The fundamental reason is that LTSs often appear in the form of fixed collocations and they are difficult to give a clear definition.

2.2 Models that Focus on Few-Shot

Subsequently, the researchers realized the importance of LTSs in the WSD task, and adopted some targeted solutions for LTSs, such as meta-learning, zero-shot learning, reinforcement learning, etc. Holla et al. (2020) proposed a meta-learning framework for few-shot WSD, where the goal is to learn features from labeled instances to disambiguate unseen words. See also (Du et al., 2021; Chen et al., 2021). Blevins and Zettlemoyer (2020) noticed the long-tail distribution of word sense in WSD tasks, and proposed a dual encoder model, that is, one Bert is used to extract the word embedding of the target word with contextual information, and another Bert is used to obtain the text embeddings of the glosses. The innovation of this work is the use of a dual-encoder joint training mechanism, but it still uses a consistent matching method for the most frequent and long-tail senses.

3 Methodology

In this section, we first explain the cognitive basis of our model derived from children's literacy behavior, then give a formal description of the WSD task, and finally clarify the structure of our model in formal language.

Masaru Ibuka (Ibuka, 1977), a Japanese educator, pointed out that children's literacy behavior is

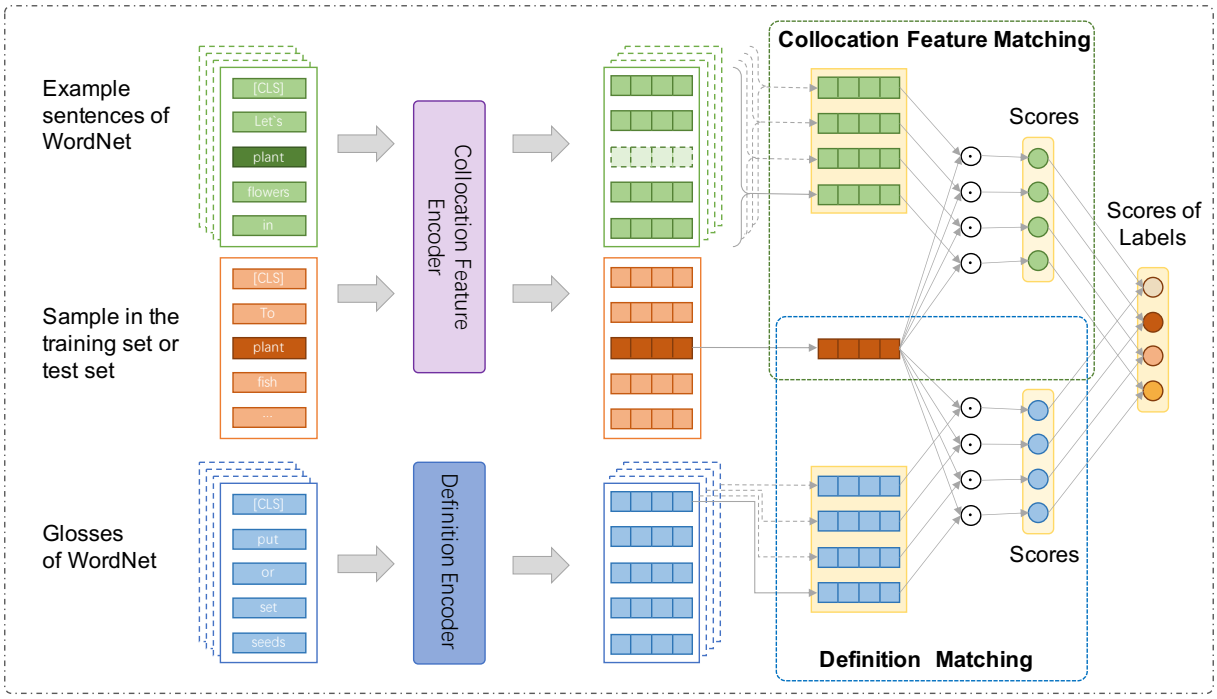

Figure 1: Schematic diagram of the Bi-MWSD architecture, which illustrates the disambiguation process of the target word *Plant*. ⊙ represents the dot product of the vector.

mainly based on mechanical memory and recognition ability in the early stage, and then gradually develops concept-oriented memory and recognition ability in the later stage. The mechanical method rigidly remembers the structure of the word itself and its application scenarios, such as collocation features of the word. The concept-oriented method establishes the relationship between the structure, meaning, and usage of words through analysis and comparison, such as the definitions given in the dictionary.

