Improved Word Sense Disambiguation Using Pre-Trained Contextualized Word Representations

Christian Hadiwinoto, Hwee Tou Ng, and **Wee Chung Gan**
Department of Computer Science, National University of Singapore
christian.hadiwinoto@u.nus.edu, nght@comp.nus.edu.sg,
gan_weechung@u.nus.edu

Abstract

Contextualized word representations are able to give different representations for the same word in different contexts, and they have been shown to be effective in downstream natural language processing tasks, such as question answering, named entity recognition, and sentiment analysis. However, evaluation on word sense disambiguation (WSD) in prior work shows that using contextualized word representations does not outperform the state-of-the-art approach that makes use of non-contextualized word embeddings. In this paper, we explore different strategies of integrating pre-trained contextualized word representations and our best strategy achieves accuracies exceeding the best prior published accuracies by significant margins on multiple benchmark WSD datasets.

1 Introduction

Word sense disambiguation (WSD) automatically assigns a pre-defined sense to a word in a text. Different senses of a word reflect different meanings a word has in different contexts. Identifying the correct word sense given a context is crucial in natural language processing (NLP). Unfortunately, while it is easy for a human to infer the correct sense of a word given a context, it is a challenge for NLP systems. As such, WSD is an important task and it has been shown that WSD helps downstream NLP tasks, such as machine translation (Chan et al., 2007a) and information retrieval (Zhong and Ng, 2012).

A WSD system assigns a sense to a word by taking into account its context, comprising the other words in the sentence. This can be done through discrete word features, which typically involve surrounding words and collocations trained using a classifier (Lee et al., 2004; Ando, 2006; Chan et al., 2007b; Zhong and Ng, 2010). The classifier can also make use of continuous word representations of the surrounding words (Taghipour and Ng, 2015; Iacobacci et al., 2016). Neural WSD systems (Kågebäck and Salomonsson, 2016; Raganato et al., 2017b) feed the continuous word representations into a neural network that captures the whole sentence and the word representation in the sentence. However, in both approaches, the word representations are independent of the context.

Recently, pre-trained contextualized word representations (Melamud et al., 2016; McCann et al., 2017; Peters et al., 2018; Devlin et al., 2019) have been shown to improve downstream NLP tasks. Pre-trained contextualized word representations are obtained through neural sentence encoders trained on a huge amount of raw texts. When the resulting sentence encoder is fine-tuned on the downstream task, such as question answering, named entity recognition, and sentiment analysis, with much smaller annotated training data, it has been shown that the trained model, with the pre-trained sentence encoder component, achieves new state-of-the-art results on those tasks.

While demonstrating superior performance in downstream NLP tasks, pre-trained contextualized word representations are still reported to give lower accuracy compared to approaches that use non-contextualized word representations (Melamud et al., 2016; Peters et al., 2018) when evaluated on WSD. This seems counter-intuitive, as a neural sentence encoder better captures the surrounding context that serves as an important cue to disambiguate words. In this paper, we explore different strategies of integrating pre-trained contextualized word representations for WSD. Our best strategy outperforms prior methods of incorporating pre-trained contextualized word representations and achieves new state-of-the-art accuracy on multiple benchmark WSD datasets.

The following sections are organized as follows.

Section 2 presents related work. Section 3 describes our pre-trained contextualized word representation. Section 4 proposes different strategies to incorporate the contextualized word representation for WSD. Section 5 describes our experimental setup. Section 6 presents the experimental results. Section 7 discusses the findings from the experiments. Finally, Section 8 presents the conclusion.

2 Related Work

Continuous word representations in real-valued vectors, or commonly known as word embeddings, have been shown to help improve NLP performance. Initially, exploiting continuous representations was achieved by adding real-valued vectors as classification features (Turian et al., 2010). Taghipour and Ng (2015) fine-tuned non-contextualized word embeddings by a feed-forward neural network such that those word embeddings were more suited for WSD. The fine-tuned embeddings were incorporated into an SVM classifier. Iacobacci et al. (2016) explored different strategies of incorporating word embeddings and found that their best strategy involved exponential decay that decreased the contribution of surrounding word features as their distances to the target word increased.

The neural sequence tagging approach has also been explored for WSD. Kågebäck and Salomonsson (2016) proposed bidirectional long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) for WSD. They concatenated the hidden states of the forward and backward LSTMs and fed the concatenation into an affine transformation followed by softmax normalization, similar to the approach to incorporate a bidirectional LSTM adopted in sequence labeling tasks such as part-of-speech tagging and named entity recognition (Ma and Hovy, 2016). Raganato et al. (2017b) proposed a self-attention layer on top of the concatenated bidirectional LSTM hidden states for WSD and introduced multi-task learning with part-of-speech tagging and semantic labeling as auxiliary tasks. However, on average across the test sets, their approach did not outperform SVM with word embedding features. Subsequently, Luo et al. (2018) proposed the incorporation of glosses from WordNet in a bidirectional LSTM for WSD, and reported better results than both SVM and prior bidirectional LSTM models.

