Quasi Bidirectional Encoder Representations from Transformers for Word Sense Disambiguation

Michele Bevilacqua and **Roberto Navigli**
Department of Computer Science
Sapienza University of Rome
{bevilacqua,navigli}@di.uniroma1.it

Abstract

While contextualized embeddings have produced performance breakthroughs in many Natural Language Processing (NLP) tasks, Word Sense Disambiguation (WSD) has not benefited from them yet. In this paper, we introduce QBERT, a Transformer-based architecture for contextualized embeddings which makes use of a co-attentive layer to produce more deeply bidirectional representations, better-fitting for the WSD task. As a result, we are able to train a WSD system that beats the state of the art on the concatenation of all evaluation datasets by over 3 points, also outperforming a comparable model using ELMo.

1 Introduction

Word Sense Disambiguation (WSD) is the task of associating a word in context with the right meaning among a finite set of possible senses (Navigli, 2009). Consider the following sentence, in which SERVED is the target word:

(1) The waiter standing near the counter SERVED the revolutionary cause well.

In WordNet (Fellbaum, 1998), the most used English computational lexicon in NLP, the following two senses are associated (among many others) to the verb *to serve*:

1. ***devotion****: devote (part of) one's life or efforts to, as of countries, institutions, or ideas*;
2. ***food****: help to some food; help with some food or drink*;

The WSD system, in this case, would be tasked to associate the target word with the correct meaning – i.e. the *devotion* sense.

Currently the best WSD systems are supervised, i.e. they leverage annotated corpora as training data (Yuan et al., 2016; Vial et al., 2018; Melacci et al., 2018). However, data labeling is a bottleneck for WSD, even more so than in other fields of NLP. Semantic annotation is a costly process, requiring expert annotators (Taghipour and Ng, 2015; Pasini and Navigli, 2017). If we consider that neural networks, the best performing approach in virtually every task in NLP, are particularly data-hungry, it appears unlikely that there will be much progress in WSD unless either more data is available, or less data is needed.

Between the two directions, we believe efforts towards the latter will prove more fruitful, firstly, because of scalability considerations, and secondly, and more importantly, because of the recent growth in the use of transfer learning, as exemplified by contextualized embeddings. Contextualized embeddings have been shown to produce much better results on downstream tasks compared to end-to-end training, even when less data is provided (Peters et al., 2018; Howard and Ruder, 2018; Devlin et al., 2019; He et al., 2018; Akbik et al., 2018). Contextualized embeddings that use words as tokenization units, such as ELMo (Peters et al., 2018), are most suited to WSD. They are usually trained through self-supervised Causal Language Modeling (CLM) (Lample and Conneau, 2019): given a word sequence $w_1, w_2, \ldots, w_n$ the system has to use w_1 to predict w_2, the sequence $w_{1:2}$ to predict w_3 and so on. CLM is inherently unidirectional, as the model must not be able to "peek" at the word it has to predict. Thus to encode the left and the right contexts two separate networks have to be used, even if they often share part of the weights and are jointly trained.

As regards the use of contextualized embeddings in WSD, this is bound to pose a problem.

Proceedings of Recent Advances in Natural Language Processing, pages 122–131,
Varna, Bulgaria, Sep 4–6 2017.
https://doi.org/10.26615/978-954-452-049-6_015

Consider the sentence (1) above. It features attractors, i.e. words or phrases pushing the sense interpretation in one direction or the other, with the left context providing a strong cue for the *food* sense and the right for the *devotion* sense. In this paper, we propose a modification of the usual CLM architecture for transfer learning that enables us to train a high-performance WSD system. In this context, we make the following contributions:

- we introduce the BiTransformer, a novel Transformer-based (Vaswani et al., 2017) coattentive layer allowing deeper bidirectionality;
- we introduce QBERT (Quasi Bidirectional Encoder Representations from Transformers), a novel Transformer-based architecture for CLM making use of the BiTransformer;
- we train a WSD model using QBERT contextualized embeddings, outperforming on the standard evaluation datasets both the previously established state of the art (by a large margin) and a comparable model using ELMo;
- we use QBERT to beat ELMo on the recently established Word-in-Context (WiC) task (Pilehvar and Camacho-Collados, 2019).

2 Related Work

Despite the limited availability of training data, the WSD systems offering the best performances are supervised ones. Many of the approaches are still end-to-end, i.e. they only make use of the information learned during the WSD training.

End-to-end WSD Systems In WSD traditional machine learning techniques are still very competitive because they are not as data-hungry as neural networks. The very popular It Makes Sense (IMS) system (Zhong and Ng, 2010), based on Support Vector Machines and hand-crafted features, performs very well when word embeddings are used as additional features (Iacobacci et al., 2016); the classifier by Papandrea et al. (2017) also gets competitive results. The system of Weissenborn et al. (2015) attains very high performances, but only disambiguates nouns. More recently, neural models have been developed (Kagebäck and Salomonsson, 2016; Uslu et al., 2018; Luo et al., 2018). Some of the most successful offer an intuitive framing of WSD as a tagging task (Raganato et al., 2017a; Vial et al., 2018).