For the WSD task, we should not only pay attention to MFSs, but also LTSs, because LTSs are an important bottleneck for the development of WSD. For MFSs, it is reasonable to distinguish senses through the definition system, because theoretically, the definition system of word senses can clearly distinguish different MFSs. But for LTSs, it is difficult to define a clear and non-confusing definition system for each sense. For example, "*go to plant fish*", where *plant* means "*place into a river*". This meaning of *Plant* mostly appears in such a collocation form. Therefore, considering the characteristics of LTSs, the collocation feature matching method is more suitable for identifying LTSs.

In this article, we propose a bi-matching mechanism model to complete the WSD task (called **Bi-MWSD**), namely the **collocation feature matching** and the **definition matching**. We will give a detailed description of the construction details and operation process of Bi-MWSD in Sec. 3.2.

3.1 Word Sense Disambiguation

WSD is to predict the senses of the target word in a given context. The formal definition can be expressed as: predict the possible sense $s \in S_{\hat{w}}$ of the target word $\hat{w}$ in a given context $C_{\hat{w}}$, which is

$$f(\hat{w}, C_{\hat{w}}) = s \in S_{\hat{w}} \quad (1)$$

where $S_{\hat{w}}$ is the candidate list of the senses of $\hat{w}$, and f refers to the WSD model.

All-words WSD is to predict all ambiguous words in a given context. This means that the WSD model may predict the noun, verb, adjective, and adverb forms of ambiguous words. In this case, the input and output of the WSD model are defined as $C = (..., w_i, ...)$ and $S = (..., s^x_{w_i}, ...)$, respectively, where $s^x_{w_i}$ represents the x^{th} sense of the target word w_i.

3.2 Bi-Matching Mechanism for WSD

The architecture of Bi-MWSD is shown in Fig. 1. Bi-MWSD uses two pre-trained models, that is, Bert-base (Devlin et al., 2019), as text feature encoders. One encoder is used to extract the collocation features of the target word in the training

samples and the example sentences, which is called the **collocation feature encoder**. The other is used to extract the definition system in the glosses of the target word, which is called the **definition encoder**. The example sentences and glosses come from the examples and definitions corresponding to each sense in WordNet. The last step is the matching process of MFSs and LTSs, which is called **matching word senses**.

3.2.1 Collocation Feature Encoder

The function of the collocation feature encoder is to memorize the collocation features of the target word, such as the structure and relationship between the target word and the collocation words, and the entire application scenario. The encoder will process two kinds of texts. One is the example sentences corresponding to each sense of the target word in WordNet, $E^x = (..., e_k^x, ...)$ where e_k^x represents the k^{th} word of the example sentence E^x of the x^{th} sense of the target word. And the other is the training samples containing the target word, $C = (..., w_i, ...)$ where w_i represents the i^{th} word. The text is encoded using Bert standard processing rules, such as adding $[CLS]$ and $[SEP]$ marks at the beginning and end of the text respectively,

$$E^x = ([CLS], ..., e_k^x, ..., [SEP]) \quad (2)$$
$$= (e_{cls}^x, ..., e_k^x, ..., e_{sep}^x). \quad (3)$$

The encoder will encode each word (including the added $[CLS]$ and $[SEP]$) to obtain a corresponding 768-dimensional vector. The processing method of the training samples is also the same. *In WordNet 3.0, there are cases where multiple example sentences are given for one sense, and for this, we only choose the first one by default.*

The reason why we use one encoder to process two kinds of texts here is that both the example sentences and the training samples contain the target word, which can all be considered that there are collocation features of the target word. Moreover, the advantage of this processing is that the training sample will truly reflect the frequency of each sense of the target word, and the example sentence can provide the collocation features of LTSs. Processing them together can make up for the lack of scene information of LFSs, but it will not (seriously) change their frequency.