A neural language model (LM) is aimed at predicting a word given its surrounding context. As such, the resulting hidden representation vector captures the context of a word in a sentence. Melamud et al. (2016) designed *context2vec*, which is a one-layer bidirectional LSTM trained to maximize the similarity between the hidden state representation of the LSTM and the target word embedding. Peters et al. (2018) designed ELMo, which is a two-layer bidirectional LSTM language model trained to predict the next word in the forward LSTM and the previous word in the backward LSTM. In both models, WSD was evaluated by nearest neighbor matching between the test and training instance representations. However, despite training on a huge amount of raw texts, the resulting accuracies were still lower than those achieved by WSD approaches with pre-trained non-contextualized word representations.

End-to-end neural machine translation (NMT) (Sutskever et al., 2014; Bahdanau et al., 2015) learns to generate an output sequence given an input sequence, using an encoder-decoder model. The encoder captures the contextualized representation of the words in the input sentence for the decoder to generate the output sentence. Following this intuition, McCann et al. (2017) trained an encoder-decoder model on parallel texts and obtained pre-trained contextualized word representations from the encoder.

3 Pre-Trained Contextualized Word Representation

The contextualized word representation that we use is BERT (Devlin et al., 2019), which is a bidirectional transformer encoder model (Vaswani et al., 2017) pre-trained on billions of words of texts. There are two tasks on which the model is trained, i.e., masked word and next sentence prediction. In both tasks, prediction accuracy is determined by the ability of the model to understand the context.

A transformer encoder computes the representation of each word through an attention mechanism with respect to the surrounding words. Given a sentence x_1^n of length n, the transformer computes the representation of each word x_i through a multi-head attention mechanism, where the query vector is from x_i and the key-value vector pairs are from the surrounding words $x_{i'}$ ($1 \leq i' \leq n$). The word representation produced by the transformer captures the contextual information of a word.

The attention mechanism can be viewed as mapping a query vector $\mathbf{q}$ and a set of key-value vector pairs $(\mathbf{k}, \mathbf{v})$ to an output vector. The attention function $A(\cdot)$ computes the output vector which is the weighted sum of the value vectors and is defined as:

$$A(\mathbf{q}, \mathbf{K}, \mathbf{V}, \rho) = \sum_{(\mathbf{k},\mathbf{v}) \in (\mathbf{K},\mathbf{V})} \alpha(\mathbf{q}, \mathbf{k}, \rho)\mathbf{v} \quad (1)$$

$$\alpha(\mathbf{q}, \mathbf{k}, \rho) = \frac{\exp(\rho \mathbf{k}^\top \mathbf{q})}{\sum_{\mathbf{k}' \in \mathbf{K}} \exp(\rho \mathbf{k}'^\top \mathbf{q})} \quad (2)$$

where $\mathbf{K}$ and $\mathbf{V}$ are matrices, containing the key vectors and the value vectors of the words in the sentence respectively, and $\alpha(\mathbf{q}, \mathbf{k}, \rho)$ is a scalar attention weight between $\mathbf{q}$ and $\mathbf{k}$, re-scaled by a scalar ρ.

Two building blocks for the transformer encoder are the multi-head attention mechanism and the position-wise feed-forward neural network (FFNN). The multi-head attention mechanism with H heads leverages the attention function in Equation 1 as follows:

$$\text{MH}(\mathbf{q}, \mathbf{K}, \mathbf{V}, \rho) = \mathbf{W}_{\text{MH}} \bigoplus_{\eta=1}^{H} \text{head}_\eta(\mathbf{q}, \mathbf{K}, \mathbf{V}, \rho) \quad (3)$$

$$\text{head}_\eta(\mathbf{q}, \mathbf{K}, \mathbf{V}, \rho) = A(\mathbf{W}_\eta^{\mathbf{Q}}\mathbf{q}, \mathbf{W}_\eta^{\mathbf{K}}\mathbf{K}, \mathbf{W}_\eta^{\mathbf{V}}\mathbf{V}, \rho) \quad (4)$$

