Using Prior Knowledge to Guide BERT's Attention in Semantic Textual Matching Tasks

Tingyu Xia
School of Artificial Intelligence, Jilin University
Key Laboratory of Symbolic Computation and Knowledge
Engineering, Jilin University
xiaty19@mails.jlu.edu.cn

Yue Wang
School of Information and Library Science
University of North Carolina at Chapel Hill
wangyue@unc.edu

Yuan Tian*
School of Artificial Intelligence, Jilin University
Key Laboratory of Symbolic Computation and Knowledge
Engineering, Jilin University
yuantian@jlu.edu.cn

Yi Chang*
School of Artificial Intelligence, Jilin University
International Center of Future Science, Jilin University
Key Laboratory of Symbolic Computation and Knowledge
Engineering, Jilin University
yichang@jlu.edu.cn

ABSTRACT

We study the problem of incorporating prior knowledge into a deep Transformer-based model, i.e., Bidirectional Encoder Representations from Transformers (BERT), to enhance its performance on semantic textual matching tasks. By probing and analyzing what BERT has already known when solving this task, we obtain better understanding of what task-specific knowledge BERT needs the most and where it is most needed. The analysis further motivates us to take a different approach than most existing works. Instead of using prior knowledge to create a new training task for fine-tuning BERT, we directly inject knowledge into BERT's multi-head attention mechanism. This leads us to a simple yet effective approach that enjoys fast training stage as it saves the model from training on additional data or tasks other than the main task. Extensive experiments demonstrate that the proposed knowledge-enhanced BERT is able to consistently improve semantic textual matching performance over the original BERT model, and the performance benefit is most salient when training data is scarce.

KEYWORDS

Prior Knowledge, Semantic Textual Similarity, Deep Neural Networks, BERT

*Joint Corresponding Authors

1 INTRODUCTION

Measuring semantic similarity between two pieces of text is a fundamental and important task in natural language understanding. Early works on this task often leverage knowledge resources such as WordNet [28] and UMLS [2] as these resources contain well-defined types and relations between words and concepts. Recent works have shown that deep learning models are more effective on this task by learning linguistic knowledge from large-scale text data and representing text (words and sentences) as continuous trainable embedding vectors [10, 17].

Among these works, deep Transformer-based models such as the Bidirectional Encoder Representations from Transformers (BERT) has shown very promising performance, thanks to the contextualized word representations learned through its multi-head attention mechanism and unsupervised pre-training on large corpus [39]. To further instill knowledge into BERT, recent works have proposed different training tasks beyond the original masked language modeling and next-sentence prediction tasks [10]. These include other unsupervised pre-training tasks such as span prediction [21] and domain-specific pre-training [14], as well as knowledge-based tasks such as semantic role prediction [48], entity recognition [47], and relation prediction [26]. Among these approaches, some are able to improve the semantic textual similarity (STS) task performance [21], while others are sometimes detrimental to the task [47, 48].

In this paper, we explore a different approach to incorporating external knowledge into deep Transformers for STS tasks. Instead of creating additional training tasks, we directly inject knowledge into a Transformer's multi-head attention. On the one hand, deep Transformers are usually seen as complex "black boxes" that can only be improved through well-formulated training (or fine-tuning) tasks. On the other hand, the research community has recently seen a surge of interest in "opening the black box" to understand the internal mechanism of Transformer-based models. These include analyses of layer-wise attentions, syntactic knowledge, semantic knowledge, and world knowledge learned at each layer [8, 12, 16, 37]. We refer the reader to Rogers et al. [33] for a comprehensive synthesis. Motivated by these recent studies, we aim to take one step further – in addition to *observing* the mechanisms

inside BERT, we explore the possibility of purposefully *steering* its internal mechanism and study the effects of such intervention. In particular, we use prior knowledge to guide the attention of BERT towards a better performance on STS tasks. Although previous work has proposed to use prior knowledge in guiding the attention of other deep learning models, such as recurrent neural networks [7] and long short-term memory networks [4], to the best of our knowledge, we are the first to explore direct intervention of BERT's attention mechanism.

To make informed decisions on what kind of knowledge to add and where to add it in BERT, we conducted in-depth analyses of the BERT model on the STS task. The results inform us to inject word similarity knowledge into BERT's attention at the first layer. Extensive experiments on standard STS datasets demonstrate that our approach consistently improves BERT on STS tasks, and the improvement effect is most salient when training data size is very limited. The main contributions of this paper are as follows:

- We show that while a pre-trained BERT model has prior knowledge on STS tasks which a non-Transformer deep model does not possess, its knowledge in word similarity is still inadequate and can be further improved.
- We propose an efficient and effective method for injecting word similarity knowledge into BERT – not through adding another training task, but by directly guiding the model's attention. This approach not only performs at least as well as adding a new training task, but also saves substantial training time.
- We show that the proposed method is able to consistently improve the STS performance of BERT and the benefit is especially salient when training data is scarce.

2 RELATED WORK

Semantic textual similarity is a well-studied task in natural language processing (NLP). In most previous works, feature engineering was the main approach. Early research on this task explored different types of sparse features and confirmed their value. This includes (1) syntactical and lexical features extracted from word and sentence pairs [9, 44], (2) knowledge-based features using WordNet, which make extensive use of word similarity information [13], (3) semantic relation knowledge [19] and logical rules [1] derived from WordNet, (4) word-alignment features which discover and align similar semantic units in a pair of sentences [34, 35]. Before the advent of BERT, other types of deep neural networks have already been used to solve STS tasks. These include feedforword neural networks (FNN), convolutional neural network s (CNN) and long short-term memory-Networks (LSTM). For example, Hu et al. used a CNN model that combines hierarchical structures with layer-by-layer composition and pooling [18]. Yin et al. presented a general attention-based CNN for modeling a pair of sentences [46]. Parikh et al. used attention to decompose the problem into sub-problems that can be solved separately in the form of natural language inference (NLI) tasks [29]. Tomar et al. proposed a variant of the decomposable attention model which achieved good results on paraphrase identification task [38]. Besides, Chen et al. utilized tree-LSTM and soft-alignment to improve the performance of the ESIM model on NLI tasks [5]. Lan et al. showed that the tree-LSTM

model also did well in STS task [23]. Among many variants of the ESIM [6, 25], KIM leveraged external knowledge to improve performance of ESIM on semantic similarity tasks [6].