Transfer Learning WSD Systems One of the best performing WSD systems (Yuan et al., 2016) employs a semi-supervised neural architecture, whereby a unidirectional LSTM was trained to predict a masked token on huge amounts of unlabeled data (over 100B tokens). The trained LSTM was used to produce contextualized embeddings for tagged tokens in SemCor; then kNN or a more sophisticated label propagation algorithm was used to predict a sense. The size of the training data makes replication difficult – a reimplementation attempt with a smaller corpus led to worse results (Le et al., 2018). A similar approach using ELMo contextualized embeddings has been presented by Peters et al. (2018), but the results were underwhelming. Another attempt at using transfer learning in WSD has been carried out by Melacci et al. (2018). The authors enhanced IMS with context2vec (Melamud et al., 2016), obtaining performance roughly on a par with Yuan et al. (2016).

Contextualized Embeddings Most of the approaches to contextualized embeddings involve CLM pretraining of directional (either attentive or recurrent) networks. Very successful CLM-based models include ELMo, in which two separate directional LSTMs are fed the output of a shared character-based Convolutional Neural Network (CNN) encoder (Peters et al., 2018), and OpenAI GPT, using Transformers instead of LSTMs and a BPE vocabulary (Sennrich et al., 2016) with regular embeddings instead of the CNN encoder (Radford et al., 2018). Another popular approach, Flair, features character-level LSTMs, outputting hidden states at word boundaries (Akbik et al., 2018). As CLM architectures are normally unidirectional, one alternative in order to guarantee a joint encoding of the context is the Masked Language Modeling (MLM) of BERT (Devlin et al., 2019), which, however, requires a variety of tricks at training time.

3 The QBERT Architecture

Similarly to other LM-based approaches to contextualized embeddings (Peters et al., 2018; Radford et al., 2018; Howard and Ruder, 2018; Devlin et al., 2019), the architecture we hereby propose has two main components, which we will refer to

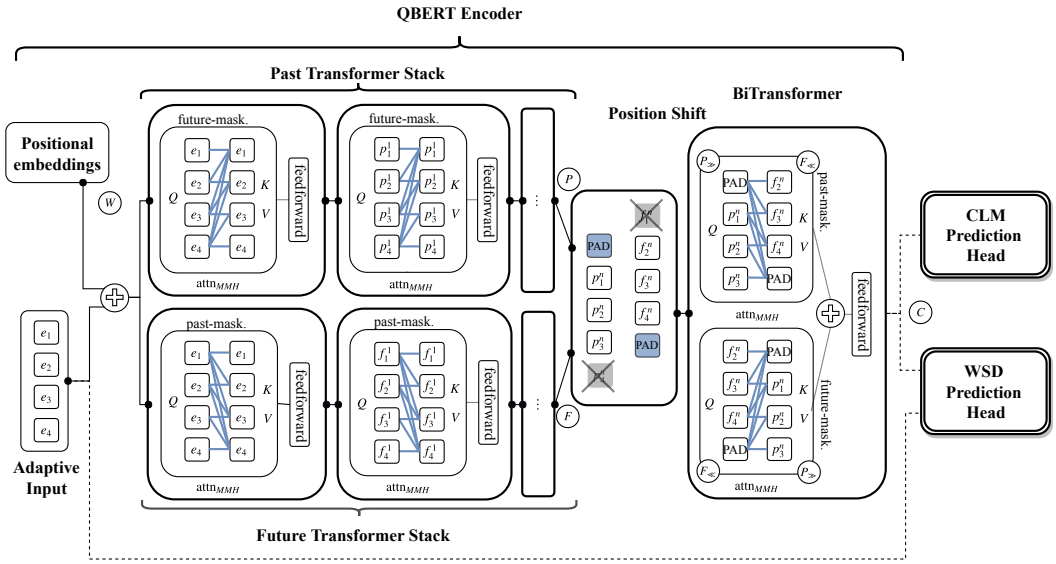

Figure 1: A high-level view of the QBERT architecture.

as Encoder and task-specific Prediction Head. In Figure 1 we show a high-level view of our system. Raw tokens are fed to the encoder, which embeds them into context-independent fixed-length vector representations (the *word embeddings*, W in Figure 1), then uses them to produce context-dependent hidden representations (the *contextualized embeddings*, C), where the context is some subset of the sequence itself. The Prediction Head exploits the vectors produced by the Encoder to perform a task.