where $\oplus$ denotes concatenation of vectors, $\mathbf{W}_{\text{MH}} \in \mathbb{R}^{d_{\text{model}} \times H d_{\mathbf{v}}}$, $\mathbf{W}_\eta^{\mathbf{Q}}, \mathbf{W}_\eta^{\mathbf{K}} \in \mathbb{R}^{d_{\mathbf{k}} \times d_{\text{model}}}$, and $\mathbf{W}_\eta^{\mathbf{V}} \in \mathbb{R}^{d_{\mathbf{v}} \times d_{\text{model}}}$. The input vector $\mathbf{q} \in \mathbb{R}^{d_{\text{model}}}$ is the hidden vector for the ambiguous word, while input matrices $\mathbf{K}, \mathbf{V} \in \mathbb{R}^{d_{\text{model}} \times n}$ are the concatenation of the hidden vectors of all words in the sentence. For each attention head, the dimension of both the query and key vectors is $d_{\mathbf{k}}$ while the dimension of the value vector is $d_{\mathbf{v}}$. The encoder model dimension is d_{model}.

The position-wise FFNN performs a non-linear transformation on the attention output corresponding to each input word position as follows:

$$\text{FF}(\mathbf{u}) = \mathbf{W}_{\text{FF2}}(\max(0, \mathbf{W}_{\text{FF1}}\mathbf{u} + \mathbf{b}_{\text{FF1}})) + \mathbf{b}_{\text{FF2}} \quad (5)$$

in which the input vector $\mathbf{u} \in \mathbb{R}^{d_{\text{model}}}$ is transformed to the output vector with dimension d_{model} via a series of linear projections with the ReLU activation function.

For the hidden layer l $(1 \leq l \leq L)$, the self-attention sub-layer output $\mathbf{f}_i^l$ is computed as follows:

$$\mathbf{f}_i^l = \text{LayerNorm}(\chi_{\mathbf{h},i}^l + \mathbf{h}_i^{l-1})$$
$$\chi_{\mathbf{h},i}^l = \text{MH}_{\mathbf{h}}^l(\mathbf{h}_i^{l-1}, \mathbf{h}_{1:n}^{l-1}, \mathbf{h}_{1:n}^{l-1}, \frac{1}{\sqrt{d_{\text{v}}}})$$

where LayerNorm refers to layer normalization (Ba et al., 2016) and the superscript l and subscript $\mathbf{h}$ indicate that each encoder layer l has an independent set of multi-head attention weight parameters (see Equations 3 and 4). The input for the first layer is $\mathbf{h}_i^0 = \mathbf{E}(x_i)$, which is the non-contextualized word embedding of x_i.

The second sub-layer consists of the position-wise fully connected FFNN, computed as:

$$\mathbf{h}_i^l = \text{LayerNorm}(\text{FF}_{\mathbf{h}}^l(\mathbf{f}_i^l) + \mathbf{f}_i^l)$$

where, similar to self-attention, an independent set of weight parameters in Equation 5 is defined in each layer.

4 Incorporating Pre-Trained Contextualized Word Representation

As BERT is trained on the masked word prediction task, which is to predict a word given the surrounding (left and right) context, the pre-trained model already captures the context. In this section, we describe different techniques of leveraging BERT for WSD, broadly categorized into nearest neighbor matching and linear projection of hidden layers.

4.1 Nearest Neighbor Matching

A straightforward way to disambiguate word sense is through 1-nearest neighbor matching. We compute the contextualized representation of each word in the training data and the test data through BERT. Given a hidden representation $\mathbf{h}_i^L$ at the L-th layer for word x_i in the test data, nearest neighbor matching finds a vector $\mathbf{h}^*$ in the L-th layer from the training data such that

$$\mathbf{h}^* = \arg\max_{\mathbf{h}'} \cos(\mathbf{h}_i^L, \mathbf{h}') \quad (6)$$

where the sense assigned to x_i is the sense of the word whose contextualized representation is $\mathbf{h}^*$. This WSD technique is adopted in earlier work on contextualized word representations (Melamud et al., 2016; Peters et al., 2018).