After processing by the collocation feature encoder, we can get the vector representation of the target word in the sample, i.e., $v_{\hat{w}}$, and the vector representation of the collocation features of each sense x provided by the example sentences, i.e., V_{E^x}. Here we provide two calculation methods for V_{E^x}, namely, the overall text vector minus the target word vector,

$$V_{E^x} = v_{e_{cls}^x} - v_{e_{\hat{w}}^x}, \quad (4)$$

and the vectors except the target word vector are added,

$$V_{E^x} = \sum_k v_{e_k^x} - v_{e_{\hat{w}}^x}. \quad (5)$$

Through experimental analysis of these two methods, we found that the first one is relatively better. The possible reason is that it can not only characterize the collocation features of the target word, but also remember the entire text, that is, the application scenario.

3.2.2 Definition Encoder

The definition encoder constructs the definition system of the target word by learning the glosses G^x for each sense x in WordNet, $G^x = (..., g_j^x, ...)$ where g_j^x represents the j^{th} word of the gloss text of the x^{th} sense of the target word. The gloss is a simple and accurate summary of the word sense, so it is suitable for refining the definition system of the target word. What needs to be emphasized here is that the target word itself is not included in the gloss, i.e., $\hat{w} \notin G^x$, so the collocation feature of the target word cannot be extracted. The input of the encoder is some gloss text. It is also necessary to add $[LCS]$ and $[SEP]$ marks to the gloss text before encoding,

$$G^x = ([CLS], ..., g_j^x, ..., [SEP]) \quad (6)$$
$$= (g_{cls}^x, ..., g_j^x, ..., g_{sep}^x). \quad (7)$$

The encoder will encode each word (including the added $[CLS]$ and $[SEP]$) to obtain a corresponding 768-dimensional vector. Here we choose the output vector corresponding to $[CLS]$, i.e., $v_{g_{cls}^x}$, to represent the entire gloss text, i.e., $V_{G^x} = v_{g_{cls}^x}$. This method is a common practice in the industry.

3.2.3 Matching Word Senses

At this point, we can calculate the score of each sense of the target word $\hat{w}$ in a given context C,

$$Score(\hat{w}|C) = F(\{v_{\hat{w}} \odot (\alpha V_{G^x} + \beta V_{E^x})\}^x) \quad (8)$$

where α and β respectively represent the proportion of the definition matching method and the collocation feature matching method. Here α and β

can be the weights learned by the model itself, or they can be the proportions of each sense provided by WordNet. Through experimental analysis, we found that they work best when they are set to the same value. $F(\cdot)$ can be a standard *Softmax* or other distribution function. When $F(\cdot)$ is selected as *Softmax*, $Score(\hat{w}|C)$ is a probability distribution of each sense of the target word in a given context. Finally, we can conclude that the one with the highest probability is the most likely sense.

3.2.4 Parameter Optimization

We use a cross-entropy loss on the scores of the candidate senses of the target word to train Bi-MWSD. The loss function is

$$Loss(Score, index) \tag{9}$$

$$= -\log\left(\frac{\exp(Score^{[index]})}{\sum_{i=1}\exp(Score^{[i]})}\right) \tag{10}$$

$$= -Score^{[index]} + \log\sum_{i=1}\exp(Score^{[i]}) \tag{11}$$

where $index$ is the index of the list of the candidate senses of the target word.

Bi-MWSD employs Adam optimizer (Kingma and Ba, 2015) to update the parameters of the model, and the specific settings of the optimizer will be given in the experimental section.

4 Experiments

4.1 Datasets

To evaluate Bi-MWSD comprehensively and objectively, we propose four evaluation settings:

- Standard evaluation setting (**S-setting**), which is the standard evaluation framework proposed by Raganato et al. (2017b), that is, only SemCor[2] is used as the training set;
- High-end evaluation setting (**H-setting**), which extends OMSTI[2] and multilingual datasets[3] as training sets;
- MFS evaluation setting (**MFS-setting**), which is based on the standard evaluation setting, modifying the number of LTSs in the training set to make them MFSs (here the number of samples is expanded in a repeated manner);
- LTS evaluation setting (**LTS-setting**), which is also to adjust the number of MFSs in the training set to make them LTSs. For example, let MFSs appear only 1, 2, 3, or 5 times like LTSs.

It can be found that their main difference lies in the different training sets. The development set and test set are given below. In addition, we selected all word senses in WordNet 3.0 as candidate senses of the target word.