In recent years, the shift from neural network architecture engineering to large-scale pre-training has significantly improved NLP tasks, demonstrating the power of unsupervised pre-training. Outstanding examples include Embedding from Language Models (ELMo) [31], Generative Pre-trained Transformers (GPT) [32], Bidirectional Encoder Representations from Transformers (BERT) [10], and Generalized Auto-regressive Pre-training (XLNet) [45]. Providing fine-grained contextual word embedding, these pre-trained models can be either easily applied to downstream tasks as encoders or directly fine-tuned for downstream tasks. As the most prominent model in recent years, BERT and many of its variants, including AlBERT [24], RoBERTa [27], SemBERT [48], ERNIE [47], K-BERT [26], and DeBERTa [15], have achieved superior results in many NLP tasks. Among them, SemBERT, ENRIE and K-BERT all add knowledge to the orginal BERT model, but in different ways. SemBERT and K-BERT are fine-tuning methods without pre-training and they are capable of loading model parameters from a pre-trained BERT. SemBERT incorporates explicit contextual semantics from pre-trained semantic role labeling. K-BERT is a knowledge-enhanced language model, in which knowledge triples are injected into the sentences as domain knowledge. K-BERT is only trained on Chinese text corpus. ENRIE is an improvement on top of BERT that utilizes both large-scale textual corpora and knowledge graph. It takes advantage of lexical, syntactic, and semantic knowledge information simultaneously. Although the above three BERT variants are shown to improve performance on a variety of language understanding tasks, the results largely depend on the size of task-specific training data. For instance, the performance of ERNIE was reported to be unstable on semantic similarity tasks with small datasets [47].

3 WHAT HAS BERT ALREADY KNOWN ABOUT SEMANTIC TEXTUAL SIMILARITY?

Before adding task-specific knowledge to BERT, the first and foremost question is: *what has BERT already known (and not known) about the task?* Answering this question will allow us to design our approach in an informed, targeted manner.

To operationalize our answer to the above question, we use a classical STS dataset – the Microsoft Research Paraphrase Corpus (MPRC) [11], as a pilot data set. Each data instance is a pair of sentences. The goal is to detect if one is a paraphrase of the other (binary classification). We conduct two pilot studies as follows.

(1) Data Augmentation Study. In this study, we augment the pilot dataset with various types of prior knowledge that are potential useful for determining semantic textual similarity, and compare the performance of BERT trained on the original data vs. the augmented data. The intuition is that if a particular data augmentation strategy can improve BERT's STS performance, it indicates that BERT still lacks the corresponding knowledge. Otherwise it implies that BERT has already "known" the corresponding knowledge. As a comparison, we apply the same procedure to a non-Transformer model for the STS task, the Enhanced Sequential Inference Model (ESIM) [5], and see if the same data augmentation strategy can

benefit such a model (which indicates that the model lacks the corresponding knowledge). We defer detailed descriptions of BERT and ESIM to Section 4.

(2) Layer-wise Performance Study. In this study, we freeze a BERT model (except for the softmax classification output head) and use the pretrained contextualized word representation to perform the STS task. The goal is to observe *where* BERT stores the most STS-related knowledge, and where such knowledge is the most lacking. This allows us to make an informed decision on which BERT layer(s) to inject prior knowledge.

Below we discuss the results and implications of the two studies.

3.1 Effect of Data Augmentation

In short, data augmentation techniques use prior knowledge and relatively simple algorithms to derive new training data from original training data [43]. It is an effective way of enriching the original data with task-specific knowledge. Below we describe several data augmentation strategies and the corresponding prior knowledge related to STS.

(1) *Split and swap*: For each sentence, split it at a random position and swap the two segments. The assumption is that although a sentence could be ungrammatical after such an operation, its essential meaning should be preserved.

(2) *Add random word*: For each sentence, pick a random word from the vocabulary that is not in the sentence, and insert it at a random position in that sentence. The assumption is that such an out-of-context word acts as a noisy typo and should not affect the main meaning of the sentence.

(3) *Back translation*: All sentences in MRPC are in English. Using Google Translate, we translate the sentence to Chinese and then back to English. The assumption is that translation should rewrite the sentence in a different way but preserve meaning.

(4) *Add high-TfIdf word*: For each sentence, find the word with the highest TfIdf weight and insert it at a random position in that sentence. Words with high-TfIdf weights are usually content words, therefore repeating them should not change the meaning of the sentence by much.

(5) *Delete low-TfIdf word*: Find k words with the lowest TfIdf weights in the sentence pair. Then for each of these words, delete it from the sentence with probability p. We set $k = 5, p = 0.5$. Words with low TfIdf weights are usually *stop words* or *functional words* (e.g., "the", "of", "and"), and therefore may not be essential to the meaning of the sentence.

(6) *Replace synonyms*: For each word in a sentence, see if the word has synonyms in WordNet. If so, pick the first word in the synonym list that is not the word itself. The assumption is that replacing words with their synonyms does not change the meaning of a sentence. In principle, other resources such as ConceptNet and UMLS also provide synonym knowledge and can be used here in place of WordNet.

We use the standard train-test split provided in the MRPC dataset. Then, we apply each of the above data augmentation strategies on the training data, each resulting in a 2× increase of training data size. Figure 1 shows the F_1 performance of BERT trained by each

augmented dataset, as well as the performance without data augmentation. To minimize the effect of randomness in BERT training, performance levels are averaged across 10 different runs.