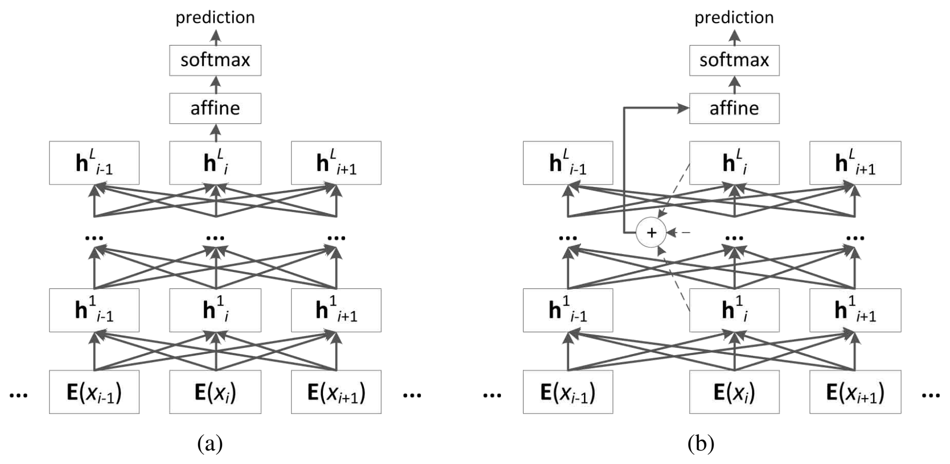

Figure 1: Illustration of WSD models by linear projection of (a) the last layer and (b) the weighted sum of all layers.

4.2 Linear Projection of Hidden Layers

Apart from nearest neighbor matching, we can perform a linear projection of the hidden vector $\mathbf{h}_i$ by an affine transformation into an output sense vector, with its dimension equal to the number of senses for word x_i. The output of this affine transformation is normalized by softmax such that all its values sum to 1. Therefore, the predicted sense $\mathbf{s}_i$ of word x_i is formulated as

$$\mathbf{s}_i = \text{softmax}(\mathbf{W}^{\text{lexelt}(x_i)}\mathbf{h}_i + \mathbf{b}^{\text{lexelt}(x_i)}) \quad (7)$$

where $\mathbf{s}_i$ is a vector of predicted sense distribution for word x_i, while $\mathbf{W}^{\text{lexelt}(x_i)}$ and $\mathbf{b}^{\text{lexelt}(x_i)}$ are respectively the projection matrix and bias vector specific to the lexical element (lexelt) of word x_i, which consists of its lemma and optionally its part-of-speech tag. We choose the sense corresponding to the element of $\mathbf{s}_i$ with the maximum value.

Training the linear projection model is done by the back-propagation algorithm, which updates the model parameters to minimize a cost function. Our cost function is the negative log-likelihood of the softmax output value that corresponds to the tagged sense in the training data. In addition, we propose two *novel* ways of incorporating BERT's hidden representation vectors to compute the sense output vector, which are described in the following sub-subsections.

4.2.1 Last Layer Projection

Similar to the nearest neighbor matching model, we can feed the hidden vector of BERT in the last layer, $\mathbf{h}_i^L$, into an affine transformation followed by softmax. That is, $\mathbf{h}_i$ in Equation 7 is instantiated by $\mathbf{h}_i^L$. The last layer projection model is illustrated in Figure 1(a).

4.2.2 Weighted Sum of Hidden Layers

BERT consists of multiple layers stacked one after another. Each layer carries a different representation of word context. Taking into account different hidden layers may help the WSD system to learn from different context information encoded in different layers of BERT.

To take into account all layers, we compute the weighted sum of all hidden layers, $\mathbf{h}_i^l$, where $1 \leq l \leq L$, corresponding to a word position i, through attention mechanism. That is, $\mathbf{h}_i$ in Equation 7 is replaced by the following equation:

$$\mathbf{h}_i = A(\mathbf{m}, \mathbf{W}^{\mathbf{s}}\mathbf{h}_i^{1:L}, \mathbf{h}_i^{1:L}, 1) \quad (8)$$

where $\mathbf{m} \in \mathbb{R}^{d_{\text{model}}}$ is a projection vector to obtain scalar values with the key vectors. The model with weighted sum of all hidden layers is illustrated in Figure 1(b).

4.2.3 Gated Linear Unit

While the contextualized representations in the BERT hidden layer vectors are features that determine the word sense, some features are more useful than the others. As such, we propose filtering the vector values by a gating vector whose values range from 0 to 1. This mechanism is known as the gated linear unit (GLU) (Dauphin et al., 2017), which is formulated as

$$\text{GLU}(\mathbf{h}) = (\mathbf{h}+\mathbf{W}^{\mathbf{h}}\mathbf{h}+\mathbf{b}^{\mathbf{h}})\odot\sigma(\mathbf{W}^{\mathbf{g}}\mathbf{h}+\mathbf{b}^{\mathbf{g}}) \quad (9)$$

where $\mathbf{W}^{\mathbf{h}}$ and $\mathbf{W}^{\mathbf{g}}$ are separate projection matrices and $\mathbf{b}^{\mathbf{h}}$ and $\mathbf{b}^{\mathbf{g}}$ are separate bias vectors. The symbols $\sigma(\cdot)$ and $\odot$ denote the sigmoid function and element-wise vector multiplication operation respectively.

