SimAlign: High Quality Word Alignments without Parallel Training Data using Static and Contextualized Embeddings

Masoud Jalili Sabet*, **Philipp Dufter***, **Hinrich Schütze**
Center for Information and Language Processing (CIS), LMU Munich, Germany
{masoud,philipp}@cis.lmu.de

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 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 alignments created from embeddings are competitive and mostly superior to traditional statistical aligners – even in scenarios with abundant parallel data. For example, for a set of 100k parallel sentences, contextualized embeddings achieve a word alignment F_1 for English-German that is more than 5% higher (absolute) than eflomal, a high quality alignment model.

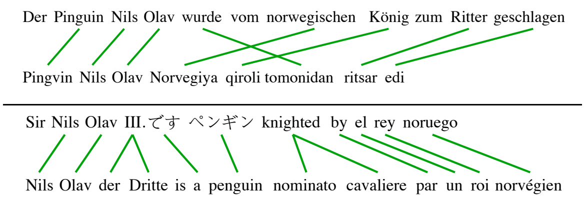

Figure 1: Algorithms that do not rely on parallel training data can align distant language pairs (e.g., German-Uzbek in top) or even mixed sentences (bottom). Alignments are created with our IterMax algorithm.

1 Introduction

Word alignments are 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, 2015b), annotation projection (Yarowsky et al., 2001; Hwa et al., 2002; Padó and Lapata, 2009) and creating multilingual embeddings (Guo et al., 2016; Ammar et al., 2016; Dufter et al., 2018).

Statistical word aligners such as the IBM models (Brown et al., 1993) and their implementations fast-align (Dyer et al., 2013), GIZA++ (Och and Ney, 2003), as well as newer models such as 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 (a sufficient amount to train high quality NMT systems) and their performance deteriorates when parallel text is scarce (Tables 1–2 in (Och and Ney, 2003)).

Recent unsupervised multilingual embedding algorithms that use only non-parallel data provide high quality static (Artetxe et al., 2018a; Conneau et al., 2018) and contextualized embeddings (Devlin et al., 2019; Liu 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, 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 (Schwenk et al., 2019) but assumes that monolingual corpora contain parallel sentences. We extract word alignments from

* Equal contribution - random order.

similarity matrices induced from pretrained multilingual word embeddings. Overall we find the quality of these alignments to be competitive with the state of the art.

Contributions: **(1)** We introduce three 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 pretrained language models have comparable and mostly superior performance to strong statistical word aligners like eflomal. **(4)** We provide evidence that subword processing is beneficial for aligning rare words. We bundle the source code of our methods in a tool called *SimAlign*, which is available online.[1] An interactive online demo is available.[2]

2 Methods

2.1 Alignments from Similarity Matrices

We propose three methods to obtain alignments from similarity matrices. ArgMax is a simple baseline, IterMax a novel iterative algorithm, and Match a graph-theoretical method based on identifying matchings in a bipartite graph.

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., $\mathcal{E}(s^{(k)}) \in \mathbb{R}^{l_k \times d}$ 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 $S_{ij} := \text{sim}\left(\mathcal{E}(s^{(e)})_i, \mathcal{E}(s^{(f)})_j\right)$ is some normalized measure of similarity, e.g., cosine-similarity normalized to be between 0 and 1. We now describe our 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 set $A_{ij} = 1$ if

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

and $A_{ij} = 0$ else. In case of ties, which are unlikely in similarity matrices, we choose the smaller index. If all entries in a row i or column j are 0 we set $A_{ij} = 0$. Similar methods have been applied to Dice coefficients (Och and Ney, 2003) and attention matrices (Garg et al., 2019).

Itermax. Argmax identifies only few alignment edges for many sentences because mutual argmaxes can be rare. As a remedy we propose to apply Argmax iteratively. To this end, we modify the similarity matrix conditioned on the alignment edges found in a previous iteration: if two words i and j have *both* been aligned, we zero out the similarity. Similarly if *neither* is aligned, we leave the similarity unchanged. In case only one of them is aligned, we multiply the similarity with a discount factor $\alpha \in [0,1]$. Intuitively, this encourages the model to focus on unaligned word pairs. However, if the similarity with an already aligned word is exceptionally high, the model can add an additional edge. Note that this explicitly allows one word to be aligned to multiple other words. For details on the algorithm see Figure 2.

