Establishing Strong Baselines for the New Decade: Sequence Tagging, Syntactic and Semantic Parsing with BERT

Han He
Computer Science
Emory University
Atlanta GA 30322, USA
han.he@emory.edu

Jinho D. Choi
Computer Science
Emory University
Atlanta GA 30322, USA
jinho.choi@emory.edu

Abstract

This paper presents new state-of-the-art models for three tasks, part-of-speech tagging, syntactic parsing, and semantic parsing, using the cutting-edge contextualized embedding framework known as BERT. For each task, we first replicate and simplify the current state-of-the-art approach to enhance its model efficiency. We then evaluate our simplified approaches on those three tasks using token embeddings generated by BERT. 12 datasets in both English and Chinese are used for our experiments. The BERT models outperform the previously best-performing models by 2.5% on average (7.5% for the most significant case). All models and source codes are available in public so that researchers can improve upon and utilize them to establish strong baselines for the next decade. We also provide a dedicated error analysis and extensive dissections in https://arxiv.org/abs/1908.04943.

1 Introduction

It is no exaggeration to say that word embeddings trained by vector-based language models (Mikolov et al. 2013; Pennington, Socher, and Manning 2014; Bojanowski et al. 2017) have changed the game of NLP once and for all. These pre-trained word embeddings trained on large corpus improve downstream tasks by encoding rich word semantics into vector space. However, word senses are ignored in these earlier approaches such that a unique vector is assigned to each word, neglecting polysemy from the context.

Recently, contextualized embedding approaches emerge with advanced techniques to dynamically generate word embeddings from different contexts. To address polysemous words, Peters et al. (2018) introduce ELMo, which is a word-level Bi-LSTM language model. Akbik, Blythe, and Vollgraf (2018) apply a similar approach to the character-level, called Flair, while concatenating the hidden states corresponding to the first and the last characters of each word to build the embedding of that word. Apart from these unidirectional recurrent language models, Devlin et al. (2018) replace the transformer decoder from Radford et al. (2018) with a bidirectional transformer encoder, then train the BERT system on 3.3B word corpus. After scaling the model size to hundreds of millions parameters, BERT brings markedly huge improvement to a wide range of tasks without substantial task-specific modifications.

In this paper, we verify the effectiveness and conciseness of BERT by first generating token-level embeddings from it, then integrating them to task-oriented yet efficient model structures (Section 3). With careful investigation and engineering, our simplified models significantly outperform many of the previous state-of-the-art models, achieving the highest scores for 11 out of 12 datasets (Section 4).

To the best of our knowledge, it is the first work that tightly integrates BERT embeddings to these three downstream tasks and present such high performance. All our resources including the models and the source codes are publicly available.1

2 Related Work

Our work builds off recent work in representation learning, tagging and parsing. To learn contextualized representations, BERT (Devlin et al. 2018) employ masked LM to jointly condition on both left and right contexts, showing impressive improvement in various tasks. As a trend for tagging, fine grained features often result in better performance. These features include the morphological and contextual information from contextual string embeddings (Akbik, Blythe, and Vollgraf 2018), the representations from both string and token based character Bi-LSTM language models (Bohnet et al. 2018), and the ensemble of multilingual BERT and conventional representations (Heinzerling and Strube 2019). Among parsing community, graph-based parsers (Dozat and Manning 2017; Clark et al. 2018; Ma et al. 2018) resurge due to GPU parallelization. Recently, (Zhou and Zhao 2019) achieved impressive results by jointly learning constituency and dependency parsing with BERT.

3 Approach

3.1 Token-level Embeddings with BERT

BERT splits each token into subwords using WordPiece (Wu et al. 2016), which do not necessarily reflect any morphology in linguistics. For example, *Rainwater* gets split into

1https://github.com/emorynlp/bert-2019
While Devlin et al. (2018) report that adding just an additional
Vollgraf 2018), is used to establish a baseline for English.