GLU transforms the input vector $\mathbf{h}$ by feeding it to two separate affine transformations. The second transformation is used as the sigmoid gate to filter the input vector, which is summed with the vector after the first affine transformation.

5 Experimental Setup

We conduct experiments on various WSD tasks. The description and the statistics for each task are given in Table 1. For English, a lexical element (lexelt) is defined as a combination of lemma and part-of-speech tag, while for Chinese, it is simply the lemma, following the OntoNotes setup.

We exploit English BERT$_{\text{BASE}}$ for the English tasks and Chinese BERT for the Chinese task. We conduct experiments with different strategies of incorporating BERT as described in Section 4, namely 1-nearest neighbor matching (**1-nn**) and linear projection. In the latter technique, we explore strategies including **simple** last layer projection, layer weighting (**LW**), and gated linear unit (**GLU**).

In the linear projection model, we train the model by the Adam algorithm (Kingma and Ba, 2015) with a learning rate of 10^{-3}. The model parameters are updated per mini-batch of 16 sentences. As update progresses, we pick the best model parameters from a series of neural network updates based on accuracy on a held-out development set, disjoint from the training set.

The state-of-the-art supervised WSD approach takes into account features from the neighboring sentences, typically one sentence to the left and one to the right apart from the current sentence that contains the ambiguous words. We also exploit this in our model, as BERT supports inputs with multiple sentences separated by a special `[SEP]` symbol.

For English all-words WSD, we train our WSD model on SemCor (Miller et al., 1994), and test it on Senseval-2 (SE2), Senseval-3 (SE3), SemEval 2013 task 12 (SE13), and SemEval 2015 task 13 (SE15). This common benchmark, which has been annotated with WordNet-3.0 senses (Raganato et al., 2017a), has recently been adopted in English all-words WSD. Following (Raganato et al., 2017b), we choose SemEval 2007 Task 17 (SE07) as our development data to pick the best model parameters after a number of neural network updates, for models that require back-propagation training.

Task	# Instances	# Lexelts
English all-words		
- SemCor (train)	226,036	22,436
- SemEval-2007 (dev)	455	330
- Senseval-2 (test)	2,282	1,093
- Senseval-3 (test)	1,850	977
- SemEval 2013 (test)	1,664	751
- SemEval 2015 (test)	1,022	512
English lexical sample		
- Senseval-2 (train)	8,611	177
- Senseval-2 (test)	4,328	146
- Senseval-3 (train)	8,022	57
- Senseval-3 (test)	3,944	57
Chinese OntoNotes		
- Train	66,409	431
- Dev	9,523	341
- BC (test)	1,769	160
- BN (test)	3,227	253
- MZ (test)	1,876	223
- NW (test)	1,483	143
- All (test)	8,355	324

Table 1: Statistics of the datasets used for the English all-words task, English lexical sample task, and Chinese OntoNotes WSD task in terms of the number of instances and the number of distinct lexelts. For Chinese WSD task, "All" refers to the concatenation of all genres BC, BN, MZ, and NW.

We also evaluate on Senseval-2 and Senseval-3 English lexical sample tasks, which come with pre-defined training and test data. For each word type, we pick 20% of the training instances to be our development set and keep the remaining 80% as the actual training data. Through this development set, we determine the number of epochs to use in training. We then re-train the model with the whole training dataset using the number of epochs identified in the initial training step.

While WSD is predominantly evaluated on English, we are also interested in evaluating our approach on Chinese, to evaluate the effectiveness of our approach in a different language. We use OntoNotes Release 5.0[1], which contains a number of annotations including word senses for Chinese. We follow the data setup of Pradhan et al. (2013) and conduct an evaluation on four genres, i.e., broadcast conversation (BC), broadcast news (BN), magazine (MZ), and newswire (NW), as well as the concatenation of all genres. While the training and development datasets are divided into genres, we train on the concatenation of all genres and test on each individual genre.