3.1 Encoder

As will become clear in what follows, the Encoder of the QBERT architecture is able to compute the hidden representation of a word w_t in a sequence $w_{1:n}$ as a function of the weights and of the whole sequence except w_t itself, i.e. of $w_{1:t-1}$ and $w_{t+1:n}$. To embed tokens, the Encoder uses the Adaptive Input layer (Baevski and Auli, 2018). Sinusoidal positional embeddings are added to the output and passed to two separate stacks of masked Transformers, computing two directional encodings of the sequence, with one (P) having past and present ($w_{1:t}$) information encoded in the present-token hidden vector and the other (F) having present and future ($w_{t:n}$) information instead. Since in the CLM training information about the present token must be hidden from the output layer, we shift and pad the sequences in order to have only the past tokens encoded in the output of the first stack ($P_{\gg}$) and only future tokens encoded in the output of the second ($F_{\ll}$). To combine the shifted sequences we use a novel Transformer layer variant taking them both as input, the BiTransformer, featuring a co-attentive mechanism in which $P_{\gg}$ attends over $F_{\ll}$ and $F_{\ll}$ attends over $P_{\gg}$. The Encoder is trained on CLM using an Adaptive Softmax layer (Grave et al., 2017) as Prediction Head.

3.2 Transformer Variants in QBERT

In the QBERT Encoder we employ three distinct variants of the plain Transformer: the future-masked Transformer, the past-masked Transformer and the BiTransformer. To introduce them we first need to elaborate further into the inner workings of the layer. A vanilla Transformer layer (Vaswani et al., 2017) can be defined as a multi-head self attention submodule followed by a time-wise 2-layer feedforward network, with additional residual connection (He et al., 2016) and layer normalization stabilizing training (Ba et al., 2016).

Core (Self) Attention The intuition behind the attention mechanism is very simple (Bahdanau et al., 2015; Luong et al., 2015). We have a sequence of vectors (the n_q queries Q of dimension d_q) and we want to compute relevance scores against some other sequence of vectors (the n_k keys K of dimension d_k) specific to each couple of vectors q and k. The $n_q \times n_k$ relevance score matrix is then used to compute n_q weighted means of another sequence of vectors (the n_k values V of dimension d_v). So, if we pack Q, K, V into matrices, the mechanism can be distilled into the

formula:

$$\text{attn}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

where $\sqrt{d_k}$ is a normalization factor meant to prevent the dot products from getting too large. In the case of the *self* attention mechanism Q, K, V stand for the same matrix. In the Transformer (Vaswani et al., 2017), multi-head attention is used, in which n attentions (the heads) are computed in parallel, concatenated and then combined through dot product with a $d_q n \times d_o$ matrix W^o. For each attention head h_i, there are three weight matrices W_i^Q, W_i^K, W_i^V, multiplying respectively Q, K and V. Formally, we define multi-head attention (attn_{MH}) as:

$$\text{attn}_{\text{MH}}(Q, K, V) = \bigoplus_{i=0}^{n} [\text{softmax}(\frac{(QW_i^Q)(KW_i^K)^T}{\sqrt{d_k}})(VW_i^Q)]W^o \quad (2)$$

where $\oplus$ denotes concatenation along the second dimension.

Transformer Masking In the past and future stacks, as well as in the BiTransfomer layer, we employ a masking mechanism on attention to enforce directionality, i.e. to force the relevance scores computed between Q and K to be 0 for tokens following or preceding the current one, as needed. Masking can be implemented by performing an elementwise sum between QK^T and a $n_q \times n_k$ masking matrix M, whose values are set to $-\infty$ if K_i and V_j are to be excluded from the attention computation, else 0. In our architecture we employ two different masking matrices: a future masking-matrix M_f which is set to $-\infty$ when $i < j$ and a past-masking matrix M_p set to $-\infty$ when $i > j$; note that $M_f = M_p^T$. Multihead and masked attention can be combined by simply using the same masking matrix in each attention head. We use M_f in the past Transformer stack and M_p in the future stack, producing, respectively, P and F. To encourage the network to encode comparable representations we tie the weights between layers at the same depth on the past and future stack.

3.2.1 Timestep Shift

Present-token information is still encoded in both P and F. To remove it, we use a simple shifting approach where we detach the nth timestep from P and the first from F and add padding to the opposite sides. This effectively shifts the hidden representation by one place to the left and by one place to the right. We refer to the resulting sequences as, respectively, $P_{\gg}$ and $F_{\ll}$. As a result, the ith position of $P_{\gg}$ encodes information from tokens $w_{1:i-1}$ while $F_{\ll}$ encodes $w_{i+1:n}$. Formally:

$$\begin{aligned} P_{\gg} &= PAD \oplus P_{1:n-1} \\ F_{\ll} &= F_{2:n} \oplus PAD \end{aligned} \quad (3)$$

where $\oplus$ denotes concatenation along the timestep dimension. The padding vector PAD is learned during training. The process is visualized in Figure 1, where tokens are aligned according to their shifted positions.

3.2.2 BiTransformer

To combine $P_{\gg}$ and $F_{\ll}$ we employ the BiTransformer, a novel Transformer layer variant that uses a masked coattentive multihead attention mechanism over two sequences. Masking allows $P_{\gg}$ to attend over $F_{\ll}$ while keeping present-token knowledge hidden from the network, and vice versa. This allows deeper bidirectionality in that the resulting output is not a naive combination of two separate directional representations but rather the result of a whole-sequence attention, albeit computed in a two-step process, where the first step can be arbitrarily deep (the masked Transformer stacks) and the second is always shallow (the BiTransformer). Unfortunately, BiTransformers cannot be stacked as each timestep in the output of the layer encodes information about every token in the sequence but the one it has to predict in CLM, so any further use of attention would make pretraining impossible.