The results show that except for *Replace synonyms*, all other strategies lead to a performance drop. This indicates that BERT may have already understood the importance of stop words and content words in STS tasks, and it knows to use syntactic ordering and semantic coherence when inferring semantic similarity. The fact that back translation does not help implies that back-translated sentences may contain semantic shifts and/or syntactic errors, which can mislead BERT.
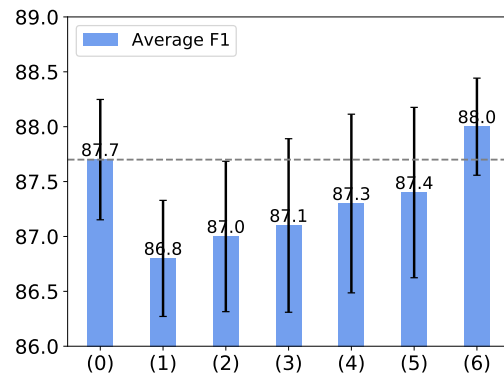

Figure 1: Effect of augmenting the MRPC data set for BERT. The X-axis represents different data augmentation strategies corresponding to the order in the text, where (0) represents the original training data without augmentation, and the serial number (1) to (6) represents *Split and swap, Add random word, Back translation, Add high-TfIdf word, Delete low-TfIdf word, Replace synonyms,* respectively. Error bars show ±1 standard deviation around the average of 10 runs.
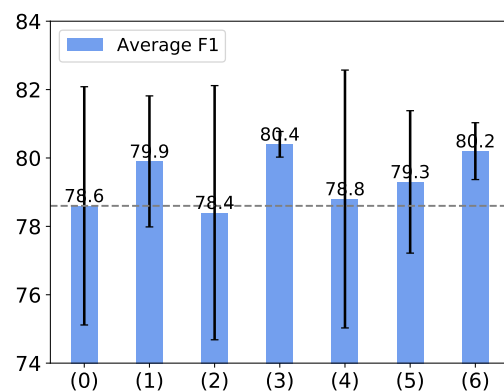

Figure 2: Effect of augmenting the MRPC data set for ESIM. X-axis labels are defined similarly as those in Figure 1.

As a comparison, we apply the same augmented data to ESIM. The results are shown in Figure 2. Note that BERT outperforms ESIM by a large margin. Here we focus on the relative differences
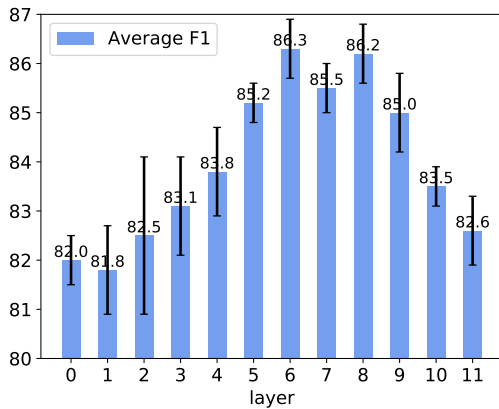
Figure 3: Performance of each BERT layer on MRPC data set.

made by different augmentation strategies on ESIM. ESIM benefits from all data augmentation strategies except for *Add random word*. Both *Back translation* and *Split and swap* are effective strategies (>1% absolute gain in F_1), which do not apply for BERT. This indicates that a pretrained BERT model has rich prior knowledge for the STS task that a pre-Transformer model does not possess. Similar to BERT, *Replace synonyms* can substantially benefit ESIM as well.

3.2 Different BERT Layers on the STS Task

Figure 3 shows the performance of using each pre-trained BERT layer on the MRPC paraphrase identification task. In this study, the original training data was used without any augmentation. Again, the performance levels are averaged across 10 different runs.

Since the BERT layer parameters are "frozen" (not trainable) in this task, the performance shows that BERT's middle layers are inherently good at this task, while lower and upper layers are not. This echoes with recent findings on BERT layers. Lower layers were found to perform broad attention across all pairs of words [8]. Middle layers were found to mostly capture transferable syntactic and semantic knowledge [16, 36]. It is not surprising that the upper layers do not perform well, as these layers are specifically tuned towards the pre-training tasks of BERT – masked language modeling and next-sentence prediction – not STS tasks.

To summarize, the data augmentation study shows that BERT knows about the importance of stop words, content words, and syntactic structure for the STS task, but its knowledge in word synonyms can be further improved. The layer-wise performance study shows that lower layers of BERT needs the most improvement for STS tasks. Combining results from the two studies, we identify a promising direction, i.e., to incorporate synonym knowledge into low layers of BERT (next section). We would like to note that the above analysis and reasoning procedure can be potentially applied to identify "knowledge deficiency" of BERT in other NLP tasks as long as one can associate different kinds of knowledge with different subsets of training data.

4 PROPOSED ALGORITHM

In this section, we design a general algorithm for incorporating synonym knowledge into the BERT model. According to the findings in Section 3, this knowledge should be added to lower layers of BERT. Since each attention layer computes similarity between all pairs of words, and it has been shown that the attention in the first layer is broad and uninformed [8, 33], we decide to use word similarity knowledge to modulate the attention at the first layer. As a comparison baseline, We add the same knowledge into the ESIM model. Below we describe our approach in detail.

4.1 Word Similarity Matrix

Given two pieces of text (i.e., sentences) $a = (w_1, \cdots, w_a, \cdots, w_{l_a})$ and $b = (w_1, \cdots, w_b, \cdots, w_{l_b})$, we construct a word similarity matrix S of size $l_a \times l_b$. The goal of this matrix S is to increase BERT's attention on semantically similar word pairs.

We calculate the value of each cell in S based on semantic relations in WordNet. For a lexical pair (w_a, w_b), if words in the pair are synonyms in WordNet, the cell value $S_{ab} = 1$. Otherwise, if w_a and w_b are not synonyms, we set S_{ab} as a value in $[0, 1]$ according to the method proposed by Wu-Palmer [42], which calculates word similarity based on their topological distance in WordNet. In cases where one or both words in a pair cannot be found in WordNet, or the pair of words do not have a valid Wu-Palmer similarity value (e.g. if one word is a stop word like "into"), we directly set the pairwise similarity value to 0. For proper names of people and places, if words in the pair are the same, the value will be set to 1, otherwise it will be 0. Figure 4 visualizes the heat map of the similarity matrix constructed for two sentence pairs. We compute word similarity matrices for all sentence pairs in both training and test dataset as a preprocessing step.

