Leveraging Static and Contextualized Embeddings for Word Alignments

Anonymous

Abstract

Word alignments are useful for tasks like statistical and neural machine translation (NMT) and annotation projection. Statistical word aligners perform well, as do methods that extract alignments jointly with translations in NMT. However, most approaches require parallel training data and quality decreases as less training data is available. We propose word alignment methods that require little or no parallel data. The key idea is to leverage multilingual word embeddings – both static and contextualized – for word alignment. Our multilingual embeddings are created from monolingual data only without relying on any parallel data or dictionaries. We find that traditional statistical aligners are outperformed by contextualized embeddings – even in scenarios with abundant parallel data. For example, for a set of 100k parallel sentences, contextualized embeddings achieve a word alignment $F_1$ that is more than 5% higher (absolute) than eflomal.

1 Introduction

Word alignment is essential for statistical machine translation and useful in NMT, e.g., for imposing priors on attention matrices (Liu et al., 2016; Alkhouli and Ney, 2017; Alkhouli et al., 2018) or for decoding (Alkhouli et al., 2016; Press and Smith, 2018). Further, word alignments have been successfully used in a range of tasks such as typological analysis (Lewis and Xia, 2008; Östling, 2015), annotation projection (Yarowsky et al., 2001; Padó and Lapata, 2009) and creating multilingual embeddings (Guo et al., 2016).

Statistical word aligners such as the IBM models (Brown et al., 1993) and their successors (e.g., fastalign (Dyer et al., 2013), GIZA++ (Och and Ney, 2003), eflomal (Östling and Tiedemann, 2016)) are widely used for alignment. With the rise of NMT (Bahdanau et al., 2014), attempts have been made to interpret attention matrices as soft word alignments (Koehn and Knowles, 2017; Ghader and Monz, 2017). Several methods create alignments from attention matrices (Peter et al., 2017; Li et al., 2018; Zenkel et al., 2019) or pursue a multitask approach for alignment and translation (Chen et al., 2016; Garg et al., 2019). However, most systems require parallel data and their performance deteriorates when parallel text is scarce (cf. Tables 1–2 in (Och and Ney, 2003)).

Recent unsupervised multilingual embedding algorithms that use only monolingual data provide high quality static and contextualized embeddings (Conneau et al., 2018; Devlin et al., 2019; Pires et al., 2019). Our key idea is to leverage these embeddings for word alignments – without relying on parallel data. Requiring no or little parallel data is advantageous in many scenarios, e.g., in the low-resource case and in domain-specific settings without parallel data. A lack of parallel data cannot be easily remedied: mining parallel sentences is possible (cf. (Schwenk et al., 2019)) but assumes that monolingual corpora contain parallel sentences. Contributions: (1) We propose two new alignment methods based on the matrix of embedding similarities. (2) We propose two post-processing algorithms that handle null words and integrate positional information. (3) We show that word alignments obtained from multilingual BERT outperform strong statistical word aligners like eflomal. (4) We investigate the differences between word and subword processing for alignments and find subword processing to be preferable. Upon acceptance we will publish the source code.

2 Methods

2.1 Alignments from Similarity Matrices

Consider parallel sentences $s^{(e)}, s^{(f)}$, with lengths $l_e, l_f$ in languages $e, f$. Assume we have access
to some embedding function $\mathcal{E}$ that assigns each word in a sentence a $d$-dimensional vector, i.e., $\mathcal{E}(s^{(k)}) \in \mathbb{R}^{d 	imes d}$ for $k \in \{e, f\}$. Let $\mathcal{E}(s^{(k)})_i$ denote the vector of the $i$-th word in sentence $s^{(k)}$. We define the similarity matrix as the matrix $S \in [0, 1]^{l_e \times l_f}$ induced by the embeddings where $S_{ij} := \text{sim}(\mathcal{E}(s^{(e)}_i), \mathcal{E}(s^{(f)}_j))$ is some normalized measure of similarity, e.g., cosine-similarity normalized to be between 0 and 1. We now introduce methods for extracting alignments from $S$, i.e., obtaining a binary matrix $A \in \{0, 1\}^{l_e \times l_f}$. 