The BiTransformer requires modifications to the first part of the vanilla Transformer intramodule architecture. First, both input sequences are layer normalized separately. We compute a masked multihead attention using the future-masked sequence $P_{\gg}$ as Q, the past-masked sequence $F_{\ll}$ as K and V, using the past-masking matrix M_p. To give an insight into what happens, the position i of the n queries, encoding information about words 1 to $i-1$, is allowed to look at positions i to n of the keys, encoding words $w_{i+1:n}$, $w_{i+2:n}$ and so on. Then we compute the reverse, using $F_{\ll}$, $P_{\gg}$ and future-masking matrix M_f. This process results in two sequences to which input residuals are added, and then added

together via a simple elementwise sum. The rest of the layer works just like a regular Transformer layer. We formally describe this coattentive mechanism as follows:

$$\begin{aligned} P'_{\gg} &= \text{LayerNorm}(P_{\gg}) \\ F'_{\ll} &= \text{LayerNorm}(F_{\ll}) \\ O &= \text{attn}_{MMH}(P'_{\gg}, F'_{\ll}, F'_{\ll}, M_p) + \\ &\quad \text{attn}_{MMH}(F'_{\ll}, P'_{\gg}, P'_{\gg}, M_f) + P_{\gg} + F_{\ll} \end{aligned} \tag{4}$$

O goes through the 2-layer feedforward to produce contextualized embeddings, which are used as input for the task-specific Prediction Heads. We describe them in the relevant paragraphs of Sections 4.1 and 5.

4 Experimental Setup

In what follows we first describe the Encoder architecture hyperparameters and CLM pretraining details (Section 4.1). In Section 4.2 we describe the contextualized embeddings systems we use as comparison in the WSD and Word-in-Context experiments. Finally, we report the setup and results of the experiments in Section 5.

4.1 QBERT Encoder Pretraining

CLM Prediction Head and Hyperparameters To train the QBERT Encoder on CLM we use an Adaptive Softmax (Grave et al., 2017) layer as Prediction Head. Following Baevski and Auli (2018), we tie the weights (Press and Wolf, 2017) of the embedding matrices but not the projective weights. Both Adaptive Input and Adaptive Softmax use a vocabulary of 400k words, with cutoffs set to 35k, 100k, 200k and a shrinking factor of 4. The past and future stacks as well as the BiTransformer feature an input and output size of 512, while the first layer of the internal feedforward projects the input to 2048 dimensions, the same as the base configuration in Vaswani et al. (2017). The masked Transformer stacks are both 5-layer deep.

Training Hyperparameters We train QBERT on the English UMBC corpus (Han et al., 2013), which contains around 3B tokens. In our training loop we feed the input in batches of 5000 tokens, splitting the corpus in sequences of max 100 tokens. We found it beneficial to accumulate the gradient for many training steps, performing an update every 16 batches, resulting in a virtual batch size of 80000 tokens. As an optimizer we employ regular Nesterov-accelerated SGD, with a learning rate that first increases linearly from 10^{-5} to 1 during a warmup phase lasting 2000 updates, and then varies from a maximum of 1 to a minimum of 10^{-5} according to a Cyclical Learning Rate (Smith, 2017) policy with cosine scheme, with a period of 2000 updates. With each cycle, the period is multiplied by 1.5 while both the maximum and minimum values are halved. We train until convergence.

We implement the system and training logic in `pytorch` with the help of the `fairseq` library.

4.2 Comparison Systems

In our experiments we compare QBERT with three different contextualized embeddings systems:

1. Off-the-shelf pretrained **ELMo** (Peters et al., 2018). We employ a model featuring 4096-sized bidirectional LSTMs and 512-sized contextualized embeddings[1], pretrained on the concatenation of a Wikipedia dump and a few English monolingual news corpora[2], for a total of 5.5B tokens.

2. Off-the-shelf pretrained **Flair** (Akbik et al., 2018). We employ the models the project's page[3] refers to as `mix-forward` and `mix-backward`, pretrained on "Web, Wikipedia, Subtitles"[4]. We concatenate their contextualized embeddings.

3. **SBERT** (Shallowly Bidirectional Encoder Representations from Transformers), a baseline featuring the same architecture as QBERT but missing the BiTransformer layer: the outputs of the past and future stacks are simply combined through elementwise sum after the position shift.

We do not include in our comparison the BPE-based systems BERT and GPT (Devlin et al., 2019; Radford et al., 2018) as they use a different tokenization unit, which is not suitable for WSD.

[1] The model implementation and weights are available in the `allennlp` library.

[2] The corpora used are the 2008 to 2012 news crawls, available at `http://data.statmt.org/news-crawl/en/`

[3] `https://github.com/zalandoresearch/flair`

[4] There are no further specifications about the composition of the training corpus.