Note that our method is general and compatible with other semantic similarity resources, such as Wikipedia, ConceptNet, and UMLS (Unified Medical Language System) Metathesaurus. We can construct S using word/concept similarity knowledge in these resources as well. Also note that simple word lookup does not resolve word sense ambiguity, and WordNet does not cover many acronyms and abbreviations. We leave further refinements of word similarity matrix S for future work.

4.2 Bidirectional Encoder Representations from Transformers (BERT)

BERT is a pre-trained language model with state-of-the-art performance on many NLP tasks. We fine-tune Google's pre-trained BERT-base model in our implementation.

Word similarity knowledge is added into BERT's multi-head attention (both self-attention and cross-attention). In the embedding stage, we use the summation of three parts: token embedding, position embedding, and segment embedding, which is the same as BERT. In the Transformer stage, our method is similar to BERT.

4.2.1 Multi-Head Attention in BERT. BERT's attention function can be described as a mapping from query vector Q and a set of key-value vector pairs (K, V) to an output vector – the attention strengths. Multi-head attention linearly projects the queries, keys and values h times (h is the number of "heads") with different linear
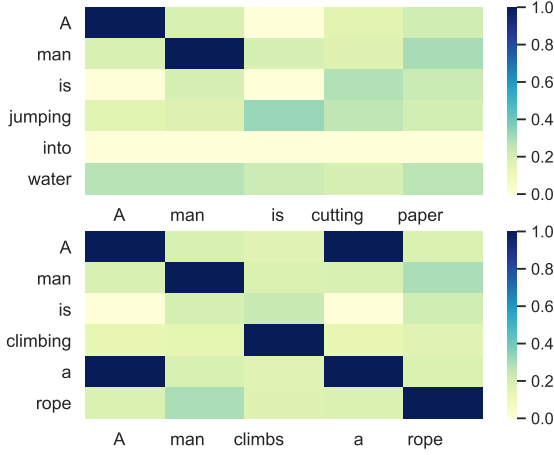
Figure 4: Heat map for two sentence pairs taken from the STS-B dataset, where the ground-truth similarity score of the first sentence pair ("A man is jumping into water" and "A man is cutting paper") is labeled as 0.8, and the similarity score of the second sentence pair is 5 ("A man is climbing a rope" and "A man climbs a rope"). The higher score, the more similar the two sentences.

projections to d_k, d_k, and d_v dimensions, respectively. Next, on each head of these projections of queries, keys and values, it performs the attention function in parallel. This produces d_v-dimensional output values. These values are concatenated and projected again to get the final attention:

$$MultiHead(Q, K, V) = Concat(head_1, \ldots, head_h)W^O ;$$
$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) ,$$
(1)

where $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W_i^O \in \mathbb{R}^{hd_v \times d_{model}}$ are parameter matrices representing the projections. d_{model} is BERT's hidden layer size. The calculation of BERT's attention uses scaled dot-product:

$$scores = QK^T + MASK ;$$
$$Attention(Q, K, V) = softmax(\frac{scores}{\sqrt{d_k}})V ,$$
(2)

where $MASK$ is a matrix used in masked language modeling, the pretraining task for BERT.

4.2.2 Knowledge-Guided Attention. To add prior knowledge, we make adjustments to the multi-head attention phase as shown in Figure 5. We calculate the Hadamard product (i.e., element-wise product) of the *scores* using the similarity matrix S to make the model pay more attention to the word pairs with higher similarities in two sentences:

$$scores = QK^T \odot S + MASK ;$$
$$Attention(Q, K, V) = softmax(\frac{scores}{\sqrt{d_k}})V ,$$
(3)

where S represents the word similarity matrix we calculated in advance (Section 4.1). As mentioned before, we only add such prior knowledge in the attention of the first BERT layer, as that layer's attention is broad and uninformed [8, 33].
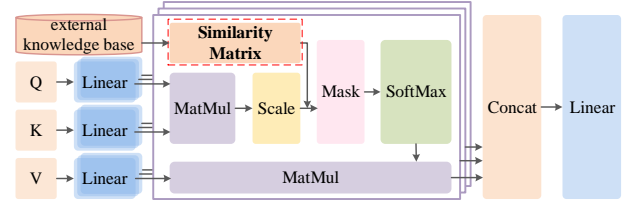

Figure 5: Injecting word similarity knowledge in BERT's multi-head attention.

4.3 Enhanced Sequential Inference Model

In this section, we show that our method can also be used to incorporate knowledge into the Enhanced Sequential Inference Model (ESIM). This model explicitly computes pairwise word similarities as a component [5]. Before BERT was proposed, ESIM was one of the state-of-the-art methods for sentence pair modeling tasks. In this paper, we use ESIM as a non-Transformer model for baseline comparison.

4.3.1 The Original ESIM. ESIM first uses a bi-directional long short-term memory network (BiLSTM) to encode the input sentences a and b. For the i-th token of sentence a, the hidden state vector generated by the BiLSTM is denoted as $\bar{a}_i$. Similarly we can define a state vector $\bar{b}_j$ for the j-th token of sentence b:

$$\bar{a}_i = BiLSTM(a, i) \quad \forall_i \in [1, \ldots, l_a] ;$$
$$\bar{b}_j = BiLSTM(b, j) \quad \forall_j \in [1, \ldots, l_b] .$$
(4)

ESIM computes the attention weights e_{ij} as the inner product of a hidden state pair $\langle \bar{a}_i, \bar{b}_j \rangle$ for all tokens in sequences a and b:

$$e_{ij} = \bar{a}_i^T \bar{b}_j .$$
(5)

The attention weights are then normalized and used as coefficients to compute a "soft alignment" between a word in one sentence and all words in the other sentence:

$$\bar{\beta}_i = \sum_{j=1}^{l_b} \frac{\exp(e_{ij})}{\sum_{k=1}^{l_a} \exp(e_{ik})} \bar{b}_j ;$$
$$\bar{\alpha}_j = \sum_{i=1}^{l_a} \frac{\exp(e_{ij})}{\sum_{k=1}^{l_b} \exp(e_{kj})} \bar{a}_i ,$$
(6)

where $\bar{\beta}_i$ is the "subphrase" in b that is softly aligned to words in a and vice versa for $\bar{\alpha}_j$.