Following previous work (Luo et al., 2018a; Huang et al., 2019; Blevins and Zettlemoyer, 2020), we employ SemEval-2007 (**SE07**; Pradhan et al., 2007) as the development set, and hold out Senseval-2 (**SE2**; Edmonds and Cotton, 2001), Senseval-3 (**SE3**; Snyder and Palmer, 2004), SemEval-2013 (**SE13**; Navigli et al., 2013), and SemEval-2015 (**SE15**; Moro and Navigli, 2015) as test sets. The statistical information of each dataset is shown in Tab. 1.

Dataset	#Sens	#Toks	#Anns	#Typs	#Amb
SE07	3	3,201	455	375	8.5
SE2	3	5,766	2,282	1,335	5.4
SE3	3	5,541	1,850	1,167	6.8
SE13	13	8,391	1,644	827	4.9
SE15	4	2,604	1,022	659	5.5

Table 1: Statistics of the datasets: the number of sentences (#Sens), tokens (#Toks), sense annotations (#Anns), sense types covered (#Typs) in each dataset. #Amb refers to the ambiguity level, which implies the difficulty of a given dataset.

4.2 Baseline Models

To evaluate Bi-MWSD comprehensively and objectively, we compare classic methods and SOTA models in recent years.

For the classic methods, we choose the classic and representative knowledge-based method **Babelfy** (Moro et al., 2014) and supervised method **IMS+emb** (Iacobacci et al., 2016). Babelfy is an entity linking algorithm based on the semantic network BabelNet; IMS+emb is a supervised method that integrates word embedding as features under the IMS framework. In addition, **Context2Vec** and **BERT-base** are also listed. The experimental results of Babelfy and IMS+emb are derived from the original paper, and Context2Vec and BERT-base are published by Loureiro and Jorge (2019) and Blevins and Zettlemoyer (2020) respectively.

[2] http://lcl.uniroma1.it/wsdeval/training-data

[3] https://github.com/SapienzaNLP/mwsd-datasets

For the related work of the past two years, the models with multilingual knowledge (*ML*) include **SyntagRank** (Scozzafava et al., 2020), **SensEmBERT$_{sup}$** (Scarlini et al., 2020a) and **ARES** (Scarlini et al., 2020b); the models with knowledge graphs (*KG*) include **EWISER** (Bevilacqua and Navigli, 2020) and (Conia and Navigli, 2021); the models with glosses (*GL*) include **EWISE** (Kumar et al., 2019), **GlossBERT** (Huang et al., 2019), **EWISER** (Bevilacqua and Navigli, 2020), (Berend, 2020) and (Yap et al., 2020). The experimental results come from the values published in the original paper.

4.3 Model Setting

Bi-MWSD (under S-setting, H-setting, MFS-setting, and LTS-setting) is designed with Pytorch 1.8[4] and uses Python 3.6[5]. It uses two GPUs (Tesla P40) to load two encoders separately for joint training. The encoder version is *bert-base-uncased* (Devlin et al., 2019); the epoch is 20; the batch size of the target context is 4; the batch size of example sentences and glosses are both 256; the learning rate is 5e-6 and 1e-5. Bi-MWSD uses *Adam* (Kingma and Ba, 2015) as the optimizer to adjust the parameters.

Other unlisted super-parameters are indicated in the published code.

4.4 Experimental Results

Tab. 2 shows the F1-score of the comparison models and Bi-MWSD on the English all-words WSD task. It needs to be emphasized that the results are experimental results in the full senses, not in the most frequent senses.

Under the classic methods, it can be seen from the results that the method of integrating Context2Vec or Word2Vec will significantly improve the overall effect; the method based on the pre-trained model does show strong advantages.

Under S-setting, Bi-MWSD performs better than all baseline models; moreover, it can be seen that the pre-training model has a unified trend, and the model for mining dictionary knowledge and language knowledge has become the mainstream. The performance of Bi-MWSD also supports this point.

Under H-setting, Bi-MWSD achieves state-of-the-art performance compared with other knowledge-based methods that use pre-trained language models. It proves that targeted processing of the most frequent and long-tail senses is beneficial to the final result.