Encoder

where

Alternatively, BERT embeddings can be concatenated with

word does not necessarily convey the key meaning of the
token. Hence, another way is to take the average embedding
of the subwords.

Table 1: Results from the PSD semantic parsing task (§4.3)
using the last and average embedding methods.

Table 1 shows results from a semantic parsing task, PSD
(Section 4.3), using the last and average embedding methods
with BERT_base and BERT_large models.2 The average method
is chosen for all our experiments since it gives a marginal
advantage to the out-of-domain dataset.

3.2 Input Embeddings with BERT

While Devlin et al. (2018) report that adding just an additional
output layer to the BERT encoder can build powerful models
in a wide range of tasks, its computational cost is too high.
Thus, we separate the BERT architecture from downstream
models, and feed pre-generated BERT embeddings, eBERT,
as input to task-specific encoders:

\[ F^1 = \text{Encoder} (X \oplus e_{\text{BERT}}) \]

Alternatively, BERT embeddings can be concatenated with
the output of a certain hidden layer:

\[ F^h = \text{Encoder}_{[h]} (\text{Encoder}_{[h]} (X \oplus e_{\text{BERT}}) \]

where Encoder_{[h]} denotes for encoder layers from 1 to h,
and Encoder_{[h]} denotes for layers from h to the last one.
Table 2 shows results from the PSD semantic parsing task
(Section 4.3) using the average method from Section 3.1. F^h
shows a slight advantage for both BERT_base and BERT_large
over F^1; thus, it is chosen for all our experiments.

3.3 Bi-LSTM-CRF for Tagging

For sequence tagging, the Bi-LSTM-CRF (Huang, Xu, and
Yu 2015) with the Flair embeddings (Akbik, Blythe, and
Vollgraf 2018), is used to establish a baseline for English.

\[^2\text{BERT base uses 12 layers, 768 hidden cells, 12 attention heads,}
\text{and 110M parameters, while BERT large uses 24 layers, 1024 hidden}
\text{cells, 16 attention heads, and 340M parameters. Both models are}
\text{uncased, since they are reported to achieve high scores for all tasks}
\text{except for NER (Devlin et al. 2018).} \]

Table 2: Results from the PSD semantic parsing task (Sec-

3.4 Biaffine Attention for Syntactic Parsing

A simplified variant of the biaffine parser (Dozat and
Manning 2017) is used for syntactic parsing (Figure 2).
Compared to the original version, the trainable word
embeddings are removed and lemmas are used instead
of forms to retrieve pre-trained embeddings in our ver-

gion, leading to less complexity yet better generalization.
Given the i’th token w_i, the feature vector is created by
concatenating its pre-trained lemma embedding e_i^{\text{LEM}}, POS
embedding e_i^{\text{POS}} learned during training and the representa-

e_i^{\text{Flair}} is then concatenated with a pre-trained token embedding
of w and fed into the Bi-LSTM-CRF. In our approach, we
present two models, one substituting the Flair and pre-trained
embeddings with BERT, and the other concatenating BERT
to the other embeddings. Note that variational dropout is not
used in our approach to reduce complexity.

Given a token w in a sequence where c_i and c_j are the start-
ing and ending characters of w (i and j are the character
offsets; i \leq j), the Flair embedding of w is generated by
concatenating two hidden states of c_{j+1} from the forward
LSTM and c_{i-1} from the backward LSTM (Figure 1):

\[ e_i^{\text{Flair}} = h^f(c_{j+1}) \oplus h^b(c_{i-1}) \]

\[ r^f = \text{LSTM}^\text{forward} (e_i^{\text{LEM}} \oplus e_i^{\text{POS}} \oplus e_i^{\text{BERT}}) \]

Two multi-layer perceptrons (MLP) are then used to extract
features for w_i being a head h_{arc} and a dependent h_{arc-d}, and
two additional MLP are used to extract h_{rel} and h_{rel-d} for
labeled dependency parsing:
which combines MLPs for arc labels are predicted by another biaffine classifier (the optimal tree based on the scores in $S$). During training, softmax cross-entropy is used to optimize $S$.