Algorithm 1 Itermax.

procedure ITERMAX($S, n_{\text{max}}, \alpha \in [0,1]$)
  $A, M = \text{zeros\_like}(S), \text{zeros\_like}(S)$
  for $n \in [1, \dots, n_{\text{max}}]$ **do**
    $\forall i, j:$

$$M_{ij} = \begin{cases} 1 \text{ if } \max\left(\sum_{l=0}^{l_e} A_{lj}, \sum_{l=0}^{l_f} A_{il}\right) = 0 \\ 0 \text{ if } \min\left(\sum_{l=0}^{l_e} A_{lj}, \sum_{l=0}^{l_f} A_{il}\right) > 0 \\ \alpha \text{ else} \end{cases}$$

    $A_{\text{to\_add}} = \text{get\_argmax\_alignments}(S \odot M)$
    $A = A + A_{\text{to\_add}}$
  end for
  return A
end procedure

Figure 2: Description of the Itermax algorithm. *zeros_like* yields a matrix with zeros and with same shape as the input, *get_argmax_alignments* returns alignments obtained from the Argmax Method, $\odot$ is elementwise multiplication.

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

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

[1] https://github.com/masoudjs/simalign
[2] http://simalign.cis.lmu.de/

subject to A being a matching (i.e., each node has at most one edge) that is maximal (i.e., no additional edges can be added). There are known algorithms to solve the above problem in polynomial time (Kuhn, 1955).

Note that alignments generated with the matching method are inherently bidirectional and do not require any symmetrization as post-processing.

2.2 Post-Processing Alignments

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

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

where κ is a hyperparameter to scale the distortion matrix P between $[(1-\kappa), 1]$. We use $\kappa = 0.5$. See §4.1 for different values. 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 (although questioned by Schulz et al. (2016)). Given an alignment matrix A, we remove alignment edges when the normalized entropy of the similarity distribution is above a threshold τ, a hyperparameter. We consider normalized entropy (i.e., entropy divided by the log of sentence length) to account for different sentence lengths. Intuitively, if a word is not particularly similar to any of the words in the target sentence, we do not align it. That is, we set $A_{ij} = 0$ if

$$\min\left(-\frac{\sum_{k=1}^{l_e} S^h_{ik} \log S^h_{ik}}{\log l_e}, -\frac{\sum_{k=1}^{l_f} S^v_{kj} \log S^v_{kj}}{\log l_f}\right) > \tau,$$

where $S^h_{ik} := S_{ik} / \sum_{j=1}^{l_e} S_{ij}$, and $S^v_{kj} := S_{kj} / \sum_{i=1}^{l_f} S_{ij}$. As the ideal value of τ depends on the actual similarity scores we set τ to a percentile of the entropy values of similarity distributions in all aligned edges. We investigate different percentiles in §4.1.

2.3 Embedding Learning

Static. We train monolingual embeddings with fastText (Bojanowski et al., 2017). Subsequently we use VecMap (Artetxe et al., 2018b) to map the embeddings into a common multilingual space. Note that this algorithm works without any crosslingual supervision (e.g., 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).[3] 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 initial embedding layer) and mBERT[conc] (for concatenation).

In addition, we use XLM-RoBERTa base (Conneau et al., 2019), which is pretrained on 100 languages on CommonCrawl data. We denote alignments obtained using the embeddings from the i-th layer by XLM-R[i], analogously to mBERT.

2.4 Word and Subword Alignments

We investigate both alignments between subwords such as BPE/wordpiece (Sennrich et al., 2016) (which are widely used for contextualized language models) and words. For the *word* level, we use NLTK tokenizer (Bird et al., 2009) (e.g., for tokenizing Wikipedia in order to train fastText). For the *subword* level, we generally use multilingual BERT's vocabulary[3] and BERT's tokenizer.[4] Only for XLM-R we use the XLM-R vocabulary.

As gold standards are all word-level, we can only evaluate on the word level. Each gold standard comes with a gold tokenization, thus no additional tokenization is necessary. To convert subword to word alignments for evaluation we apply the rule: "two words are aligned if any of their subwords are aligned" (see Figure 3). Thus a single word can be aligned with multiple other words.