[4] https://pytorch.org/

[5] https://www.python.org/

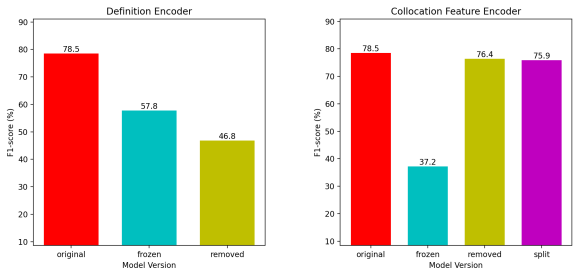

Figure 2: The figure shows the results of ablation experiments for each encoder, namely definition encoder and collocation feature encoder. The experimental result is the overall result under all test sets, namely **ALL** in Tab. 2.

Analysis of poor performance on indicators *Adj.* and *Adv.* of Tab. 2: In linguistics, nouns and verbs are words with a serious long-tail, and adjectives and adverbs are relatively weaker. In other words, there are fewer LTSs in adjectives and adverbs. For datasets where the proportion of LTSs is not high, the method of not distinguishing or ignoring LTSs has advantages.

4.5 Ablation Study

Bi-MWSD uses a bi-matching mechanism to complete the WSD task, that is, **definition matching** and **collocation feature matching**. A detailed analysis of the contribution of each part to the overall effect is necessary. Therefore, we will use the method of ablation experiment to verify.

4.5.1 Ablation Study for Definition Matching

For the analysis of the definition matching mechanism, we use the method of **ablation function** (i.e., frozen the encoder) and **ablation module** (i.e., directly remove the encoder). The method of freezing the encoder will prevent the encoder from fine-tuning the parameters on the training set, that is, preventing the encoder from learning more semantic information on the training set. We know that LTSs are marked in the training set. Preventing the encoder from fine-tuning the parameters on the training set will hinder the encoder's ability to recognize LTSs. Compared with the original model, this method will directly reflect the contribution of the defined encoder to solving LTSs. The method of removing the encoder is more direct, which directly reflects the contribution of the

	Model	Dev	Test Datasets				Concatenation of all Datasets				
		SE07	SE2	SE3	SE13	SE15	Nouns	Verbs	Adj.	Adv.	**ALL**
Classic Methods											
KG	Babelfy (2014)	51.6	67.0	63.5	66.4	70.3	68.9	50.7	73.2	79.8	66.4
EM	IMS+emb (2016)	62.6	72.2	70.4	65.9	71.5	71.9	56.6	75.9	84.7	70.1
	Context2Vec (2016)	61.3	71.8	69.1	65.6	71.9	71.0	57.6	75.2	82.7	69.0
Bert	BERT-base (2019)	68.6	75.9	74.4	70.6	75.2	75.7	63.7	78.0	85.8	73.7
Baseline Models under S-setting											
ML	SyntagRank (2020)	59.3	71.6	72.0	72.2	75.8	-	-	-	-	71.7
	SensEmBERT$_{sup}$ (2020a)	60.2	72.2	69.9	78.7	75.0	80.5	50.3	74.3	80.9	72.8
KG	EWISER (2020)	71.0	77.5	77.9	76.4	77.8	79.9	66.4	79.0	85.5	77.0
	Conia and Navigli (2021)	72.2	78.4	77.8	76.7	78.2	80.1	67.0	80.5	86.2	77.6
GL	EWISE (2019)	67.3	73.8	71.1	69.4	74.5	74.0	60.2	78.0	82.1	71.8
	GlossBERT (2019)	72.5	77.7	75.2	76.1	80.4	79.8	67.1	**79.6**	**87.4**	77.0
	Berend (2020)	68.8	77.9	77.8	76.1	77.5	-	-	-	-	76.8
	Bi-MWSD	**75.2**	**78.4**	**77.9**	**78.8**	**80.8**	**81.0**	**68.7**	78.5	85.5	**78.5**
SOTA Models under H-setting											
ML	ARES (2020b)	71.0	78.0	77.1	77.3	83.2	80.6	68.3	80.5	83.5	77.9
KG	EWISER(2020)	75.2	80.8	79.0	80.7	81.8	82.9	69.4	82.9	**87.6**	80.1
	Conia and Navigli (2021)	76.2	80.4	77.8	81.8	83.3	82.9	70.3	83.4	85.5	80.2
GL	Berend (2020)	73.0	79.6	77.3	79.4	71.3	-	-	-	-	78.8
	ESCHER(2021)	76.3	**81.7**	77.8	82.2	83.2	83.9	69.3	**83.8**	86.7	80.7
GL+ES	Yap et al. (2020)	73.6	79.4	76.8	77.4	81.5	80.6	67.9	82.2	87.3	78.2
	Bi-MWSD	**77.3**	80.8	**79.9**	**83.8**	**83.7**	**84.0**	**70.7**	81.5	86.5	**81.5**