$H_{\text{arc}}^i = \text{MLP}(\text{arc-h}) \left( r_i^h \oplus r_i^h \right) \in \mathbb{R}^{k \times 1}$

$H_{\text{arc}}^i = \text{MLP}(\text{arc-d}) \left( r_i^d \oplus r_i^d \right) \in \mathbb{R}^{k \times 1}$

$H_{\text{rel}}^i = \text{MLP}(\text{rel-h}) \left( r_i^h \oplus r_i^h \right) \in \mathbb{R}^{l \times 1}$

$H_{\text{rel}}^i = \text{MLP}(\text{rel-d}) \left( r_i^d \oplus r_i^d \right) \in \mathbb{R}^{l \times 1}$

$H_{\text{arc}}^{i..n}$ are stacked into a matrix $H_{\text{arc}}$ with a bias for the prior probability of each token being a head, and $H_{\text{arc}}^{i..n}$ are stacked into another matrix $H_{\text{arc}}$ as follows ($n$: # of tokens, $U(\text{arc}) \in \mathbb{R}^{k \times (k+1)}$):

$H_{\text{arc}} = \left( h_{1\text{arc}}, \ldots, h_{n\text{arc}} \right) \in \mathbb{R}^{k \times n}$

$S_{\text{arc}} = H_{\text{arc}}^{\top} \cdot U(\text{arc}) \cdot H_{\text{arc}} \in \mathbb{R}^{n \times n}$

$S_{\text{arc}}$ is called a bilinear classifier that predicts head words. Arc labels are predicted by another bilinear classifier $S_{\text{arc}}$, which combines $m$ bilinear classifiers for multi-classification ($m$: # of labels, $U(\text{rel}) \in \mathbb{R}^{l \times (l+1)}$, $V(\text{rel}) \in \mathbb{R}^{2 \times (l+1) \times m}$):

$H_{\text{rel}}^{\text{rel-h}} = \left( h_{1\text{rel-h}}, \ldots, h_{n\text{rel-h}} \right) \in \mathbb{R}^{l \times n}$

$H_{\text{rel}}^{\text{rel-d}} = \left( h_{1\text{rel-d}}, \ldots, h_{n\text{rel-d}} \right) \oplus 1 \in \mathbb{R}^{l \times (l+1) \times n}$

$U_{\text{rel}} = H_{\text{rel}}^{\text{rel-h}} \cdot V(\text{rel}) \cdot H_{\text{rel}}^{\text{rel-d}} \in \mathbb{R}^{n \times n}$

$S_{\text{rel}} = \left( U_{1\text{rel}}, \ldots, U_{m\text{rel}} \right)$

$S_{\text{rel}} = \left( H_{\text{rel}}^{\text{rel-h}} \oplus H_{\text{rel}}^{\text{rel-d}} \right)^{\top} \cdot V(\text{rel}) \in \mathbb{R}^{m \times n \times n}$

During training, softmax cross-entropy is used to optimize $S_{\text{arc}}$ and $S_{\text{rel}}$. Note that for the optimization of $S_{\text{rel}}$, gold heads are used instead of predicted ones. During decoding, a maximum spanning tree algorithm is adopted for searching the optimal tree based on the scores in $S_{\text{arc}}$.

### 3.5 Biaffine Attention for Semantic Parsing

Dozat and Manning (2018) adapted their original biaffine parser to generate dependency graphs for semantic parsing, where each token can have zero to many heads. Since the tree structure is no longer guaranteed, sigmoid cross-entropy is used instead so that independent binary predictions can be made for every token to be considered a head of any other token. The label predictions are made as outputting the labels with the highest scores in $S_{\text{rel}}$ once arc predictions are made, as illustrated in Figure 2.