[3] https://github.com/google-research/bert/blob/master/multilingual.md

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

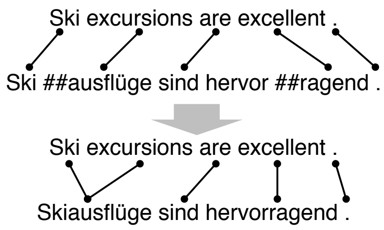

Figure 3: Subword alignments are converted to word alignments for evaluation.

2.5 Baselines

We compare to three popular statistical alignment models that all require parallel training data. **fast-align** (Dyer et al., 2013) is an implementation of an alignment algorithm based on IBM Model 2. It is popular because of its speed and high quality. **eflomal**[5] (based on efmaral by 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). Further we use the widely used software package **GIZA++** (Och and Ney, 2003), which combines IBM Alignment Models. We use its standard settings: 5 iterations each for the HMM model, IBM Model 1, 3 and 4 with $p_0 = 0.98$.

Symmetrization. Traditional word alignment models create forward and backward alignments and then symmetrize them (Koehn, 2010). We compared the symmetrization methods grow-diag-final-and (GDFA) and intersection and found them to perform comparably. See Table 7 in the appendix for a comparison. We use GDFA throughout the paper.

3 Experiments

3.1 Data

We work with a diverse set of 7 languages. As **test data** we use three language pairs from the WPT2005 shared task:[6] English-Hindi, English-French (Och and Ney, 2000), and English-Romanian (Mihalcea and Pedersen, 2003). In addition, we use Europarl gold alignments[7] for English-German, gold alignments by Tavakoli and Faili (2014) for English-Persian, and by Bojar and Prokopová (2006) for English-Czech. Note that the Persian gold standard is lowercased. FAS, CES and RON contain only sure edges and no possible edges.

For models requiring parallel training data we select additional parallel **training data** that is consistent with the target domain where available. See Table 1 for an overview of the used data as well as the corresponding size. Unless indicated otherwise we use the whole parallel training data for training of eflomal, fast-align and GIZA++. We show the effect of adding more or less training data in Figure 7. Since mBERT is pretrained on Wikipedia, we train fastText embeddings on Wikipedia as well. For hyperparameters of all models see Table 8 in the appendix.

3.2 Evaluation Measures

Given a set of predicted alignment edges A and a set of sure (possible) gold standard edges S (P), we use the following evaluation measures:

$$\begin{aligned}
\text{prec.} &= \frac{|A \cap P|}{|A|} \\
\text{rec.} &= \frac{|A \cap S|}{|S|} \\
F_1 &= \frac{2 \text{ prec. rec.}}{\text{prec.} + \text{rec.}} \\
\text{AER} &= 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|},
\end{aligned}$$

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

4 Results

Given the large amount of possible experiments when considering 6 language pairs we do not have space to present all numbers for all languages. If we show results only for one pair, we choose ENG-DEU as it is an established and well-known dataset (EuroParl). If we show results for more languages we fall back to DEU, CES and HIN, to show effects on a mid-resource morphologically rich language (CES) and a low-resource language written in a different script (HIN).

4.1 Hyperparameter Investigation

Layers. Figure 4 shows a parabolic trend across layers of mBERT as well as of XLM-R with layer 8 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

[5] https://github.com/robertostling/eflomal
[6] http://web.eecs.umich.edu/~mihalcea/wpt05/
[7] www-i6.informatik.rwth-aachen.de/goldAlignment/

Lang.	Gold Standard	Gold St. Size	Parallel Data	Parallel Data Size	Wikipedia Size
ENG-CES	(Bojar and Prokopová, 2006)	2501	EuroParl (Koehn, 2005)	646K	8M
ENG-DEU	EuroParl[a]	508	EuroParl (Koehn, 2005)	1920K	48M
ENG-FAS	(Tavakoli and Faili, 2014)	400	TEP (Pilevar et al., 2011)	600K	5M
ENG-FRA	WPT2005, (Och and Ney, 2000),	447	Hansards[b](Germann, 2001)	1130K	32M
ENG-HIN	WPT2005[c]	90	Emille (McEnery et al., 2000)	3K	1M
ENG-RON	WPT2005, (Mihalcea and Pedersen, 2003)	203	Constitution, Newspaper[d]	50K	3M

[a] www-i6.informatik.rwth-aachen.de/goldAlignment/
[b] https://www.isi.edu/natural-language/download/hansard/index.html
[c] http://web.eecs.umich.edu/ mihalcea/wpt05/
[d] http://web.eecs.umich.edu/ mihalcea/wpt05/

Table 1: Overview of datasets. "Size" refers to the number of sentences. "Parallel Data Size" refers to the number of parallel sentences in addition to the gold alignments. Our sentence tokenized version of the English Wikipedia has 105M sentences.