**Argmax.** A simple baseline is to align each word in sentence $s^{(e)}$ with the most similar word in $s^{(f)}$ and vice versa. That is, we define $A_{ij} = 1$ if and only if

$$(i = \arg \max_l S_{i,j}) \land (j = \arg \max_l S_{i,l})$$

and $A_{ij} = 0$ else. Similar methods have been applied to Dice coefficients (cf. (Och and Ney, 2003)) and attention matrices (Garg et al., 2019).

**Match.** Argmax finds a local, not a global optimum. To address this, we frame alignment as an assignment problem: we search for a maximum-weight maximal matching (cf. (Ramshaw and Tarjan, 2012)) in the bipartite weighted graph induced by the similarity matrix. This optimization problem is given by

$$A^* = \arg \max_{A \in \{0, 1\}^{l_e \times l_f}} \sum_{i=1}^{l_e} \sum_{j=1}^{l_f} A_{ij} S_{ij}$$

subject to $A$ being a valid maximal matching (i.e., every word in the shorter sentence is aligned). There are known algorithms to solve the above problem in polynomial time (cf. Kuhn (1955)). Note that alignments generated with the matching method are inherently bidirectional and do not require any symmetrization as post-processing.

**Distortion Correction [Dist].** Distortion, as introduced in IBM Model 2, is essential for alignments based on non-contextualized embeddings since the similarity of two words is solely based on their surface form, independent of position. To penalize high distortion, we multiply the similarity matrix $S$ componentwise with

$$P_{i,j} = 1 - \kappa \left(\frac{i/l_e - j/l_f}{2}\right)^2,$$

where $\kappa$ is a hyperparameter to scale the matrix between $[(1 - \kappa), 1]$. We use $\kappa = 0.5$. We can interpret this as imposing a locality-preserving prior:

given a choice, a word should be aligned to a word with a similar relative position $((i/l_e - j/l_f)^2$ close to 0) rather than a more distant word (large $(i/l_e - j/l_f)^2$).

**Null.** Null words model untranslated words and are an important part of alignment models. We remove alignment edges when the normalized entropy of the similarity distribution is above a threshold $\tau$, a hyperparameter. Intuitively if a word is not similar to any of the words in the target sentence, we do not align it. That is, we set $A_{ij} = 0$ if and only if

$$\min(-\frac{\sum_{k=1}^{l_e} S_{ik} \log S_{ik}}{\log l_e} - \frac{\sum_{k=1}^{l_f} S_{kj} \log S_{kj}}{\log l_f}) > \tau.$$  

### 2.2 Baselines

**fast-align (Dyer et al., 2013)**, a version of IBM Model 2, is popular because of its speed and high quality. efmaral1 (based on efmaral *Östling and Tiedemann (2016)*), a Bayesian model with Markov Chain Monte Carlo inference, is claimed to outperform fast-align on speed and quality (*Östling and Tiedemann, 2016*). GIZA++ (Och and Ney, 2003) is a standard word aligner.

### 2.3 Symmetrization

Traditional word alignment models create forward and backward alignments and then symmetrize them (Koehn, 2010). We compared grow-diagonal (GDFA) and intersection and found them to perform comparably.4 We use G DFA throughout the paper.

### 2.4 Word vs. Subword Level

We investigate both subword segmentations such as BPE/wordpiece (Sennrich et al., 2016) (which are widely used for contextualized language models) and word-level representations for word alignment. For the word level, we use NLTK tokenizer (Bird et al., 2009). For the subword level, we use multilingual BERT’s vocabulary and BERT’s tokenizer. As gold standards are all word-level, we evaluate on the word level. To convert subword to word alignments we apply the rule: “two words are aligned if any of their subwords are aligned.” Thus

---

1See supplementary for different values.

2For values of this hyperparameter see supplementary.

3https://github.com/google-research/bert

4https://github.com/google-research/bert

5For a comparison see supplementary.

6https://github.com/robustostling/efmaral

7https://github.com/google-research/bert
a single word can be aligned with multiple other words.