5 Evaluation tasks

As the first and main experiment we train and evaluate a WSD Transformer classifier (Section 5.1.1) using QBERT and comparison contextualized embeddings. To corroborate the results, as further experiments we evaluate the performance of the contextualized embeddings on the Word-in-Context task (Pilehvar and Camacho-Collados, 2019) (Section 5.2).

5.1 Word Sense Disambiguation

5.1.1 Setup

To perform our WSD experiment we train a simple Transformer-based classifier, which we evaluate on all-words WSD benchmark datasets. We use F1 on the test set as a measure of performance.

Architecture Our Transformer classifier takes as input the w-weighted mean between the word embeddings produced by the Encoder (the Adaptive Input layer in the case of QBERT, the character-level CNN in the case of ELMo) and the contextualized embeddings. We freeze the Encoder and only train w and the weights of the Transformer classifier. As Flair has no word embeddings, we concatenate the outputs of the forward and backward models with GloVe embeddings (Pennington et al., 2014), and substitute the weighted mean with a dense layer projecting the concatenated matrix to the Transformer hidden dimension. The classifier produces a probability distribution over an output vocabulary which includes all the possible synsets plus a special `<untagged>` symbol for words with no associated tag. During training only, we treat monosemous words as tagged. At test time, we predict the synset with the highest probability among those associated with the lemma of the target word. Importantly, we do not employ any Most Frequent Sense backoff strategy.

Hyperparameters All models are trained with Adam for a maximum of 60 epochs. We use a similar learning rate scheduling scheme as in the CLM training, first linearly increasing the value from 10^{-5} to 10^{-3}, then using a cosine CLR scheduler with period 200, maximum learning rate 10^{-3} and minimum learning rate 10^{-4}; with each cycle the maximum and minimum are halved, while the period is doubled.

Training and Test Data For each comparison system we train two WSD classifiers, one using only SemCor as training corpus and the other using the concatenation of SemCor and the corpus of WordNet's Tagged Glosses[5] (WTG). WTG includes 117659 manually disambiguated WordNet synset glosses, with 496776 annotated tokens. We test the performance of the models on the English all-words evaluation datasets from the SensEval and SemEval WSD evaluation campaigns, namely Senseval-2 (Edmonds and Cotton, 2001), Senseval-3 (Snyder and Palmer, 2004), SemEval-07 (Pradhan et al., 2007), SemEval-13 (Navigli et al., 2013), SemEval-15 (Moro and Navigli, 2015) and their concatenation (ALL). We use SemEval-2015 as our development set, to select the best epoch of the run. We use the version of SemCor and the evaluation datasets included in the WSD framework[6] of Raganato et al. (2017b).

5.1.2 Results

We show in Table 1 and Table 2 the results of the evaluation on all-words WSD of the Prediction Head trained on top of QBERT and the comparison systems. Our best model beats all the previously established results on all evaluation datasets. While the performance of the systems using SBERT and ELMo are also very competitive, in many cases exceeding the state of the art, QBERT consistently outperforms them, achieving one of the largest performance gains in years.

SemCor If we restrict the comparison to models trained on SemCor (Table 1), QBERT beats the state of the art with a margin of 0.7 points on Semeval-07 and 1.5 points on SemEval-13. On our development set, SemEval-15, we get a score 2 points over the state of the art. On Semeval-2 and Senseval-3 our F1 score is in the same ballpark as, respectively, Yuan et al. (2016) and Uslu et al. (2018). QBERT also performs well measured against our comparison systems. SBERT achieves lower performance across the board, but attains overall competitive results on all the datasets. ELMo performs on a par with SBERT on the concatenation of all datasets, but gets better results than QBERT on the development set. Flair, perhaps as a result of its purely character-based nature, is severely outperformed on most datasets.

[5] http://wordnetcode.princeton.edu/glosstag.shtml

[6] http://lcl.uniroma1.it/wsdeval/

Systems	Dev. set	S2	S3	S07	S13	S15	ALL
IMS (Melacci et al., 2018)	–	0.702	0.688	0.622	0.653	0.693	0.681
IMSWE (Melacci et al., 2018)	–	0.722	0.699	0.629	0.662	0.719	0.696
IMSC2V$_{+PR}$ (Melacci et al., 2018)	–	**0.738**	0.719	0.633	0.682	0.728	0.713
supWSDEmb (Papandrea et al., 2017)	–	0.727	0.706	0.631	0.668	0.718	–
BiLSTM$_{att+lex}$ (Raganato et al., 2017a)	S07	0.720	0.694	0.637	0.664	0.724	0.699
GAS$_{ext}$(concat) (Luo et al., 2018)	S07	0.722	0.705	–	0.672	0.726	0.706
BiLSTM (Vial et al., 2018)	WTG	0.735$_\dagger$	0.709$_\dagger$	0.625$_\dagger$	0.676$_\dagger$	0.716$_\dagger$	0.705$_\dagger$
BiLSTM+VR (ensemble) (Vial et al., 2018)	WTG	0.731	0.706	0.613	0.712	0.716	0.718
LSTM+LP (Yuan et al., 2016)	–	**0.738**	0.718	0.635	0.695	0.726	–
fastSense (Uslu et al., 2018)	S2	0.735	**0.735**	0.624	0.662	0.732	–
SotA (single model)	–	0.735	0.735	0.637	0.695	0.728	0.713
SotA (ensemble)	–	0.735	0.735	0.637	0.712	0.728	0.718
ELMo + WSD Pred. Head	S15	0.719	0.718	0.607	0.703	**0.762**	0.714
Flair + WSD Pred. Head	S15	0.702	0.702	0.615	0.694	0.732	0.699
SBERT + WSD Pred. Head	S15	0.731	0.719	0.640	0.694	0.741	0.715
QBERT + WSD Pred. Head$\star$	S15	0.734	0.732	**0.644**	**0.710**	0.743	**0.724**