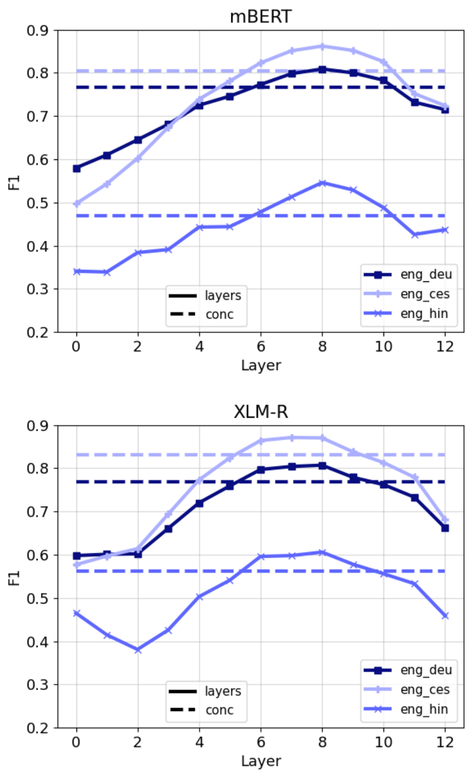

Figure 4: Word alignment performance across layers of mBERT (top) and XLM-R (bottom). Results are F_1 on subword level with Argmax and no post-processing applied.

too weak for high-quality alignments while the last layers are too specialized on the pretraining task (masked language modeling).

Itermax. Table 2 shows results for Argmax (i.e., 1 Iteration) as well as Itermax (i.e., 2 or more iterations of Argmax). As expected, with more iterations precision drops in favor of recall. Overall Itermax achieves higher F_1 scores for the three language pairs (equal for ENG-CES). For Hindi the performance increase is the highest. We hypothesize that for more distant languages Itermax is more beneficial as similarity between wordpieces may be generally lower, thus exhibiting fewer mutual argmaxes. For the rest of the paper we use for Itermax 2 Iterations with $\alpha = 0.9$ as it exhibits best performance (5 out of 6 wins in Table 2).

Emb.	Iter.	α	ENG-DEU Prec.	ENG-DEU Rec.	ENG-DEU F_1	ENG-DEU AER	ENG-CES Prec.	ENG-CES Rec.	ENG-CES F_1	ENG-CES AER	ENG-HIN Prec.	ENG-HIN Rec.	ENG-HIN F_1	ENG-HIN AER
mBERT[8]	**1**	-	**.92**	.69	.79	.21	**.95**	.80	**.87**	**.13**	**.84**	.39	.53	.47
	2	**.90**	.85	.77	**.81**	**.19**	.87	.86	**.87**	.14	.75	.46	.57	.43
	2	.95	.83	.80	**.81**	**.19**	.85	.89	**.87**	**.13**	.73	.48	.58	.42
	2	1	.77	.79	.78	.22	.80	.86	.83	.17	.63	.46	.53	.47
	3	**.90**	.81	.80	.80	.20	.83	.88	.85	.15	.70	.49	.57	.43
	3	.95	.78	**.83**	**.81**	.20	.81	**.91**	.86	.15	.68	**.52**	**.59**	**.41**
	3	1	.73	**.83**	.77	.23	.76	**.91**	.82	.18	.58	.51	.54	.46
fastText	**1**	-	**.81**	.48	.60	.40	**.86**	.59	.70	.30	**.75**	.35	.48	.52
	2	**.90**	.69	.56	**.62**	**.38**	.74	.69	**.71**	**.29**	.63	.42	**.50**	**.50**
	2	.95	.66	.56	.61	.39	.71	.69	.70	.30	.59	.41	.48	.52
	2	1	.59	.55	.57	.43	.62	.65	.63	.37	.53	.39	.45	.55
	3	**.90**	.63	**.59**	.61	.39	.67	.72	.70	.31	.57	.43	.49	.51
	3	.95	.59	**.59**	.59	.41	.63	**.73**	.68	.33	.53	**.44**	.48	.52
	3	1	.53	.58	.55	.45	.55	.70	.62	.39	.48	.43	.45	.55

Table 2: Itermax with different number of iterations as well as different α. Results are on word level.