After soft-alignment, ESIM uses Tree-LSTM model to help collect local inference information over linguistic phrases and clauses. The output of the Tree-LSTM model is then max-pooled, averaged, and concatenated as input to a final multilayer perception. We refer the reader to [5] for more details.
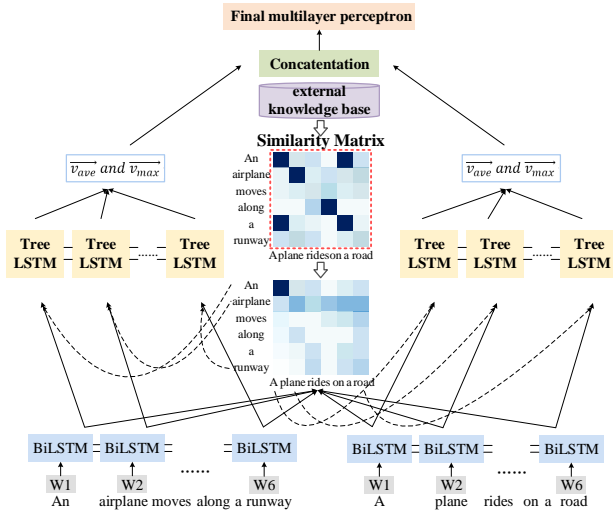
Figure 6: Injecting Word Similarity Knowledge in ESIM

4.3.2 Knowledge-Guided ESIM. The overall architecture of the knowledge-enhanced ESIM is illustrated in Figure 6. To inject word similarity knowledge, we also compute the Hadamard product between the attention weight matrix e and the word similarity matrix S defined in Section 4.1:

$$p_{ij} = e_{ij} \odot S_{ij} . \tag{7}$$

So that the soft alignment will be updated as follows:

$$
\begin{aligned}
\overline{\beta}_i &= \sum_{j=1}^{l_b} \frac{\exp(p_{ij})}{\sum_{k=1}^{l_a} \exp(p_{ik})} \overline{b}_j ; \\
\overline{\alpha}_j &= \sum_{i=1}^{l_a} \frac{\exp(p_{ij})}{\sum_{k=1}^{l_b} \exp(p_{kj})} \overline{a}_i ,
\end{aligned}
\tag{8}
$$

The remaining steps are similar to the original ESIM.

5 EXPERIMENTS

In this section, we evaluate our proposed method and compare it with baseline models that do not incorporate prior knowledge.

The benefit of incorporating prior knowledge in machine learning models is most salient when training data is small. Indeed, when training data is abundant, those data already contain sufficient task-specific knowledge, diminishing the benefit of prior knowledge. Therefore besides the standard setting where different models are evaluated on 100% training data, we are more interested in evaluating our approach as we vary the size of training data from small to large. This motivates us to further conduct learning curve analyes.

5.1 Datasets

We use four popular benchmark datasets in our experiments: two relatively small semantic textual similarity datasets and two relatively large paraphrase identification datasets.

- **MRPC** [11] is a corpus of sentence pairs with artificial annotations automatically extracted from online news sources

to indicate whether each pair of sentences captures the paraphrase/semantic equivalence relationship through binary judgment. It has 4,076 train data and 1,725 test data. In this paper, we split 10% training data as the validation set according to GLUE [40] standardized splits. In Section 3, we used this dataset as a pilot to understand BERT.

- **STS-B** [3] is a collection of sentence pairs extracted from news headlines, video headlines, image headlines, and natural language inference data. It comprises a selection of the English datasets used in the STS tasks organized in the context of SemEval between 2012 and 2017 which includes 5,749 train data, 1,500 development data and 1,379 test data.
- **QQP** [20] is a dataset used to determine whether a question pair is duplicated. It consists of more than 400,000 rows of potential question duplicate pairs collected from Quora.com. In this paper, we use the same data and split by Wang in article [41], with 10,000 question pairs each for development and test. Besides, in both development and test set, the number of both paraphrasing and non-paraphrasing sentence pairs is 5000. Accuracy is used as the evaluation metric for this dataset.
- **Twitter-URL** [22] is collected from tweets that share the same URL of news articles by Lan. It includes 56,787 sentence pairs, and each sentence pair is annotated by 6 Amazon Mechanical Turk workers. Therefore, a total of 6 workers judged if this pair is a paraphrase or not. If n ≤ 2 workers were positive, we treat them as non-paraphrasing; if n ≥ 4, we treat them as paraphrasing; if n = 3, we discard them. After this treatment, there were 42,220 pairs for training and 9,334 pairs for test.

5.2 Implementation Details

We implement the models with the same PyTorch framework. Below, we summarize the implementation details that are key for reproducing results for each model. (The source code is available at https://github.com/xiatingyu/Bert_sim).

- **BERT**: We compare our proposed model with a BERT model without prior knowledge. Both models adopt the configuration of Google's BERT$_{base}$ [10]. We set the number of both self-attention layers and heads as 12, and the dimension of embedding vectors as 768. The total number of trainable parameters of both the original BERT$_{base}$ and our proposed model (called BERT$_{base}$-Sim) are the same (110M), therefore we are performing a head-to-head comparison.
- **ESIM**: Word embeddings of ESIM model is initialized with 840B-300d Glove word vectors [30]. Embeddings of out-of-vocabulary words are randomly initialized. All parameters, including word embeddings, are updated during training. In order to verify the influence of prior knowledge on the model effect, we use the same parameters mentioned in paper [5], including optimization method, learning rate, dropout rate. We call our proposed model ESIM-Sim.

5.3 Performance Across Four STS Datasets

We add prior knowledge to both ESIM model and BERT model and train them on the above four datasets. Table 1 shows the results

Table 1: Semantic textual matching performance in four datasets (%), based on our re-implementation of each method in PyTorch. Each performance value of ESIM and ESIM-Sim ($BERT_{base}$ and $BERT_{base}$-Sim) are the average of 10 (5) runs with different random seeds, respectively. Values in the parentheses are standard deviations (SD). PC: Pearson correlation coefficient. SC: Spearman's rank correlation coefficient. The best average performance in each column is in bold.