2.5 Embedding Learning

**Static.** For monolingual embeddings, we use fastText (Bojanowski et al., 2017) and then VecMap (Artetxe et al., 2018) to map the embeddings into a common multilingual space. Note that this algorithm works without any crosslingual supervision (i.e., multilingual dictionaries). We use the same procedure for word and subword levels. We use the label fastText to refer to these embeddings as well as to the word alignments induced by them.

**Contextualized.** We use the multilingual BERT model (mBERT). It is pretrained on the 104 largest Wikipedia languages. This model only provides embeddings on the subword level. To obtain a word embedding, we simply average the vectors of its subwords. We consider word representations from all 12 layers as well as the concatenation of all layers. Note that the model is not finetuned. We denote this method as mBERT[i] (when using embeddings from the i-th layer, where 0 means using the non-contextualized embedding layer) and mBERT[conc] (for concatenation).

3 Experiments

Our test data are three language pairs in different domains. We use Europarl gold alignments\(^7\) for English-German, Bible gold alignments (Melamed, 1998) for English-French and gold alignments by Tavakoli and Faili (2014) for English-Persian (domain: books).

We select additional parallel training data that is consistent with the target domain where available: Europarl (Koehn, 2005) for English-German and Parallel Bible Corpus (PBC) (Mayer and Cysouw, 2014) for English-French and English-Persian. For fast-align, GIZA++ and eflomal we add 10,000 parallel sentences (simulating a mid-resource setting) to the gold standard as training data. We show the effect of adding more or less training data in Figure 1. Since mBERT is pretrained on Wikipedia, we train fastText embeddings on Wikipedia as well. For hyperparameters of all models see supplementary.

Our evaluation measures are precision, recall, \(F_1\) and alignment error rate (AER).\(^8\)

---

**Figure 1:** Learning curves of fast-align/eflomal vs. embedding-based alignments. Results shown are \(F_1\) on word and subword level for ENG-DEU.

<table>
<thead>
<tr>
<th>Method</th>
<th>ENG-DEU Prec.</th>
<th>Rec.</th>
<th>(F_1)</th>
<th>AER</th>
<th>ENG-FRA Prec.</th>
<th>Rec.</th>
<th>(F_1)</th>
<th>AER</th>
<th>ENG-FAS Prec.</th>
<th>Rec.</th>
<th>(F_1)</th>
<th>AER</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast-align</td>
<td>57.60</td>
<td>60.40</td>
<td>60.34</td>
<td>32</td>
<td>48.44</td>
<td>46.54</td>
<td>48.44</td>
<td>54</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GIZA++</td>
<td>60.60</td>
<td>60.40</td>
<td>60.34</td>
<td>32</td>
<td>50.48</td>
<td>49.51</td>
<td>50.48</td>
<td>51</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>eflomal</td>
<td>77.68</td>
<td>72.28</td>
<td>71.96</td>
<td>16</td>
<td>64.55</td>
<td>59.41</td>
<td>64.55</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fastText(Argmax+Dist)</td>
<td>84.54</td>
<td>65.34</td>
<td>73.39</td>
<td>17</td>
<td>73.39</td>
<td>51.49</td>
<td>73.39</td>
<td>49</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mBERT<a href="Argmax">8</a></td>
<td>92.69</td>
<td>79.21</td>
<td>86.73</td>
<td>17</td>
<td>86.73</td>
<td>67.33</td>
<td>86.73</td>
<td>33</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fast-align</td>
<td>57.65</td>
<td>61.59</td>
<td>60.85</td>
<td>32</td>
<td>47.44</td>
<td>45.55</td>
<td>47.44</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GIZA++</td>
<td>69.67</td>
<td>68.32</td>
<td>68.70</td>
<td>32</td>
<td>55.47</td>
<td>50.50</td>
<td>55.47</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>eflomal</td>
<td>75.73</td>
<td>74.26</td>
<td>74.48</td>
<td>16</td>
<td>63.67</td>
<td>60.41</td>
<td>63.67</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fastText(Argmax+Dist)</td>
<td>79.52</td>
<td>62.37</td>
<td>76.50</td>
<td>38</td>
<td>28.08</td>
<td>12.88</td>
<td>28.08</td>
<td>88</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mBERT<a href="Argmax">8</a></td>
<td>91.73</td>
<td>81.19</td>
<td>94.74</td>
<td>16</td>
<td>85.56</td>
<td>67.33</td>
<td>85.56</td>
<td>33</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Comparison of word and subword levels. Best overall result per column in bold.