Hyperparameters κ and τ. In Figure 5 we plot the performance for different values of κ. 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. We choose $\kappa = 0.5$.

For τ in null-word post-processing, we plot precision, recall and F_1 in Figure 6 when assigning τ different percentile values. Recall that values for τ depend on the similarity distribution of all aligned edges. As expected, when using the 100 percentile no edges are removed and thus the performance is not changed compared to not having a null-word post-processing. With decreasing the value of τ the precision increases and recall goes down, while F_1 remains fairly stable. We assign τ

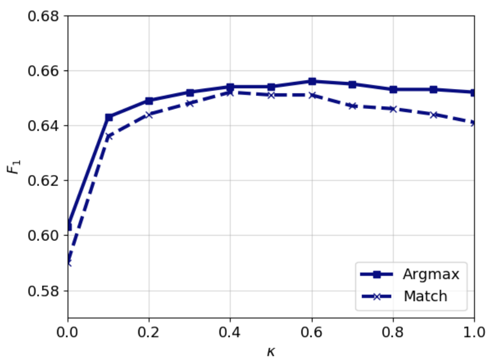

Figure 5: F1 for ENG-DEU with fastText (Argmax) on word level for different values of κ.

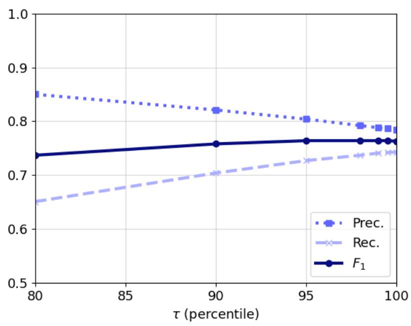

Figure 6: Performance for ENG-DEU with mBERT[8] (Match) on word level when setting the value of τ to different percentiles. τ can be used for trading precision against recall. F_1 remains stable although it decreases slightly when assigning τ the value of a smaller percentile (e.g., 80)

the 95th percentile from now on.

Table 3 compares **alignment and post-processing methods**. Argmax and Itermax generally have higher precision whereas Match has higher recall. Adding Null almost always increases precision, but at the cost of recall, resulting mostly in a lower F_1 score. Adding a distortion prior boosts performance for static embeddings, e.g., from .70 to .77 for ENG-CES Argmax F_1. However, for Hindi a distortion prior is harmful. Further Dist has little and sometimes harmful effects on mBERT indicating that mBERT's contextualized representations already match well across languages.

To summarize: Itermax exhibits the best and most stable performance, for high precision alignments one should use Argmax, for high recall Match is recommended. A distortion prior is recommended for static embeddings (except for HIN). Null should be applied when one wants to push precision even higher (e.g., for annotation projection).

		ENG-DEU				ENG-CES				ENG-HIN			
Emb.	Method	Prec.	Rec.	F_1	AER	Prec.	Rec.	F_1	AER	Prec.	Rec.	F_1	AER
fastText	Argmax	.81	.48	.60	.40	.86	.59	.70	.30	**.75**	**.35**	**.48**	**.52**
	+Dist	**.84**	**.53**	**.65**	**.35**	**.89**	**.68**	**.77**	**.23**	.64	.29	.40	.60
	+Null	.81	.46	.59	.41	.86	.56	.68	.32	.74	.33	.46	.54
	Itermax	.66	.56	.61	.39	.71	.69	.70	.30	.59	**.41**	**.48**	**.52**
	+Dist	**.71**	**.61**	**.66**	**.34**	**.75**	**.76**	**.76**	**.25**	.54	.36	.43	.57
	+Null	.69	.53	.60	.40	.74	.66	.70	.30	**.63**	.39	**.48**	**.52**
	Match	.60	.58	.59	.41	.65	.71	.68	.32	.55	**.42**	**.48**	**.52**
	+Dist	**.67**	**.64**	**.65**	**.35**	**.72**	**.78**	**.75**	**.25**	.49	.38	.43	.57
	+Null	.61	.56	.58	.42	.66	.69	.67	.33	**.56**	.41	.47	.53
mBERT[8]	Argmax	.92	**.69**	**.79**	**.21**	**.95**	**.80**	**.87**	**.13**	.84	**.39**	**.53**	**.47**
	+Dist	.91	.67	.77	.23	.93	.79	.85	.15	.68	.29	.41	.60
	+Null	**.93**	.67	.78	.22	**.95**	.77	.85	.15	**.85**	.38	.52	.48
	Itermax	.83	**.80**	**.81**	**.19**	.85	**.89**	**.87**	**.13**	.73	**.48**	**.58**	**.42**
	+Dist	.82	.75	.79	.22	.84	.85	.85	.15	.56	.34	.42	.58
	+Null	**.86**	.75	.80	.20	**.88**	.84	.86	.14	**.76**	.45	.56	.44
	Match	.78	**.74**	**.76**	**.24**	.81	**.85**	**.83**	**.17**	.67	**.51**	**.58**	**.42**
	+Dist	.75	.71	.73	.27	.79	.83	.81	.20	.45	.34	.39	.61
	+Null	**.80**	.73	**.76**	**.24**	**.83**	.83	**.83**	**.17**	**.68**	.50	**.58**	.43