Table 2: F1-score (%) on the English all-words WSD task. **ALL** is the concatenation of all test sets and development set; Dev refers to the development sets. The ones in bold are the best results among all the models. *KG*, *EM*, *Bert*, *ML*, *GL*, and *ES* respectively refer to the knowledge graph, word or text embedding, Bert model, multilingual knowledge, gloss, and example sentence used in the model.

definition matching method to the overall model.

We separately freeze and remove the definition encoder on the original model, and adjust the hyperparameters to get the best results. The experimental results are shown in Fig. 2.

1. Comparing the original version and the frozen version, it can be seen that the definition encoder can indeed learn new semantic knowledge by fine-tuning the parameters on the training set, and it can greatly improve the overall result.

2. Comparing the original version and the removed version, it can be seen that the contribution of the definition encoder to the overall effect is huge. This result is in line with reality, because MFSs are indeed far greater than the usage rate of LTSs in life, and the function of the definition encoder is reflected in the recognition of MFSs. Similarly, comparing the frozen version with the removed version confirms this.

4.5.2 Ablation Study for Collocation Feature Matching

For the analysis of the collocation feature matching mechanism, in addition to the **ablation function** and **ablation module**, we also need to **disassemble the two functions** of the collocation feature encoder, that is, target word vectorization and example sentence vectorization. It should be emphasized that the removed version here only removes the example sentence learning function of the encoder. We fine-tune the hyperparameters of the modified versions to obtain the best results. The experimental results are shown in Fig. 2.

1. Comparing the original version and the frozen version, it can be seen that the model will show the worst case without fine-tuning the parameters under the training set. The main reason is that the encoder is responsible for the learning of the target word vector. If there is no good target word representation, it will directly affect the overall result.

2. Comparing the original version with the removed version, that is, removing collocation feature matching, it can be seen that intro-

ducing this matching mechanism can indeed improve the effectiveness of the model. Although there is only two percentage point improvement, considering the difficulty of LTS recognition, it also shows that our model does contribute to the recognition of LTSs.

3. Regarding whether the training process of merging the target word and the collocation feature can improve the overall effect of the model, we can compare the results of the original version and the split version. An improvement of close to 3% proves that this design is reasonable. Example sentences of LTSs in the dictionary improve the ability of the pre-trained model to represent low-tail target words.

5 Evaluation for LTSs and MFSs

Compared with the ablation experiments, the significance of this section is more to test the potential of Bi-MWSD in terms of MFSs and LTSs. The LTS-setting is equivalent to evaluating the performance of the model in few-shot WSD; the MFS-setting is equivalent to testing the performance of the model in practical scenarios.

To compare science, we chose two models that are similar to our model as the baseline, that is, GlossBERT (Huang et al., 2019) and BEM (Blevins and Zettlemoyer, 2020). Similar to our model, both GlossBERT and BEM employ Bert-base to extract text features of glosses in the dictionary; BEM also uses two pre-trained models as dual encoder models.

5.1 Bi-MWSD under LTS-setting

Under LTS-setting, the experimental results of GlossBert, BEM and Bi-MWSD are shown in Tab. 3. Experimental results show that the bi-matching mechanism is indeed conducive to the recognition of LTSs, indicating that the recognition or evaluation of word senses from multiple angles can improve the recognition rate. The inspiration of this result for few-shot research is that few-shot tasks can be completed from the perspective of multi-angle evaluation.

5.2 Bi-MWSD under MFS-setting

Under MFS-setting, the experimental results on BEM, GlossBERT, and Bi-MWSD are shown in Tab. 4. According to the experimental results, our model performs best, but its advantages are small.