This updated implementation is further simplified in our approach by removing the trainable word embeddings, the character-level feature detector, and their corresponding linear transformers. Moreover, instead of using the interpolation between the head and label losses, equal weights are applied to both losses, reducing hyperparameters to tune.

### 4 Experiments

Three sets of experiments are conducted to evaluate the impact of our approaches using BERT (Section 3). For sequence tagging (Section 4.1), part-of-speech tagging is chosen where each token gets assigned with a fine-grained POS tag. For syntactic parsing (Section 4.2), dependency parsing is chosen where each token finds exactly one head, generating a tree per sentence. For semantic parsing (Section 4.3), semantic dependency parsing is chosen where each token finds zero to many heads, generating a graph per sentence. Every task is tested on both English and Chinese to ensure robustness across languages. Standard datasets are adapted to all experiments for fair comparisons to many previous approaches. All our models are experimented three times and average scores with standard deviations are reported.

### 4.1 Sequence Tagging

For part-of-speech tagging, the Wall Street Journal corpus from the Penn Treebank 3 (Marcus, Marcinkiewicz, and San-
Table 4 shows parsing results on the test sets. The same datasets for POS tagging are also used for syntactic parsing.

4.2 Syntactic Parsing

Table 4 shows parsing results on the test sets. The same datasets for POS tagging are also used for syntactic parsing.

Our simplified version of the biaffine parser (Section 3.4) is used for baseline, where GloVe and FastText embeddings are used for English and Chinese, respectively. The baseline model gives a comparable result to the original model (Dozat and Manning 2017) for English, yet shows a notably better result for Chinese, which can be due to higher quality embeddings from FastText. \BERT substitutes the pre-trained embeddings with BERT and +BERT adds BERT embeddings to the baseline. Moreover, BERT’s uncased base model is used for English.

Between \BERT and +BERT, no significant difference is found, implying that those pre-trained embeddings are not so useful when coupled with BERT. All BERT models show significant improvement over the baselines for both languages, and outperform the previous state-of-the-art approaches using cross-view training (Clark et al. 2018) and stack-pointer networks (Ma et al. 2018) by 0.29% and 3% in LAS for English and Chinese, respectively. Considering the simplicity of our +BERT models, these results are remarkable.

4.3 Semantic Parsing

The English dataset from the SemEval 2015 Task 18: Broad-Coverage Semantic Dependency Parsing (Oepen et al. 2015) and the Chinese dataset from the SemEval 2016 Task 9: Chinese Semantic Dependency Parsing (Che et al. 2016) are used for semantic dependency parsing.

Table 5 shows the English results on the test sets. The same datasets for POS tagging are also used for semantic dependency parsing.

Our simplified version of the biaffine parser (Section 3.4) is used for baseline, where GloVe and FastText embeddings are used for English and Chinese, respectively. The baseline model gives a comparable result to the original model (Dozat and Manning 2017) for English, yet shows a notably better result for Chinese, which can be due to higher quality embeddings from FastText. \BERT substitutes the pre-trained embeddings with BERT and +BERT adds BERT embeddings to the baseline. Moreover, BERT’s uncased base model is used for English.

4.3.1 BERT Substitutes

For Chinese, both the \BERT and +BERT models perform slightly lower accuracy than the original model (Akbik, Blythe, and Vollgraf 2018) due to the lack of variational dropout. However, interesting trends are found in the OOV test. The +BERT model shows good improvement over the baseline.

For Chinese, the Bi-LSTM-CRF model with FastText embeddings is used for baseline (Sec. 3.3). \BERT substitutes FastText embeddings with BERT and +BERT that adds BERT embeddings to the baseline show progressive improvement over the prior model for both the ALL and OOV tests. +BERT gives an accuracy that is 1.25% higher than the previous state-of-the-art using joint-learning between tagging and parsing (Wang and Xue 2014).

Table 5 shows the English results on the test sets. The same datasets for POS tagging are also used for semantic dependency parsing.