4 Results

**Overall.** Table 1 shows that mBERT performs consistently best. Eflomal has high precision and is on par for ENG-FRA. Surprisingly, fastText outperforms fast-align in two out of three languages (e.g., \(F_1\) 65 vs. 61 for ENG-DEU) despite not having access to parallel data. Among the statistical baselines, eflomal outperforms GIZA++, while GIZA++ is better than fast-align, as expected.

**Parallel Data.** Figure 1 shows that fast-align and eflomal get better with more training data with eflomal outperforming fast-align, as expected. However, even with \(10^6\) parallel sentences mBERT outperforms both statistical baselines. fastText becomes competitive for fewer than 1000 parallel sentences. The main takeaway is that mBERT-based alignments, a method that does not need any parallel training data, outperform state-of-the-art aligners, even in the high resource case.

**Word vs. Subword.** In Table 1 subword processing mainly benefits fast-align, GIZA++ and eflomal (except for ENG-FRA). fastText is harmed by subword processing. We use VecMap to match (sub)word distributions across languages. We hypothesize that it is harder to match subword than...
Figure 2: Word alignment performance across all layers of mBERT. Results are $F_1$ on subword level with Argmax and no post-processing applied.

Table 2: Comparison of methods for inducing alignments from similarity matrices. All results are subword-level. Best result per embedding type across columns in bold.

<table>
<thead>
<tr>
<th>Emb</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENG-DEU</td>
</tr>
<tr>
<td>ENG-DEU</td>
</tr>
</tbody>
</table>

Table 4 compares alignment and post-processing methods. Argmax generally yields higher precision whereas Match has higher recall. For fastText using Argmax with Dist yields best $F_1$ on two languages. Adding a distortion prior boosts performance for static embeddings, e.g., from .46 to .61 for ENG-FRA $F_1$. Null-word processing increases precision, e.g., from .91 to .96 for ENG-DEU, but does not increase $F_1$. For mBERT Argmax performs best in two out of three language pairs. Dist has little and sometimes harmful effects on mBERT indicating that mBERT’s contextualized representations already match well across languages.

mBERT Layers. Figure 2 shows a parabolic trend with layer 8 of mBERT yielding the best performance. This is consistent with other work (Voita et al., 2019; Tenney et al., 2019): in the first layers the contextualization is too weak for high-quality alignments while last layers are too specialized on the pretraining task (masked language modeling).

5 Related Work

Brown et al. (1993) introduced the IBM models, the best known statistical word aligners. More recent aligners, often based on IBM models, include fast-align (Dyer et al., 2013), GIZA++ (Och and Ney, 2003) and eflomal (Östling and Tiedemann, 2016). All of these models are trained on parallel text. Our method instead aligns based on embeddings that are induced from monolingual data only.

Prior work on using learned representations for alignment includes (Smadja et al., 1996; Och and Ney, 2003) (Dice coefficient), (Sabet et al., 2016) (incorporation of embeddings into IBM models), (Legrand et al., 2016) (neural network alignment model) and (Pourdamghani et al., 2018) (embeddings are used to encourage words to align to similar words). All of this work requires parallel data.

Attention in NMT (Bahdanau et al., 2014) is related to a notion of soft alignment, but often deviates from conventional word alignments (Ghader and Monz, 2017; Koehn and Knowles, 2017). One difference is that standard attention does not have access to the target word. To address this, Peter et al. (2017) tailor attention matrices to obtain higher quality alignments. Li et al. (2018)’s and Zenkel et al. (2019)’s models perform similarly to GIZA++. Chen et al. (2016), Mi et al. (2016) and Garg et al. (2019) obtain alignments and translations in a multitask setup. Stengel-Eskin et al. (2019) compute similarity matrices of encoder-decoder representations that are leveraged for word alignments, together with supervised learning which requires manually annotated alignment. In contrast to our work, they all require parallel data.