Table 3: Comparison of methods for inducing alignments from similarity matrices. All results are word-level. Best result per embedding type and method across columns in bold.

4.2 Comparison with SoTA

Overall. Table 4 shows that mBERT and XLM-R consistently perform well. Our three baselines, eflomal, fast-align and GIZA++, are mostly outperformed (except for RON). XLM-R yields mostly higher values than mBERT. In comparison with other published numbers, alignments from contextualized embeddings outperform them or are almost on-par.

Only Garg et al. (2019) has higher performance for ENG-DEU and ENG-FRA. They train a multitask NMT system. However, extracting alignments from similarity matrices is a very simple and efficient method which yields surprisingly strong performance – we attribute this to the strong contextualization in mBERT and XLM-R.

Numbers for ENG-RON are worse than eflomal and (Östling, 2015a). We will investigate the reason for this more closely.

Surprisingly, fastText outperforms fast-align in two languages. We consider this surprising as fastText did not have access to parallel data or any multilingual signal. Thus for very small parallel corpora (<10K sentences) using fastText embeddings is an alternative to fast-align.

Parallel Data. Figure 7 shows that fast-align and eflomal get better with more training data with eflomal outperforming fast-align, as expected. However, even with $1.9M$ parallel sentences mBERT outperforms both statistical baselines. fastText becomes competitive for fewer than 1000 par-

		Method	ENG-CES F_1	ENG-CES AER	ENG-DEU F_1	ENG-DEU AER	ENG-FAS F_1	ENG-FAS AER	ENG-FRA F_1	ENG-FRA AER	ENG-HIN F_1	ENG-HIN AER	ENG-RON F_1	ENG-RON AER
Prior Work		(Dyer et al., 2011)		.21										
		(Tamura et al., 2014) RNN							.93					
		(Östling, 2015a)							**.94**	.06	.58	.42	**.73**	**.27**
		(Legrand et al., 2016)		.16										
		(Östling and Tiedemann, 2016) efmaral								.08		.47		.28
		(Zenkel et al., 2019) fast-align				.27				.11				.32
		(Zenkel et al., 2019) GIZA++				.21				.06				.28
		(Garg et al., 2019) Multitask				**.16**				**.05**				
Baselines	Word	fast-align	.76	.25	.71	.29	.46	.54	.84	.18	.34	.66	.68	.33
		GIZA++	.82	.18	.77	.23	.57	.43	.92	.09	.48	.52	.69	.32
		eflomal	.85	.15	.77	.23	.59	.41	.93	.08	.51	.49	.71	.29
	Subword	fast-align	.78	.23	.71	.30	.45	.55	.83	.19	.38	.62	.68	.32
		GIZA++	.82	.18	.78	.22	.57	.43	.92	.09	.48	.52	.69	.32
		eflomal	.84	.17	.76	.24	.63	.37	.92	.09	.52	.48	.72	.28
Methods	Word	fastText - Argmax	.70	.30	.60	.40	.50	.50	.77	.22	.48	.52	.47	.53
		mBERT[8] - Argmax	**.87**	**.13**	.79	.21	.67	.33	**.94**	.06	.53	.47	.64	.36
		XLM-R[8] - Argmax	**.87**	**.13**	.79	.22	.70	**.30**	.93	.06	.58	.42	.70	.30
	Subword	fastText - Argmax	.58	.42	.56	.44	.09	.91	.73	.26	.04	.96	.43	.58
		mBERT[8] - Argmax	.86	.14	**.81**	.19	.67	.33	**.94**	.06	.54	.46	.65	.35
		XLM-R[8] - Argmax	**.87**	**.13**	**.81**	.19	**.71**	**.30**	.93	.07	**.60**	**.40**	.71	.29

Table 4: Comparison of out methods, baselines and related work. Best overall result per column in bold.