Our simplified version of the biaffine parser (Section 3.4) is used for baseline, where GloVe and FastText embeddings are used for English and Chinese, respectively. The baseline model gives a comparable result to the original model (Dozat and Manning 2017) for English, yet shows a notably better result for Chinese, which can be due to higher quality embeddings from FastText. \BERT substitutes the pre-trained embeddings with BERT and +BERT adds BERT embeddings to the baseline. Moreover, BERT’s uncased base model is used for English.

Between \BERT and +BERT, no significant difference is found, implying that those pre-trained embeddings are not so useful when coupled with BERT. All BERT models show significant improvement over the baselines for both languages, and outperform the previous state-of-the-art approaches using cross-view training (Clark et al. 2018) and stack-pointer networks (Ma et al. 2018) by 0.29% and 3% in LAS for English and Chinese, respectively. Considering the simplicity of our +BERT models, these results are remarkable.

4.3 Semantic Parsing

The English dataset from the SemEval 2015 Task 18: Broad-Coverage Semantic Dependency Parsing (Oepen et al. 2015) and the Chinese dataset from the SemEval 2016 Task 9: Chinese Semantic Dependency Parsing (Che et al. 2016) are used for semantic dependency parsing.

Table 4 shows parsing results on the test sets. The same datasets for POS tagging are also used for syntactic parsing.

Our simplified version of the biaffine parser (Section 3.4) is used for baseline, where GloVe and FastText embeddings are used for English and Chinese, respectively. The baseline model gives a comparable result to the original model (Dozat and Manning 2017) for English, yet shows a notably better result for Chinese, which can be due to higher quality embeddings from FastText. \BERT substitutes the pre-trained embeddings with BERT and +BERT adds BERT embeddings to the baseline. Moreover, BERT’s uncased base model is used for English.

Between \BERT and +BERT, no significant difference is found, implying that those pre-trained embeddings are not so useful when coupled with BERT. All BERT models show significant improvement over the baselines for both languages, and outperform the previous state-of-the-art approaches using cross-view training (Clark et al. 2018) and stack-pointer networks (Ma et al. 2018) by 0.29% and 3% in LAS for English and Chinese, respectively. Considering the simplicity of our +BERT models, these results are remarkable.
embeddings are complementary to those pre-trained embeddings, and surpasses the previous state-of-the-art scores by 3% and 2% for ID and OOD, respectively.

<table>
<thead>
<tr>
<th></th>
<th>DM</th>
<th>PAS</th>
<th>PSD</th>
<th>AVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Du et al. (2015)</td>
<td>89.1</td>
<td>91.3</td>
<td>75.7</td>
<td>85.3</td>
</tr>
<tr>
<td>Almeida and Martins (2015)</td>
<td>89.4</td>
<td>91.7</td>
<td>77.6</td>
<td>86.2</td>
</tr>
<tr>
<td>Wang et al. (2018)</td>
<td>90.3</td>
<td>91.7</td>
<td>78.6</td>
<td>86.9</td>
</tr>
<tr>
<td>Peng, Thomson, and Smith (2017)</td>
<td>90.4</td>
<td>92.7</td>
<td>78.5</td>
<td>87.2</td>
</tr>
<tr>
<td>Dozat and Manning (2018)</td>
<td>93.7</td>
<td>93.9</td>
<td>81.0</td>
<td>89.5</td>
</tr>
<tr>
<td>Che et al. (2019)</td>
<td>92.9</td>
<td>94.4</td>
<td>81.6</td>
<td>89.6</td>
</tr>
<tr>
<td>Baseline</td>
<td>92.48</td>
<td>94.56</td>
<td>85.00</td>
<td>90.68</td>
</tr>
<tr>
<td>Baseline \ BERT</td>
<td>94.36</td>
<td>96.03</td>
<td>86.59</td>
<td>92.33</td>
</tr>
<tr>
<td>Baseline + BERT</td>
<td>94.57</td>
<td>96.13</td>
<td>86.80</td>
<td>92.50</td>
</tr>
</tbody>
</table>

(a) Results from the in-domain (ID) test sets.