	MRPC		STS-B		QQP		TwitterURL	
	F1 (SD)	Macro-F1 (SD)	PC (SD)	SC (SD)	F1 (SD)	Accuracy (SD)	F1 (SD)	Macro-F1 (SD)
ESIM	78.5 (2.0)	56.5 (6.2)	47.2 (0.7)	44.0 (0.8)	87.6 (0.5)	87.5 (0.5)	59.1 (2.3)	71.3 (2.1)
ESIM-Sim	79.1 (1.5)	60.3 (5.8)	59.3 (1.3)	56.4 (1.4)	87.5 (0.5)	87.6 (0.5)	65.3 (1.7)	78.2 (1.3)
$BERT_{base}$	87.0 (1.5)	80.7 (1.7)	84.4 (0.7)	82.9 (0.7)	**90.7** (0.4)	**90.8** (0.4)	76.0 (0.7)	84.8 (0.5)
$BERT_{base}$-Sim	**88.3** (0.3)	**81.2** (1.1)	**85.1** (1.2)	**83.8** (1.1)	90.5 (0.4)	90.7 (0.4)	**76.2** (0.7)	**85.0** (0.6)

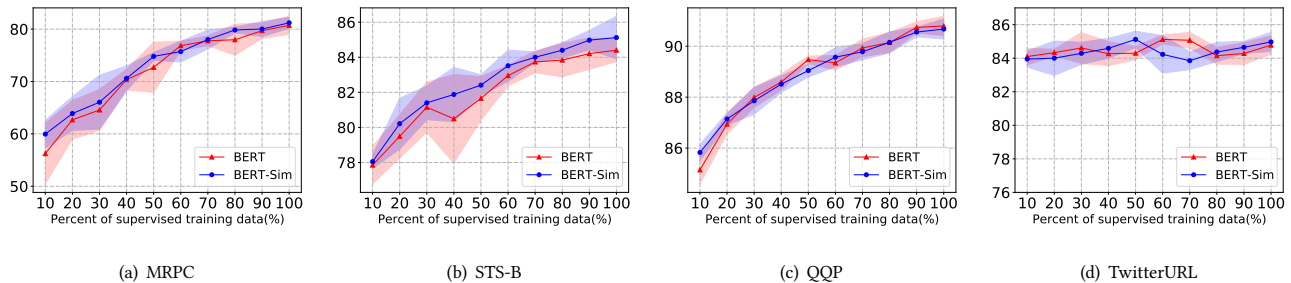| (a) MRPC	(b) STS-B	(c) QQP	(d) TwitterURL

Figure 7: Performance of BERT and BERT-Sim with different amounts of training data. X-axis: Percent of supervised training data. Y-axis: Macro-F1 for MRPC and Twitter-URL, Accuracy for QQP and Pearson Correlation for STS-B . The colored bands indicate ±1 standard deviation corresponding to different percentages of training data.

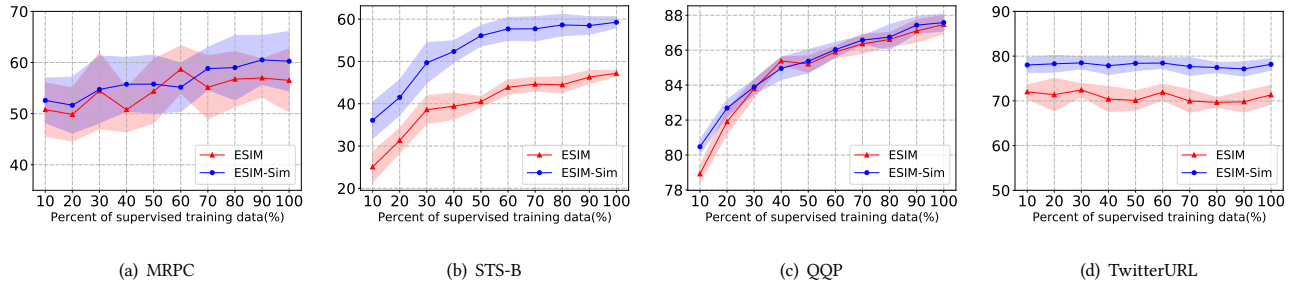| (a) MRPC	(b) STS-B	(c) QQP	(d) TwitterURL

Figure 8: Performance of ESIM and ESIM-Sim with different amounts of training data. X-axis: Percent of supervised training data. Y-axis: Macro-F1 for MRPC and Twitter-URL, Accuracy for QQP and Pearson Correlation for STS-B. The colored bands indicate ±1 standard deviation corresponding to different percentages of training data.

with our implementation. We use accuracy, F_1 score of the positive class, macro-F_1 score, Pearson correlation, Spearman correlation for evaluation on different datasets. These metrics follow previous literature [10, 40, 41]. Compared with the BERT model, ESIM model has a faster training speed and takes up less memory, so in Table 1 the results of ESIM model and ESIM-Sim model are the average after training 10 times. For BERT model, due to its slow training speed and large memory usage, the results are the average over 5 runs. In addition, our results have slight deviation from the results reported in the Google BERT paper [10]. We suspect the following potential reasons: (i) The training set and the validation set are split in different ways, for example QQP dataset. (ii) The random shuffling of the training set causes the deviation of the test result.

5.4 Learning Curve Analysis

At the same time, for each dataset, we randomly select from 10% to 100% data from training set as training data. For BERT and BERT-Sim, we have trained 5 times for each training scale of each dataset. We show the results in Figure 7. Similarly, for ESIM and ESIM-Sim, we have trained 10 runs; the results are shown in Figure 8.

To verify our hypothesis that word similarity knowledge is most needed in the first layer, we explored an alternative approach for comparison – to inject this knowledge into BERT's attention on *all* layers. We used the MRPC dataset in this analysis. The learning curve is shown in Figure 9.

Table 2: Example sentence pairs in the TwitterURL dataset. The four sentence pairs shares a common sentence.