[1]LDC2013T19

System	SE07	SE2	SE3	SE13	SE15	Avg
Reported in previous papers						
MFS baseline	54.5	65.6	66.0	63.8	67.1	65.6
IMS (Zhong and Ng, 2010)	61.3	70.9	69.3	65.3	69.5	68.8
IMS+emb (Iacobacci et al., 2016)	60.9	71.0	69.3	67.3	71.3	69.7
SupWSD (Papandrea et al., 2017)	60.2	71.3	68.8	65.8	70.0	69.0
SupWSD+emb (Papandrea et al., 2017)	63.1	72.7	70.6	66.8	71.8	70.5
BiLSTMatt+LEX (Raganato et al., 2017b)	63.7	72.0	69.4	66.4	72.4	70.1
GASext Concat (Luo et al., 2018)	–	72.2	70.5	67.2	72.6	70.6
context2vec (Melamud et al., 2016)	61.3	71.8	69.1	65.6	71.9	69.6
ELMo (Peters et al., 2018)	62.2	71.6	69.6	66.2	71.3	69.7
BERT nearest neighbor (ours)						
1nn (1sent)	64.0	73.0	69.7	67.8	73.3	71.0
1nn (1sent+1sur)	63.3	73.8	71.6	69.2	74.4	72.3
BERT linear projection (ours)						
Simple (1sent)	67.0	75.0	71.6	69.7	74.4	72.7
Simple (1sent+1sur)	**69.3***	75.9*	73.4	70.4*	75.1	73.7*
LW (1sent)	66.7	75.0	71.6	69.9	74.2	72.7
LW (1sent+1sur)	69.0*	**76.4***	**74.0***	70.1*	75.0	73.9*
GLU (1sent)	64.9	74.1	71.6	69.8	74.3	72.5
GLU (1sent+1sur)	68.1*	75.5*	73.6*	**71.1***	**76.2***	**74.1***
GLU+LW (1sent)	65.7	74.0	70.9	68.8	73.6	71.8
GLU+LW (1sent+1sur)	68.5*	75.5*	73.4*	71.0*	**76.2***	74.0*

Table 2: English all-words task results in F1 measure (%), averaged over three runs. SemEval 2007 Task 17 (SE07) test set is used as the development set. We show the results of nearest neighbor matching (**1nn**) and linear projection, by **simple** last layer linear projection, layer weighting (**LW**), and gated linear units (**GLU**). Apart from BERT representation of one sentence (*1sent*), we also show BERT representation of one sentence plus one surrounding sentence to the left and one to the right (*1sent+1sur*). The best result in each dataset is shown in bold. Statistical significance tests by bootstrap resampling ($*$: $p < 0.05$) compare 1nn (1sent+1sur) with each of Simple (1sent+1sur), LW (1sent+1sur), GLU (1sent+1sur), and GLU+LW (1sent+1sur).

For Chinese WSD evaluation, we train IMS (Zhong and Ng, 2010) on the Chinese OntoNotes dataset as our baseline. We also incorporate pre-trained non-contextualized Chinese word embeddings as IMS features (Taghipour and Ng, 2015; Iacobacci et al., 2016). The pre-trained word embeddings are obtained by training the *word2vec* skip-gram model on Chinese Gigaword Fifth Edition[2], which after automatic word segmentation contains approximately 2 billion words. Following (Taghipour and Ng, 2015), we incorporate the embedding features of words within a window surrounding the target ambiguous word. In our experiments, we take into account 5 words to the left and right.

[2]LDC2011T13

6 Results

We present our experimental results and compare them with prior baselines.

6.1 English All-Words Tasks

For English all-words WSD, we compare our approach with three categories of prior approaches. Firstly, we compare our approach with the supervised SVM classifier approach, namely IMS (Zhong and Ng, 2010). We compare our approach with both the original IMS without word embedding features and IMS with non-contextualized word embedding features, that is, *word2vec* with exponential decay (Iacobacci et al., 2016). We also compare with SupWSD (Papandrea et al., 2017), which is an alternative implementation of IMS. Secondly, we compare our approach with the neural WSD approaches that leverage bidirectional LSTM (bi-LSTM). These include the bi-LSTM model with attention trained jointly with lexical semantic labeling task (Raganato et al., 2017b) (BiLSTMatt+LEX) and the bi-LSTM model enhanced with gloss representation from WordNet (GAS). Thirdly, we show comparison with prior contextualized word representations for WSD, pre-trained on a large number of texts, namely *context2vec* (Melamud et al., 2016) and ELMo (Peters et al., 2018). In these two models, WSD is treated as nearest neighbor matching as described in Section 4.1.

Table 2 shows our WSD results in F1 measure. It is shown in the table that with the nearest neighbor matching model, BERT outperforms *context2vec* and ELMo. This shows the effectiveness of BERT's pre-trained contextualized word representation. When we include surrounding sentences, one to the left and one to the right, we get improved F1 scores consistently.

We also show that linear projection to the sense output vector further improves WSD performance, by which our best results are achieved. While BERT has been shown to outperform other pre-trained contextualized word representations through the nearest neighbor matching experiments, its potential can be maximized through linear projection to the sense output vector. It is worthwhile to note that our more advanced linear projection, by means of layer weighting (§4.2.2 and gated linear unit (§4.2.3) gives the best F1 scores on all test sets.