<table>
<thead>
<tr>
<th></th>
<th>DM</th>
<th>PAS</th>
<th>PSD</th>
<th>AVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Du et al. (2015)</td>
<td>81.8</td>
<td>87.2</td>
<td>73.3</td>
<td>80.8</td>
</tr>
<tr>
<td>Almeida and Martins (2015)</td>
<td>83.8</td>
<td>87.6</td>
<td>76.2</td>
<td>82.5</td>
</tr>
<tr>
<td>Wang et al. (2018)</td>
<td>84.9</td>
<td>87.6</td>
<td>75.9</td>
<td>82.8</td>
</tr>
<tr>
<td>Peng, Thomson, and Smith (2017)</td>
<td>85.3</td>
<td>89.0</td>
<td>76.4</td>
<td>83.6</td>
</tr>
<tr>
<td>Dozat and Manning (2018)</td>
<td>88.9</td>
<td>90.6</td>
<td>79.4</td>
<td>86.3</td>
</tr>
<tr>
<td>Che et al. (2019)</td>
<td>89.2</td>
<td>92.4</td>
<td>81.0</td>
<td>87.5</td>
</tr>
<tr>
<td>Baseline</td>
<td>86.98</td>
<td>91.35</td>
<td>77.28</td>
<td>85.34</td>
</tr>
<tr>
<td>Baseline \ BERT</td>
<td>90.49</td>
<td>94.31</td>
<td>79.31</td>
<td>88.07</td>
</tr>
<tr>
<td>Baseline + BERT</td>
<td>90.86</td>
<td>94.38</td>
<td>79.48</td>
<td>88.21</td>
</tr>
</tbody>
</table>

(b) Results from the out-of-domain (OOD) test sets.

Table 5: Test results for semantic dependency parsing in English; labeled dependency F1 scores are used as the evaluation metrics. DM: DELPH-IN dependencies, PAS: Enju dependencies, PSD: Prague dependencies, AVG: macro-average of (DM, PAS, PSD).

Table 6 shows the Chinese results on the test sets. No significant difference is found between \ BERT and +BERT. +BERT significantly outperforms the previous state-of-the-art by 4% and 7.5% in LF for NEWS and TEXT, which confirms that BERT embeddings are very effective for semantic dependency parsing in both English and Chinese.

<table>
<thead>
<tr>
<th></th>
<th>NEWS</th>
<th>TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UF</td>
<td>LF</td>
</tr>
<tr>
<td>Artsymenia, Dounar, and Yermakovich (2016)</td>
<td>77.04</td>
<td>69.00</td>
</tr>
<tr>
<td>Wang et al. (2018)</td>
<td>81.14</td>
<td>63.30</td>
</tr>
<tr>
<td>Baseline</td>
<td>80.51</td>
<td>68.90</td>
</tr>
<tr>
<td>Baseline \ BERT</td>
<td>82.91</td>
<td>67.17</td>
</tr>
<tr>
<td>Baseline + BERT</td>
<td>82.92</td>
<td>67.27</td>
</tr>
</tbody>
</table>

Table 6: Test results for semantic dependency parsing in Chinese, where unlabeled and labeled dependency F1 scores (UF and LF) are used as the evaluation metrics. NEWS: newswire, TEXT: textbook.

5 Conclusion

In this paper, we describe our methods of exploiting BERT as token-level embeddings for tagging and parsing tasks. Our experiments empirically show that tagging and parsing can be tackled using much simpler models without losing accuracy. Out of 12 datasets, our approaches with BERT have established new state-of-the-art for 11 of them. As the first work of employing BERT with syntactic and semantic parsing, our approach is much simpler yet more accurate than the previous state-of-the-art.

Acknowledgments

We gratefully acknowledge the support of the AWS Machine Learning Research Awards (MLRA) and the GluonNLP team (Guo et al. 2019).

References