6 Conclusion

We presented word aligners based on contextualized (resp. static) embeddings that perform better than (resp. comparably with) statistical word aligners. Our method is the first that does not require parallel data and is particularly useful for scenarios where a medium number of parallel sentences need to be aligned, but no additional parallel data is available. For a set of 100k parallel sentences, contextualized embeddings achieve an alignment $F_1$ that is 5% higher (absolute) than eflomal.
References


Steven Bird, Ewan Klein, and Edward Loper. 2009. Natural language processing with Python: analyzing text with the natural language toolkit. O'Reilly Media, Inc.


Alexis Conneau, Guillaume Lample, Marc’Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data.


A Evaluation Metrics

Given a set of predicted alignment edges $A$ and a set of sure (possible) gold standard edges $S$ ($P$) we computed our evaluation measures as follows:

\[
\text{prec.} = \frac{|A \cap P|}{|A|}, \\
\text{rec.} = \frac{|A \cap S|}{|S|}, \\
F_1 = \frac{2 \cdot \text{prec.} \cdot \text{rec.}}{\text{prec.} + \text{rec.}}, \\
\text{AER} = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|},
\]

where $|\cdot|$ denotes the cardinality of a set. This is the usual way of evaluating alignments (Och and Ney, 2003).

B Symmetrization

For asymmetric alignments different symmetrization methods exist. fast-align (Dyer et al., 2013) provides an overview and implementation for these methods, which we use. We compare intersection and grow-diag-final-and (G DFA) in Table 3. In terms of F1 G DFA performs better (Intersection wins once, G DFA five times, three ties). As expected, Intersection yields higher precision while G DFA yields higher recall. Thus intersection is preferable for tasks like annotation projection, whereas G DFA is typically used in statistical machine translation.

Table 3: Comparison of symmetrization methods on subword level. Best result across columns in bold.

C Further Results

The analogous numbers from Table 1 in the main paper on word-level can be found in Table 4. Again Distortion is essential for fastText and not necessary for mBERT. Adding Null helps especially for mBERT. Overall the takeaways are consistent with the results from subword-level.

D Hyperparameter Details

We provide a list of customized hyperparameters used in our computations in Table 5. For remaining hyperparameters we used default values as provided in the corresponding implementation (see respective links to the code repositories).

E Hyperparameters $\kappa$ and $\tau$

In the main paper we introduced the hyperparameter $\kappa$. In Figure 3 we plot the performance for different values of $\kappa$. We observe that introducing distortion indeed helps (i.e., $\kappa > 0$) but the actual value is not decisive for performance. This is rather intuitive, as a small adjustment to the similarities is sufficient while larger adjustments do not necessarily hurt or change the Argmax or the optimal point in the Matching Algorithm. In the main paper we have chosen $\kappa = 0.5$.

For $\tau$ in the null-word post-processing we need to use high values as the similarity distributions tend to be quite uniform in the high-dimensional spaces. See Figure 4 for different values of $\tau$. As expected, for $\tau = 1$ no edges are removed and thus the performance is not changed compared to not having a null-word post-processing. With decreasing $\tau$ the precision increases and recall goes down. We use $\tau = 0.999$ for fastText and $\tau = 0.9995$ for mBERT.

F Code

Upon acceptance we will publish the code together with instructions on how to reproduce the results.

G Download Links for Data

Table 6 provides an overview on the data used in the main paper together with download links.
Table 5: Overview on hyperparameters. We only list parameters where we do not use default values.