Table 1: Results of the evaluation on the English datasets of models trained on SemCor. We include as competitors supervised systems capable of performing all-words WSD on the whole WordNet inventory. We report in the 'Dev set.' column the development corpus used (if any). The $\dagger$ symbol indicates that the result is an average of 20 training runs. **Bold** means that the result is the highest one among non ensemble models. We use $\star$ to mark significant improvement against best single model performance on ALL according to a z-test ($p < 0.05$). We report in the four row blocks 1) competitor SVM-based systems; 2) competitor neural networks; 3) state of the art as the maximum value in the previous rows; 4) QBERT and our comparison systems.

Systems	Dev. set	S2	S3	S07	S13	S15	ALL
BiLSTM (Vial et al., 2018)	SMP	0.744$_\dagger$	0.708$_\dagger$	0.625$_\dagger$	0.708$_\dagger$	0.745$_\dagger$	0.719$_\dagger$
BiLSTM+VR (ensemble) (Vial et al., 2018)	SMP	0.752	0.701	0.668	0.726	0.745	0.727
SotA (single model)	–	0.744	0.735	0.637	0.708	0.745	0.719
SotA (ensemble)	–	0.752	0.735	0.668	0.726	0.745	0.727
ELMo + WSD Pred. Head$\star$	S15	0.743	0.726	0.648	**0.754**	0.786	0.741
Flair + WSD Pred. Head	S15	0.728	0.715	0.646	0.725	0.775	0.725
SBERT + WSD Pred. Head$\star$	S15	0.746	0.722	**0.675**	0.717	0.783	0.734
QBERT + WSD Pred. Head$\star$	S15	**0.757**	**0.739**	0.659	0.746	**0.791**	**0.749**

Table 2: Results of the evaluation on the English datasets of models trained on the concatenation of SemCor and WTG. We use the same notation as in Table 1, employing $\star$ to mark significance against the single model state of the art. Models from Vial et al. (2018), marked by SMP, use a random sample of sentences from SemCor and WTG as development. In the row blocks we report 1) competitor neural networks; 2) the state of the art as the maximum value in the previous rows and in Table 1; 3) QBERT and our comparison systems.

SemCor and WTG When we report in the comparison systems trained on the concatenation of SemCor and WTG (Table 2), QBERT beats the state of the art more consistently and by a larger margin. We reach 1.3 points above the previous state of the art on Senseval-2, 0.4 points on Senseval-3, 2.4 on Semeval-07, 3.8 on Semeval-13 and 4.6 on Semeval-15 (which is however our development set). On the concatenation of all datasets, our margin is of 3 points. Even if we consider the ensemble of 20 models trained on SemCor and WTG by Vial et al. (2018), we get better results on every dataset with the exception of SemEval-07, with a difference of 2.2 points on ALL. With respect to our own comparison systems, QBERT performs better than ELMo, Flair and SBERT in this setting as well. ELMo gets very competitive results compared to the previous state of the art, which it beats on many datasets. Compared to QBERT, however, it gets worse results on almost every dataset, with the single exception of SemEval-13. Flair underperforms also in this setting. SBERT achieves good performances, but still consistently lower than QBERT, except for SemEval-07, which is however a small dataset whose F1 scores show high variance across different training runs.

5.2 Word-in-Context

The Word-in-Context task (WiC) was recently established by Pilehvar and Camacho-Collados (2019). Like WSD, WiC requires identification of a contextually appropriate meaning, but it is framed as a simpler binary classification task: given two contextual occurrences of the same lemma, predict whether the pair shares the same sense. The dataset includes 8320 context pairs, divided between training and development (the test set has not yet been released). By including the same target word in each element of the pair, the dataset is constructed in such a way that context-insensitive word embeddings would not perform better than the random baseline. Thus, the dataset is an ideal evaluation set for assessing the quality of the semantic information encoded in contextualized embeddings.

5.2.1 Setup

Among the baselines offered by Pilehvar and Camacho-Collados (2019), one uses ELMo contextualized embeddings as input to a simple two-layer feed-forward classifier. We replicate the same setting, but using the concatenation of QBERT Encoder word and contextualized embeddings as input instead. Also, as the authors have not yet released the gold keys for the development set and evaluation can only be performed by uploading a prediction file to the Codalab competition page[7], we take $\frac{1}{10}$ of the training instances as our development set, and use the provided development set for testing. We train the system for a max of 40 epochs, submitting the epoch with best accuracy on the development split. WiC's scorer reports the accuracy calculated on the predictions. We implement the same system employing ELMo, Flair (using the concatenation of GloVe and contextualized embeddings) and SBERT as well. Performance is measured by mean accuracy over 5 runs.