All our BERT WSD systems outperform gloss-enhanced neural WSD, which has the best overall score among all prior systems.

6.2 English Lexical Sample Tasks

For English lexical sample tasks, we compare our approach with the original IMS (Zhong and Ng, 2010) and IMS with non-contextualized word embedding features. The embedding features incorporated into IMS include CW embeddings (Collobert et al., 2011), obtained from a convolutional language model, fine-tuned (adapted) to WSD (Taghipour and Ng, 2015) (*+adapted CW*), and *word2vec* skip-gram (Mikolov et al., 2013) with exponential decay (Iacobacci et al., 2016) (*+w2v+expdecay*). We also compare our approach with the bi-LSTM, on top of which sense classification is defined (Kågebäck and Salomonsson, 2016), and *context2vec* (Melamud et al., 2016), which is a contextualized pre-trained bi-LSTM model trained on 2B words of text. Finally, we also compare with a prior multi-task and semi-supervised WSD approach learned through alternating structure optimization (ASO) (Ando, 2006), which also utilizes unlabeled data for training.

System	SE2	SE3
Reported		
IMS	65.2	72.3
IMS+adapted CW	66.2	73.4
IMS+w2v+expdecay	69.9	75.2
BiLSTM	66.9	73.4
context2vec	–	72.8
ASO+multitask+semisup	–	74.1
BERT nearest neighbor (ours)		
1nn (1sent)	67.7	72.7
1nn (1sent+1sur)	71.5	75.7
BERT linear projection (ours)		
Simple (1sent)	73.3	78.6
Simple (1sent+1sur)	**76.9***	**80.0***
LW (1sent+1sur)	76.7*	**80.0***
GLU (1sent+1sur)	75.1*	79.4*
GLU+LW (1sent+1sur)	74.2*	79.8*

Table 3: English lexical sample task results in accuracy (%), averaged over three runs. Best accuracy in each dataset is shown in bold. Statistical significance tests by bootstrap resampling ($*$: $p < 0.05$) compare 1nn (1sent+1sur) with each of Simple (1sent+1sur), LW (1sent+1sur), GLU (1sent+1sur), and GLU+LW (1sent+1sur).

As shown in Table 3, our BERT-based WSD approach with linear projection model outperforms all prior approaches. *context2vec*, which is pre-trained on a large amount of texts, performs worse than the prior semi-supervised ASO approach on Senseval-3, while our best result outperforms ASO by a large margin.

Neural bi-LSTM performs worse than IMS with non-contextualized word embedding features. Our neural model with pre-trained contextualized word representations outperforms the best result achieved by IMS on both Senseval-2 and Senseval-3.

6.3 Chinese OntoNotes WSD

We compare our approach with IMS without and with word embedding features as the baselines. The results are shown in Table 4.

Across all genres, BERT outperforms the baseline IMS with word embedding (non-contextualized word representation) features (Taghipour and Ng, 2015). The latter also improves over the original IMS without word embedding features (Zhong and Ng, 2010). Linear

System	BC	BN	MZ	NW	All
Baseline					
IMS	79.8	85.7	83.0	91.0	84.8
IMS+w2v	80.2	86.4	82.3	92.0	85.2
BERT					
1nn	81.7	88.5	85.1	93.3	87.1
Simple	84.6*	**90.4***	**88.9***	93.9	**89.5***
LW	**84.7***	90.3*	88.8*	**94.0**	**89.5***
GLU	84.6*	90.0*	88.3*	93.3	89.0*
GLU+LW	84.6*	90.2*	88.2*	93.4	89.2*

Table 4: Chinese OntoNotes WSD results in accuracy (%), averaged over three runs, for each genre. All BERT results in this table are obtained from the representation of one sentence plus one surrounding sentence to the left and to the right (*1sent+1sur*). We show results of various BERT incorporation strategy, namely nearest neighbor matching (**1nn**), **simple** projection, projection with layer weighting (**LW**) and gated linear unit (**GLU**). Best accuracy in each genre is shown in bold. Statistical significance tests by bootstrap resampling ($*$: $p < 0.05$) compare 1nn with each of Simple, LW, GLU, and GLU+LW.

projection approaches consistently outperform nearest neighbor matching by a significant margin, similar to the results on English WSD tasks.

The best overall result for the Chinese OntoNotes test set is achieved by the models with simple projection and hidden layer weighting.