Sentence pair	Label
s_a: Scientists create 3D-printed objects that can change shape after they are printed.	Paraphrase
s_b: Shape-shifting 3D printed objects are now a thing.	
s_a: Scientists create 3D-printed objects that can change shape after they are printed.	Paraphrase
s_b: 3D-printed objects change shape after printed an open door to new use cases.	
s_a: Scientists create 3D-printed objects that can change shape after they are printed.	Non-paraphrase
s_b: Imagine a drunk guy in the bar's bathroom , watching the urinal cakes morph.	
s_a: Scientists create 3D-printed objects that can change shape after they are printed.	Non-paraphrase
s_b: 4D printed materials are the future of architecture . So many great potential applications , esp. in green design.	
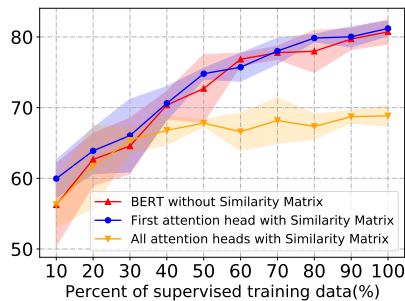
Figure 9: The effect of adding prior knowledge to different attention heads on the model performance on MRPC dataset. The y-axis represents Macro-F1.

5.5 Discussion

In Table 1, on the two relatively small datasets (MRPC and STS-B), we observe salient performance gain when knowledge is incorporated in the model (ESIM vs. ESIM-Sim; BERT vs. BERT-Sim). On the two relatively large datasets (QQP and TwitterURL), the performance gain is minimal if any. This is what we expected – prior knowledge is most beneficial when the training data is small. Comparing the performance gain within each model family, we observe that BERT gains less performance than ESIM when the word similarity knowledge is added. This echos with our finding in Section 3 that a pre-trained BERT already contains much knowledge about semantic textual matching tasks. In comparison, ESIM lacks the same level of knowledge even it is initialized with pre-trained word vectors.

Across the datasets, we observe that the gain in F_1 score is mainly caused by a salient increase in precision and a negligible decrease in recall. In other words, prior knowledge mainly helped BERT reduce false positives on the semantic text similarity task. This implies that the original BERT may not be able to tell the subtle semantic difference between a pair of related (but not synonymous) words, and therefore wrongly classified non-paraphrase sentences as paraphrases.

The learning curves in Figure 7 and 8 reveal a number of interesting patterns. Our proposed method for adding prior knowledge improves model performance almost consistently across all training data sizes. It is encouraging to see that the injected prior knowledge did not impose an unnecessarily strong inductive bias (in the sense

of bias-variance trade-off) to the models. In other words, the prior knowledge is consistent with the learning objective and beneficial to the models all along the learning process.

On MRPC, STS-B, and QQP datasets, prior knowledge provides the most salient performance gain happens when the training data is small, and the ESIM gains more than BERT. This is in agreement with our observation in Table 1. Also note that BERT trained on 10% to 20% of the training data can already outperform ESIM trained on full training data, and BERT gains performance at a faster rate than ESIM. This again highlights the rich knowledge and superior learning capability of BERT, compared to non-Transformer models. These findings suggest that it is most sensible to consider adding knowledge to BERT if the training data is scarce, i.e. on the order of hundreds of sentence pairs for a STS task.

In Figure 9, we compared the results between adding prior knowledge to the first layer of BERT's multi-head attention vs. adding prior knowledge to all layers, on the MRPC dataset. We observe that adding knowledge to all layers will impede the model from learning, creating an undesirable inductive bias. In contrast, adding knowledge to the first layer is more reasonable. This indicates that the broad/uniform attention on lower layers of BERT needs to be guided, while in higher layers attention heads have dedicated roles in collecting specific syntactic and semantic information.

What's peculiar is that on the TwitterURL dataset, we do not see similar results as the other datasets. We speculate this is mainly related to the form of the dataset. We show part of the TwitterURL dataset in detail in Table 2. In the TwitterURL dataset, it often happens that the same sentence s_a corresponds to many different s_b's, so when we increase the training data from 10% to 100%, a model does not see many completely new sentence pairs – it may have already seen at least one of the two sentences. This explains why the results obtained when the training data is 10% of the entire training set are similar to the results on 100% training data.

In Section 3, we use synonym knowledge to augment the MRPC dataset and train a BERT model that achieves 88.0 ± 0.4 % F1 score. In Table 1, BERT-Sim achieves 88.3 ± 0.3 % F1 score using the same form of knowledge. This suggests BERT-Sim performs at the same level as (or slightly better than) a data-augmented BERT. In Table 3, we show this type of comparison on all four datasets. BERT-sim slightly outperform data augmentation because it can inject word similarity knowledge – e.g. "happy" and "glad" are synonyms – into the model even if those words do not appear in the training data. Whereas for data augmentation, if no sentence in the training data contains the word "happy" or "glad", then the knowledge

that "happy" and "glad" are synonyms cannot be added by data augmentation, i.e., replacing synonyms in training data. Therefore the benefit of data augmentation depends on training data, while the benefit of knowledge injection does not.

Table 3: Performance in four datasets (%) using data augmentation and knowledge injection. We report for one metric per dataset as the other metric has similar trends. Numbers in parentheses are standard deviations.

Dataset / metric	Data-augmented BERT	BERT-Sim
MRPC / F1	88.0 (0.4)	88.3 (0.3)
STS-B / Pearson corr.	84.4 (0.7)	85.1 (1.2)
QQP / Accuracy	90.6 (0.3)	90.7 (0.4)
TwitterURL / F1	75.9 (0.7)	76.2 (0.7)

Note that because of extra training data, the data augmentation approach takes twice as much time to train than directly injecting knowledge into the model. Table 4 shows per-epoch training time on four datasets. This highlights an advantage of our proposed approach – it performs at least as well as the data augmentation approach with only half the training time.

Table 4: Per-epoch training time (in seconds) on the four datasets using data augmentation and knowledge injection.