Model	ALL			
	1-shot	2-shot	3-shot	5-shot
GlossBERT	56.8	55.9	62.1	67.7
BEM	**68.7**	68.7	70.4	70.5
Bi-MWSD	67.3	**71.3**	**72.0**	**73.0**

Table 3: The experimental results of GlossBERT, BEM and Bi-MWSD under LTS-setting.

It shows that the contribution of the bi-matching mechanism to MFSs that is easy to identify is limited. It also proves from the side that the definition (gloss) is directly and effective in terms of MFSs, and the advantages of collocation features are not obvious compared with the definition.

Model	Dev	**ALL**
GlossBERT (2019)	**74.7**	78.0
BEM (2020)	74.5	79.0
Bi-MWSD	74.3	**79.1**

Table 4: The experimental results of GlossBERT, BEM and Bi-MWSD under MFS-setting.

6 Conclusion

By analyzing the characteristics of most frequent and long-tail senses, we respectively propose targeted matching methods, namely, using definition matching mechanism for MFSs and collocation feature matching mechanism for LTSs. This method achieves the purpose of identifying the most frequent and long-tail senses at the same time with the same model. Compared with early models that mainly focused on MFSs, our model does not ignore the value of LTSs; compared with few-shot learning or meta-learning methods, our model can capture the advantage of easy identification of MFSs. The main contribution of this paper is to fill the gaps in the multi-matching mechanism in the task of WSD.

The experimental results show that designing targeted recognition methods for different word sense types is effective for improving the overall performance of the WSD task. In future research, we will design the more targeted multi-matching mechanism models. Moreover, we will also try to use the multi-matching mechanism in other few-shot tasks.

References

Edoardo Barba, Tommaso Pasini, and Roberto Navigli. 2021. Esc: Redesigning wsd with extractive sense comprehension. In *NAACL*.

Gábor Berend. 2020. Sparsity makes sense: Word sense disambiguation using sparse contextualized word representations. In *EMNLP*.

Michele Bevilacqua and R. Navigli. 2020. Breaking through the 80% glass ceiling: Raising the state of the art in word sense disambiguation by incorporating knowledge graph information. In *ACL*.

Michele Bevilacqua, Tommaso Pasini, Alessandro Raganato, and Roberto Navigli. 2021. Recent trends in word sense disambiguation: A survey. In *IJCAI*.

Terra Blevins and Luke Zettlemoyer. 2020. Moving down the long tail of word sense disambiguation with gloss informed bi-encoders. In *ACL*.

Howard Chen, M. Xia, and Danqi Chen. 2021. Non-parametric few-shot learning for word sense disambiguation. *ArXiv*, abs/2104.12677.

Simone Conia and Roberto Navigli. 2021. Framing word sense disambiguation as a multi-label problem for model-agnostic knowledge integration. In *EACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Dnyaneshwar A. Dewadkar, Y. Haribhakta, P. Kulkarni, and Pradnya D. Balvir. 2010. Unsupervised word sense disambiguation in natural language understanding. In *IC-AI*.

O Dongsuk, Sunjae Kwon, Kyungsun Kim, and Youngjoong Ko. 2018. Word sense disambiguation based on word similarity calculation using word vector representation from a knowledge-based graph. In *COLING*.

Jiaju Du, Fanchao Qi, and Maosong Sun. 2019. Using bert for word sense disambiguation. *ArXiv*, abs/1909.08358.

Yingjun Du, Nithin Holla, Xiantong Zhen, Cees Snoek, and Ekaterina Shutova. 2021. Meta-learning with variational semantic memory for word sense disambiguation. In *ACL/IJCNLP*.

Philip Edmonds and Scott Cotton. 2001. SENSEVAL-2: overview. In *Proceedings of Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 1–5.

Andres Duque Fernandez, Mark Stevenson, Juan Martínez-Romo, and Lourdes Araujo. 2018. Co-occurrence graphs for word sense disambiguation in the biomedical domain. *Artificial intelligence in medicine*, 87:9–19.

Christian Hadiwinoto, H. Ng, and Wee Chung Gan. 2019. Improved word sense disambiguation using pre-trained contextualized word representations. In *EMNLP/IJCNLP*.