5.2.2 Results

In Table 3 we show the results of the evaluation on the WiC development set.

[7] https://competitions.codalab.org/competitions/20010

System	Acc. μ	Acc. σ
Elmo (ours)	59.97	1.41
Flair	60.23	0.91
SBERT	60.03	1.13
QBERT	**60.74**	1.22

Table 3: Results of our ELMo, SBERT and QBERT models on the WiC dataset evaluation dataset. We report the mean and standard deviation of the accuracy for 5 runs.

In this setting ELMo performs on a par with SBERT and Flair, while QBERT achieves the best result. Note that the quasi-deeply bidirectional encoding that QBERT can exploit through the BiTransformer might see its effectiveness reduced in this setting since many pairs feature limited context, even as short as 2 or 3 words. Still, the results of the WiC task corroborate those of the all-words WSD, providing evidence that joint encoding is crucial to better performance in word-level semantics.

6 Conclusion

In this paper we showed that the use of contextualized embeddings enables a WSD system to beat the previous state of the art. Moreover, we demonstrated that the use of the BiTransformer coattentive mechanism in the QBERT contextualized embeddings model itself results in even stronger performance. As a result, we attain one of the largest gains in WSD performance in years, with a margin of 3 points over the best reported single model on the concatenation of all datasets, and of 2.2 points over the best ensemble model in the literature. We leave for future work the assessment of whether the gains brought about by the use of the BiTransformer in QBERT carry over to other tasks, helping to bridge the gap between CLM-based and fully bidirectional MLM-based contextualized embeddings. We release the code to train the QBERT Encoder and the WSD classifier, along with pretrained models at https://github.com/mbevila/qbert.

Acknowledgments

The authors gratefully acknowledge the support of the ERC Consolidator Grant MOUSSE No. 726487 under the European Union's Horizon 2020 research and innovation programme.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual String Embeddings for Sequence Labeling. In *Proc. of COLING*. pages 1638–1649. https://aclanthology.info/papers/C18-1139/c18-1139.

Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR* abs/1607.06450. http://arxiv.org/abs/1607.06450.

Alexei Baevski and Michael Auli. 2018. Adaptive input representations for neural language modeling. *CoRR* abs/1809.10853. http://arxiv.org/abs/1809.10853.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*. https://arxiv.org/abs/1409.0473.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. of NAACL-HLT*. pages 4171–4186. https://aclweb.org/anthology/papers/N/N19/N19-1423/.

Philip Edmonds and Scott Cotton. 2001. SENSEVAL-2: Overview. In *Proc. of SENSEVAL-2*. pages 1–5. https://www.aclweb.org/anthology/papers/S/S01/S01-1001/.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Language, Speech, and Communication. MIT.

Edouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. 2017. Efficient softmax approximation for GPUs. In *Proc. of ICML*. pages 1302–1310. http://proceedings.mlr.press/v70/grave17a.html.

Lushan Han, Abhay L. Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. 2013. UMBC_ebiquity-CORE: Semantic Textual Similarity Systems. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*. pages 44–52. http://aclweb.org/anthology/S/S13/S13-1005.pdf.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proc. of CVPR*. pages 770–778. https://doi.org/10.1109/CVPR.2016.90.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly Predicting Predicates and Arguments in Neural Semantic Role Labeling. In *Proc. of ACL*. pages 364–369. https://aclanthology.info/papers/P18-2058/p18-2058.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *Proc. of ACL*. pages 328–339. https://www.aclweb.org/anthology/papers/P/P18/P18-1031/.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for Word Sense Disambiguation: An Evaluation Study. In *Proc. of ACL*. pages 897–907. http://aclweb.org/anthology/P/P16/P16-1085.pdf.

Mikael Kagebäck and Hans Salomonsson. 2016. Word Sense Disambiguation using a Bidirectional LSTM. In *Proc. of COLING*. pages 51–56. https://aclanthology.info/papers/W16-5307/w16-5307.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *CoRR* abs/1901.07291. http://arxiv.org/abs/1901.07291.

Minh Le, Marten Postma, Jacopo Urbani, and Piek Vossen. 2018. A Deep Dive into Word Sense Disambiguation with LSTM. In *Proc. of COLING*. Association for Computational Linguistics, pages 354–365. https://aclanthology.info/papers/C18-1030/c18-1030.

Fuli Luo, Tianyu Liu, Qiaolin Xia, Baobao Chang, and Zhifang Sui. 2018. Incorporating Glosses into Neural Word Sense Disambiguation. In *Proc. of ACL*. Association for Computational Linguistics, pages 2473–2482. https://aclanthology.info/papers/P18-1230/p18-1230.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proc. of EMNLP*. Association for Computational Linguistics, pages 1412–1421. http://aclweb.org/anthology/D/D15/D15-1166.pdf.