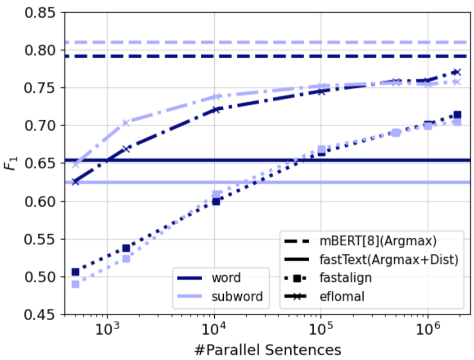

Figure 7: Learning curves of fast-align/eflomal vs. embedding-based alignments. Results shown are F_1 on word and subword level for ENG-DEU.

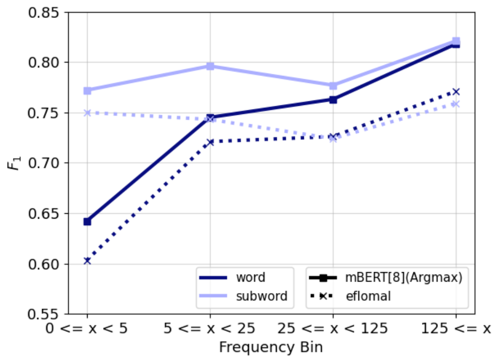

Figure 8: Results for different frequency bins. An edge in S, P, or A is attributed to exactly one bin based on the minimum frequency of the involved words (denoted by x). Eflomal is trained on 100k parallel sentences. Word frequencies are computed on this 100k parallel corpus. For a version with 1000k parallel sentences see appendix.

allel sentences and outperforms fast-align even with 10K sentences. *The main takeaway is that mBERT-based alignments, a method that does not need any parallel training data, are competitive with state-of-the-art aligners, even in the high resource case.*

4.3 Words and Subwords

In Table 4 subword processing yields slight improvements over word-level processing for most methods. Only 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 word distributions – this effect is strongest for Persian and Hindi, probably due to different scripts and thus different subword distributions. Initial experiments showed that adding supervision in terms of a dictionary helps restore performance. We will investigate this more closely in future work.

We hypothesize that subword processing is beneficial for aligning rare words. To show this, we compute our evaluation measures for different frequency bins. More specifically, we only consider alignment edges for the computation where at least one of the member words has a certain frequency in a reference corpus (in our case 100k lines from the ENG-DEU EuroParl corpus). That is, we only consider the edge (i, j) in A, S or P if the minimum of the source and target word frequency is in $[\gamma_l, \gamma_u)$ where γ_l and γ_u are bin boundaries.

Figure 8 shows F_1 for different frequency bins.

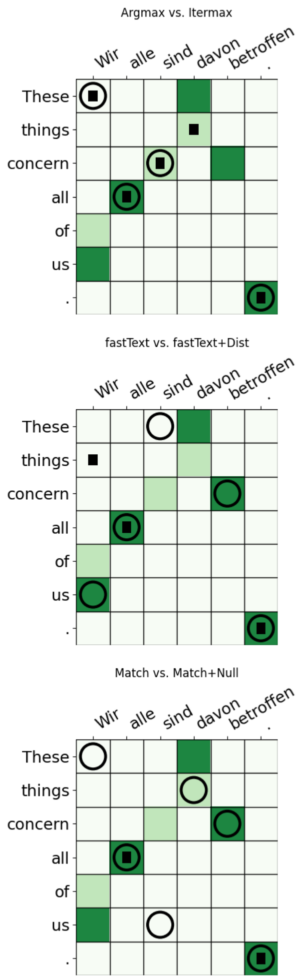

Figure 9: Comparison of some alignment systems. Dark/light green: sure/possible edges in the gold standard. Circles are alignments from the first mentioned system in the headline, boxes alignemnts from the second system.