<table>
<thead>
<tr>
<th>System</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fastText</td>
<td>Version</td>
<td>0.9.1</td>
</tr>
<tr>
<td></td>
<td>Code URL</td>
<td><a href="https://github.com/facebookresearch/fastText/archive/v0.9.1.zip">https://github.com/facebookresearch/fastText/archive/v0.9.1.zip</a></td>
</tr>
<tr>
<td></td>
<td>Downloaded on</td>
<td>11.11.2019</td>
</tr>
<tr>
<td></td>
<td>Embedding Dimension</td>
<td>300</td>
</tr>
<tr>
<td></td>
<td>Maximum Sequence Length</td>
<td>128</td>
</tr>
<tr>
<td>fastalign</td>
<td>Code URL</td>
<td><a href="https://github.com/clab/fast_align">https://github.com/clab/fast_align</a></td>
</tr>
<tr>
<td></td>
<td>Git Hash</td>
<td>7e2bca3b549a1a91b44b0643b0e137e3</td>
</tr>
<tr>
<td></td>
<td>Flags</td>
<td>-d -o -v</td>
</tr>
<tr>
<td>eflomal</td>
<td>Code URL</td>
<td><a href="https://github.com/robertostling/eflomal">https://github.com/robertostling/eflomal</a></td>
</tr>
<tr>
<td></td>
<td>Git Hash</td>
<td>9ef1ace192c97867a4817ce67567c7e6849daff</td>
</tr>
<tr>
<td></td>
<td>Flags</td>
<td>-model 3</td>
</tr>
<tr>
<td></td>
<td>Iterations</td>
<td>5 iter. HMM, 5 iter. Model 1, 5 iter. Model3, 5 iter. Model 4 (DEFAULT)</td>
</tr>
<tr>
<td></td>
<td>k</td>
<td>0.98</td>
</tr>
<tr>
<td>Vecmap</td>
<td>Code URL</td>
<td><a href="https://github.com/artetxem/vecmap.git">https://github.com/artetxem/vecmap.git</a></td>
</tr>
<tr>
<td></td>
<td>Git Hash</td>
<td>b82246f0c29b633039b87fa1565c1d52dbd7a3</td>
</tr>
<tr>
<td></td>
<td>Manual Vocabulary Cutoff</td>
<td>500000</td>
</tr>
</tbody>
</table>

Table 6: Download links for the used data.

<table>
<thead>
<tr>
<th>Data</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENG-DEU Gold</td>
<td><a href="http://www-i6.informatik.rwth-aachen.de/goldAlignment/">http://www-i6.informatik.rwth-aachen.de/goldAlignment/</a></td>
</tr>
<tr>
<td>ENG-FRA Gold</td>
<td><a href="https://nlp.cs.nyu.edu/blinker/">https://nlp.cs.nyu.edu/blinker/</a></td>
</tr>
<tr>
<td>ENG-FAS Gold</td>
<td><a href="http://eceold.ut.ac.ir/en/node/940">http://eceold.ut.ac.ir/en/node/940</a></td>
</tr>
<tr>
<td>Parallel Bible Corpus (We have used a private version of the corpus)</td>
<td><a href="http://opus.nlpl.eu/bible-uedin.php">http://opus.nlpl.eu/bible-uedin.php</a></td>
</tr>
</tbody>
</table>

Figure 3: F1 for ENG-DEU with fastText (Argmax) on word level for different values of κ.

Table 6: Download links for the used data.

<table>
<thead>
<tr>
<th>Data</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENG-DEU Gold</td>
<td><a href="http://www-i6.informatik.rwth-aachen.de/goldAlignment/">http://www-i6.informatik.rwth-aachen.de/goldAlignment/</a></td>
</tr>
<tr>
<td>ENG-FRA Gold</td>
<td><a href="https://nlp.cs.nyu.edu/blinker/">https://nlp.cs.nyu.edu/blinker/</a></td>
</tr>
<tr>
<td>ENG-FAS Gold</td>
<td><a href="http://eceold.ut.ac.ir/en/node/940">http://eceold.ut.ac.ir/en/node/940</a></td>
</tr>
<tr>
<td>Parallel Bible Corpus (We have used a private version of the corpus)</td>
<td><a href="http://opus.nlpl.eu/bible-uedin.php">http://opus.nlpl.eu/bible-uedin.php</a></td>
</tr>
</tbody>
</table>

Table 6: Download links for the used data.

Figure 4: Performance for ENG-DEU with fastText (Argmax) in the top and mBERT[8] (Argmax) in the bottom on word level for different values of τ.