Nithin Holla, Pushkar Mishra, H. Yannakoudakis, and Ekaterina Shutova. 2020. Learning to learn to disambiguate: Meta-learning for few-shot word sense disambiguation. In *FINDINGS*.

Luyao Huang, Chi Sun, Xipeng Qiu, and Xuanjing Huang. 2019. GlossBERT: BERT for word sense disambiguation with gloss knowledge. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3509–3514.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for word sense disambiguation: An evaluation study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 897–907.

Masaru Ibuka. 1977. *Kindergarten is Too Late!* Souvenir Press London.

Mikael Kågebäck and Hans Salomonsson. 2016. Word sense disambiguation using a bidirectional lstm. In *CogALex@COLING*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Sawan Kumar, S. Jat, Karan Saxena, and P. Talukdar. 2019. Zero-shot word sense disambiguation using sense definition embeddings. In *ACL*.

Minh Nguyen Le, Marten Postma, Jacopo Urbani, and P. Vossen. 2018. A deep dive into word sense disambiguation with lstm. In *COLING*.

Wei Li, Harish Tayyar Madabushi, and Mark Lee. 2021. UoB_UK at SemEval 2021 task 2: Zero-shot and few-shot learning for multi-lingual and cross-lingual word sense disambiguation. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, pages 738–742, Online. Association for Computational Linguistics.

Daniel Loureiro and Alípio Jorge. 2019. Language modelling makes sense: Propagating representations through WordNet for full-coverage word sense disambiguation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5682–5691.

Fuli Luo, Tianyu Liu, Zexue He, Qiaolin Xia, Zhifang Sui, and Baobao Chang. 2018a. Leveraging gloss knowledge in neural word sense disambiguation by hierarchical co-attention. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1402–1411.

Fuli Luo, Tianyu Liu, Qiaolin Xia, Baobao Chang, and Zhifang Sui. 2018b. Incorporating glosses into neural word sense disambiguation. In *ACL*.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61.

George A Miller. 1998. *WordNet: An electronic lexical database*. MIT press.

Michael T. Mills and N. Bourbakis. 2014. Graph-based methods for natural language processing and understanding—a survey and analysis. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44:59–71.

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@NAACL-HLT 2015, Denver, Colorado, USA, June 4-5, 2015*, pages 288–297.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244.

R. Navigli. 2009. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41:10:1–10:69.

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.

Steven Neale, Luís Manuel dos Santos Gomes, Eneko Agirre, Oier Lopez de Lacalle, and A. Branco. 2016. Word sense-aware machine translation: Including senses as contextual features for improved translation models. In *LREC*.

Tommaso Pasini. 2020. The knowledge acquisition bottleneck problem in multilingual word sense disambiguation. In *IJCAI*.

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.

Alessandro Raganato, Claudio Delli Bovi, and R. Navigli. 2017a. Neural sequence learning models for word sense disambiguation. In *EMNLP*.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017b. 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.

Annette Rios Gonzales, Laura Mascarell, and Rico Sennrich. 2017. Improving word sense disambiguation in neural machine translation with sense embeddings. In *Proceedings of the Second Conference on Machine Translation*, pages 11–19.

Bianca Scarlini, Tommaso Pasini, and R. Navigli. 2020a. Sensembert: Context-enhanced sense embeddings for multilingual word sense disambiguation. In *AAAI*.

Bianca Scarlini, Tommaso Pasini, and R. Navigli. 2020b. With more contexts comes better performance: Contextualized sense embeddings for all-round word sense disambiguation. In *EMNLP*.

Federico Scozzafava, Marco Maru, Fabrizio Brignone, Giovanni Torrisi, and Roberto Navigli. 2020. Personalized PageRank with syntagmatic information for multilingual word sense disambiguation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–46.

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.

Gregor Wiedemann, Steffen Remus, Avi Chawla, and Chris Biemann. 2019. Does bert make any sense? interpretable word sense disambiguation with contextualized embeddings. *ArXiv*, abs/1909.10430.

Boon Peng Yap, Andrew Koh Jin Jie, and Chng Eng Siong. 2020. Adapting bert for word sense disambiguation with gloss selection objective and example sentences. *ArXiv*, abs/2009.11795.

Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. 2016. Semi-supervised word sense disambiguation with neural models. In *COLING*.