Dataset	Data-augmented BERT	BERT-Sim
MRPC	73	36
STS-B	99	53
QQP	6372	3206
TwitterURL	775	338

6 CONCLUSION

In this paper, by analyzing what BERT has already known, what task-specific knowledge BERT needs and where it needs it, we proposed an effective and efficient method for injecting word similarity knowledge into BERT without adding new training task but directly guiding model's attention. This method is also applicable to non-Transformer deep model. Through experiments on different scale datasets on the semantic textual similarity (STS) task, we prove that the prior knowledge of word similarity is able to consistently improve the STS performance of BERT and the benefit is especially salient when training data is scarce.

ACKNOWLEDGMENTS

REFERENCES

[1] Islam Beltagy, Stephen Roller, Pengxiang Cheng, Katrin Erk, and Raymond J. Mooney. 2015. Representing Meaning with a Combination of Logical Form and Vectors. *CoRR* abs/1505.06816 (2015).

[2] Olivier Bodenreider. 2004. The unified medical language system (UMLS): integrating biomedical terminology. *Nucleic acids research* 32, suppl_1 (2004), D267–D270.

[3] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055* (2017).

[4] Jindong Chen, Yizhou Hu, Jingping Liu, Yanghua Xiao, and Haiyun Jiang. 2019. Deep short text classification with knowledge powered attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 6252–6259.

[5] Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for Natural Language Inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*. ACL, Vancouver, 1657–1668.

[6] Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Diana Inkpen, and Si Wei. 2018. Neural Natural Language Inference Models Enhanced with External Knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2406–2417.

[7] Yun-Nung Chen, Dilek Hakkani-Tur, Gokhan Tur, Asli Celikyilmaz, Jianfeng Gao, and Li Deng. 2016. Knowledge as a teacher: Knowledge-guided structural attention networks. *arXiv preprint arXiv:1609.03286* (2016).

[8] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What Does BERT Look At? An Analysis of BERT's Attention. In *BlackBoxNLP@ACL*.

[9] Dipanjan Das and Noah A Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. 468–476.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805* (2018).

[11] William B Dolan and Chris Brockett. 2005. Automatically Constructing a Corpus of Sentential Paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

[12] Allyson Ettinger. 2020. What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics* 8 (2020), 34–48.

[13] Samuel Fernando and Mark Stevenson. 2008. A semantic similarity approach to paraphrase detection. In *Proceedings of the 11th annual research colloquium of the UK special interest group for computational linguistics*. 45–52.

[14] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 8342–8360.

[15] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. DeBERTa: Decoding-enhanced BERT with Disentangled Attention. arXiv:2006.03654 [cs.CL]

[16] John Hewitt and Christopher D Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 4129–4138.

[17] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[18] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*. 2042–2050.

[19] Adrian Iftene and Alexandra Balahur. 2007. Hypothesis transformation and semantic variability rules used in recognizing textual entailment. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*. 125–130.

[20] Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. 2017. First quora dataset release: Question pairs. *URL https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs* (2017).

[21] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2019. SpanBERT: Improving Pre-training by Representing and Predicting Spans. *arXiv preprint arXiv:1907.10529* (2019).

[22] Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A Continuously Growing Dataset of Sentential Paraphrases. In *Proceedings of The 2017 Conference on Empirical Methods on Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 1235–1245.

[23] Wuwei Lan and Wei Xu. 2018. Neural Network Models for Paraphrase Identification, Semantic Textual Similarity, Natural Language Inference, and Question Answering. In *Proceedings of the 27th International Conference on Computational Linguistics*. Association for Computational Linguistics, 3890–3902.

[24] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *International Conference on Learning Representations*.

[25] Guanyu Li, Pengfei Zhang, and Caiyan Jia. 2018. Attention Boosted Sequential Inference Model. *CoRR* abs/1812.01840 (2018).

[26] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-BERT: Enabling Language Representation with Knowledge Graph. In *Proceedings of AAAI 2020*.

[27] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692* (2019).

[28] George A Miller. 1995. WordNet: a lexical database for English. *Commun. ACM* 38, 11 (1995), 39–41.

[29] Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A Decomposable Attention Model for Natural Language Inference. In *EMNLP*. The Association for Computational Linguistics, 2249–2255.

[30] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[31] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*. 2227–2237.

[32] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

[33] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how bert works. *arXiv preprint arXiv:2002.12327* (2020).

[34] Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2014. Back to Basics for Monolingual Alignment: Exploiting Word Similarity and Contextual Evidence. *Transactions of the Association for Computational Linguistics* 2 (2014), 219–230.

[35] Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2015. Dls@ cu: Sentence similarity from word alignment and semantic vector composition. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. 148–153.

[36] Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT Rediscovers the Classical NLP Pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 4593–4601.

[37] Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R Bowman, Dipanjan Das, et al. 2019. What do you learn from context? probing for sentence structure in contextualized

word representations. *arXiv preprint arXiv:1905.06316* (2019).

[38] Gaurav Singh Tomar, Thyago Duque, Oscar Täckström, Jakob Uszkoreit, and Dipanjan Das. 2017. Neural Paraphrase Identification of Questions with Noisy Pretraining. In *SWCN@EMNLP*. Association for Computational Linguistics, 142–147.

[39] 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 *Advances in neural information processing systems*. 5998–6008.

[40] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In the Proceedings of ICLR.

[41] Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 4144–4150.

[42] Zhibiao Wu and Martha Palmer. 1994. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*. 133–138.

[43] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. 2019. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848* (2019).

[44] Wei Xu, Alan Ritter, Chris Callison-Burch, William B Dolan, and Yangfeng Ji. 2014. Extracting lexically divergent paraphrases from Twitter. *Transactions of the Association for Computational Linguistics* 2 (2014), 435–448.

[45] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*. 5753–5763.

[46] Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics* 4 (2016), 259–272.

[47] Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced Language Representation with Informative Entities. In *Proceedings of ACL 2019*.

[48] Zhuosheng Zhang, Yuwei Wu, Hai Zhao, Zuchao Li, Shuailiang Zhang, Xi Zhou, and Xiang Zhou. 2020. Semantics-Aware BERT for Language Understanding. In *AAAI*. AAAI Press, 9628–9635.