Stefano Melacci, Achille Globo, and Leonardo Rigutini. 2018. Enhancing Modern Supervised Word Sense Disambiguation Models by Semantic Lexical Resources. In *Proc. of LREC*. pages 1012–1017. https://www.aclweb.org/anthology/papers/L/L18/L18-1163/.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning Generic Context Embedding with Bidirectional LSTM. In *Proc. of COLING*. pages 51–61. http://aclweb.org/anthology/K/K16/K16-1006.pdf.

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. http://aclweb.org/anthology/S/S15/S15-2049.pdf.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Comput. Surv.* 41(2):10:1–10:69. https://doi.org/10.1145/1459352.1459355.

Roberto Navigli, David Jurgens, and Daniele Vannella. 2013. SemEval-2013 Task 12: Multilingual Word Sense Disambiguation. In *Proc. of SemEval*. pages 222–231. http://aclweb.org/anthology/S/S13/S13-2040.pdf.

Simone Papandrea, Alessandro Raganato, and Claudio Delli Bovi. 2017. SupWSD: A Flexible Toolkit for Supervised Word Sense Disambiguation. In *Proc. of EMNLP*. pages 103–108. https://aclanthology.info/papers/D17-2018/d17-2018.

Tommaso Pasini and Roberto Navigli. 2017. Train-O-Matic: Large-Scale Supervised Word Sense Disambiguation in Multiple Languages without Manual Training Data. In *Proc. of EMNLP*. pages 78–88. https://aclanthology.info/papers/D17-1008/d17-1008.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *Proc. of EMNLP*. pages 1532–1543. http://aclweb.org/anthology/D/D14/D14-1162.pdf.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proc. of NAACL-HLT*. pages 2227–2237. https://aclanthology.info/papers/N18-1202/n18-1202.

Mohammad Taher Pilehvar and José Camacho-Collados. 2019. Wic: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proc. of NAACL-HLT*. pages 1267–1273. https://aclweb.org/anthology/papers/N/N19/N19-1128/.

Sameer Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007. SemEval-2007 Task-17: English Lexical Sample, SRL and All Words. In *Proc. of SemEval 2007*. pages 87–92. http://aclweb.org/anthology/S07-1016.

Ofir Press and Lior Wolf. 2017. Using the Output Embedding to Improve Language Models. In *Proc. of EACL*. pages 157–163. https://aclanthology.info/papers/E17-2025/e17-2025.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-Training .

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017a. Neural Sequence Learning Models for Word Sense Disambiguation. In *Proc. of EMNLP*. pages 1156–1167. https://aclanthology.info/papers/D17-1120/d17-1120.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017b. Word Sense Disambiguation: A Unified Evaluation Framework and Empirical Comparison. In *Proc. of ACL*. pages 99–110. https://www.aclweb.org/anthology/papers/E/E17/E17-1010/.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proc. of ACL*. pages 1717–1725. http://aclweb.org/anthology/P/P16/P16-1162.pdf.

Leslie N. Smith. 2017. Cyclical Learning Rates for Training Neural Networks. In *Proc. of WACV*. pages 464–472. https://doi.org/10.1109/WACV.2017.58.

Benjamin Snyder and Martha Palmer. 2004. The English all-words task. In *Proc. of SENSEVAL-3*. pages 41–43. https://aclanthology.info/papers/W04-0811/w04-0811.

Kaveh Taghipour and Hwee Tou Ng. 2015. One Million Sense-Tagged Instances for Word Sense Disambiguation and Induction. In *Proc. of CoNLL*. pages 338–344. http://aclweb.org/anthology/K/K15/K15-1037.pdf.

Tolga Uslu, Alexander Mehler, Daniel Baumartz, Alexander Henlein, and Wahed Hemati. 2018. FastSense: An Efficient Word Sense Disambiguation Classifier. In *Proc. of LREC*. pages 1042–1046. https://www.aclweb.org/anthology/papers/L/L18/L18-1168/.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Proc. of NIPS*. pages 6000–6010. https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Loïc Vial, Benjamin Lecouteux, and Didier Schwab. 2018. Improving the coverage and the generalization ability of neural word sense disambiguation through hypernymy and hyponymy relationships. *CoRR* abs/1811.00960. http://arxiv.org/abs/1811.00960.

Dirk Weissenborn, Leonhard Hennig, Feiyu Xu, and Hans Uszkoreit. 2015. Multi-Objective Optimization for the Joint Disambiguation of Nouns and Named Entities. In *Proc. of ACL*. pages 596–605. http://aclweb.org/anthology/P/P15/P15-1058.pdf.

Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. 2016. Semi-supervised Word Sense Disambiguation with Neural Models. In *Proc. of COLING*. pages 1374–1385. http://aclweb.org/anthology/C/C16/C16-1130.pdf.

Zhi Zhong and Hwee Tou Ng. 2010. It Makes Sense: A Wide-Coverage Word Sense Disambiguation System for Free Text. In *Proc. of ACL, System Demonstrations*. pages 78–83. http://www.aclweb.org/anthology/P10-4014.