7 Discussion

Across all tasks (English all-words, English lexical sample, and Chinese OntoNotes), our experiments demonstrate the effectiveness of BERT over various prior WSD approaches. The best results are consistently obtained by linear projection models, which project the last hidden layer or the weighted sum of all hidden layers to an output sense vector.

We can view the BERT hidden layer outputs as contextual features, which serve as useful cues in determining the word senses. In fact, the attention mechanism in transformer captures the surrounding words. In prior work like IMS (Zhong and Ng, 2010), these contextual cues are captured by the manually-defined surrounding word and collocation features. The features obtained by the hidden vector output are shown to be more effective than the manually-defined features.

We introduced two advanced linear projection techniques, namely layer weighting and gated linear unit (GLU). While Peters et al. (2018) showed that the second biLSTM layer results in better WSD accuracy compared to the first layer (nearer to the individual word representation), we showed that taking into account different layers by means of the attention mechanism is useful for WSD. GLU as an activation function has been shown to be effective for better convergence and to overcome the vanishing gradient problem in the convolutional language model (Dauphin et al., 2017). In addition, the GLU gate vector, with values ranging from 0 to 1, can be seen as a filter for the features from the hidden layer vector.

Compared with two prior contextualized word representations models, *context2vec* (Melamud et al., 2016) and ELMo (Peters et al., 2018), BERT achieves higher accuracy. This shows the effectiveness of the attention mechanism used in the transformer model to represent the context.

Our BERT WSD models outperform prior neural WSD models by a large margin. These prior neural WSD models perform comparably with IMS with embeddings as classifier features, in addition to the discrete features. While neural WSD approaches (Kågebäck and Salomonsson, 2016; Raganato et al., 2017b; Luo et al., 2018) exploit non-contextualized word embeddings which are trained on large texts, the hidden layers are trained only on a small amount of labeled data.

8 Conclusion

For the WSD task, we have proposed novel strategies of incorporating BERT, a pre-trained contextualized word representation which effectively captures the context in its hidden vectors. Our experiments show that linear projection of the hidden vectors, coupled with gating to filter the values, gives better results than the prior state of the art. Compared to prior neural and feature-based WSD approaches that make use of non-contextualized word representations, using pre-trained contextualized word representation with our proposed incorporation strategy achieves significantly higher scores.

References

Rie Kubota Ando. 2006. Applying alternating structure optimization to word sense disambiguation. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 77–84.

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*.

Yee Seng Chan, Hwee Tou Ng, and David Chiang. 2007a. Word sense disambiguation improves statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 33–40.

Yee Seng Chan, Hwee Tou Ng, and Zhi Zhong. 2007b. NUS-PT: Exploiting parallel texts for word sense disambiguation in the English all-words tasks. In *Proceedings of the Fourth International Workshop on Semantic Evaluations*, pages 253–256.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the Thirty-Fourth International Conference on Machine Learning*, pages 933–941.

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*, pages 4171–4186.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

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*, pages 897–907.

Mikael Kågebäck and Hans Salomonsson. 2016. Word sense disambiguation using a bidirectional LSTM. In *Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon*, pages 51–56.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.

Yoong Keok Lee, Hwee Tou Ng, and Tee Kiah Chia. 2004. Supervised word sense disambiguation with support vector machines and multiple knowledge sources. In *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 137–140.

Fuli Luo, Tianyu Liu, Qiaolin Xia, Baobao Chang, and Zhifang Sui. 2018. Incorporating glosses into neural word sense disambiguation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 2473–2482.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics,*, pages 1064–1074.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Proceedings of the Thirty-First Conference on Neural Information Processing Systems*, pages 6297–6308.

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.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at the International Conference on Learning Representations*.

George A. Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G. Thomas. 1994. Using a semantic concordance for sense identification. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey*, pages 240–243.

Simone Papandrea, Alessandro Raganato, and Claudio Delli Bovi. 2017. SupWSD: A flexible toolkit for supervised word sense disambiguation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 103–108.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2227–2237.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017a. 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*, pages 99–110.

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017b. Neural sequence learning models for word sense disambiguation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1156–1167.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the Twenty-Eighth Conference on Neural Information Processing Systems*, pages 3104–3112.

Kaveh Taghipour and Hwee Tou Ng. 2015. Semi-supervised word sense disambiguation using word embeddings in general and specific domains. In *Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL*, pages 314–323.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Thirty-First Conference on Neural Information Processing Systems*, pages 5998–6008.

Zhi Zhong and Hwee Tou Ng. 2010. It Makes Sense: a wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 System Demonstrations*, pages 78–83.

Zhi Zhong and Hwee Tou Ng. 2012. Word sense disambiguation improves information retrieval. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 273–282.