For rare words both eflomal and mBERT show a severely decreased performance on word level, but not on subword level. This provides some evidence for our hypothesis.

4.4 Alignment Examples

Figure 9 gives alignment examples. One can see that Itermax adds a correct alignment edge in the second iteration (Argmax vs Itermax). The distortion prior in fastText does not help in this example, as it is heavily distorted, but we see that the prior works as intended (fastText vs fastText+Dist). The null-alignment removes some wrong edges, but unfortunately also the correct edge between "betroffen" and "concern" (Match vs Match+Null).

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). Neural network based extensions of these models have been considered as well (Ayan et al., 2005; Ho and Yvon, 2019) . All of these models are trained on parallel text. Our method instead aligns based on embeddings that are induced from monolingual data only. Niehues and Vogel (2008) model the alignment matrix with a conditional random field. To train this they require a manually created gold alignment.

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). Tamura et al. (2014) use recurrent neural networks to learn alignments. They use noise contrastive estimation to avoid supervision. All of this work requires parallel data. Concurrent to us, Libovický et al. (2019) find that mBERT gives raise to good word alignments. But they do not focus on mBERT's use as a high performance alignment tool, but rather on evaluating the "language-neutrality" of mBERT.

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++. Ding et al. (2019) propose better decoding algorithms to deduce word alignments from NMT predictions. Chen et al. (2016), Mi et al. (2016) and Garg et al. (2019) obtain alignments and translations in a multitask setup. Garg et al. (2019) find that operating on subword level can be beneficial for word alignment models. Li et al. (2019) propose two methods to extract alignments from NMT models, however they do not outperform fast-align. 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. We find our proposed methods to be competitive with some of this work. Further, 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 does not require parallel data and is particularly useful for scenarios where a low or 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. In future work we plan to investigate how to leverage existing parallel data effectively in combination with our proposed methods.

We gratefully **acknowledge** funding through a Zentrum Digitalisierung.Bayern fellowship awarded to the second author. This work was supported by the European Research Council (# 740516). We thank Matthias Huck, Jindřich Libovický and Alex Fraser for interesting discussions and valuable comments. Thanks to Jindřich for pointing out that mBERT can align mixed-language sentences as shown in Figure 1.

References

Tamer Alkhouli, Gabriel Bretschner, and Hermann Ney. 2018. On the alignment problem in multi-head attention-based neural machine translation. In *Proceedings of the Third Conference on Machine Translation*.

Tamer Alkhouli, Gabriel Bretschner, Jan-Thorsten Peter, Mohammed Hethnawi, Andreas Guta, and Hermann Ney. 2016. Alignment-based neural machine translation. In *Proceedings of the First Conference on Machine Translation.*

Tamer Alkhouli and Hermann Ney. 2017. Biasing attention-based recurrent neural networks using external alignment information. In *Proceedings of the Second Conference on Machine Translation*.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. 2016. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018a. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018b. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Necip Fazil Ayan, Bonnie J Dorr, and Christof Monz. 2005. Neuralign: Combining word alignments using neural networks. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

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

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5.

Ondrej Bojar and Magdalena Prokopová. 2006. Czech-english word alignment. In *Proceedings of the International Conference on Language Resources and Evaluation*.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2).

Wenhu Chen, Evgeny Matusov, Shahram Khadivi, and Jan-Thorsten Peter. 2016. Guided alignment training for topic-aware neural machine translation. *AMTA 2016*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Shuoyang Ding, Hainan Xu, and Philipp Koehn. 2019. Saliency-driven word alignment interpretation for neural machine translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*.

Philipp Dufter, Mengjie Zhao, Martin Schmitt, Alexander Fraser, and Hinrich Schütze. 2018. Embedding learning through multilingual concept induction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of IBM Model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Chris Dyer, Jonathan Clark, Alon Lavie, and Noah A Smith. 2011. Unsupervised word alignment with arbitrary features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics.

Sarthak Garg, Stephan Peitz, Udhyakumar Nallasamy, and Matthias Paulik. 2019. Jointly learning to align and translate with transformer models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*.

Ulrich Germann. 2001. Aligned hansards of the 36th parliament of canada.

Hamidreza Ghader and Christof Monz. 2017. What does attention in neural machine translation pay attention to? In *Proceedings of the Eighth International Joint Conference on Natural Language Processing*.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2016. A representation learning framework for multi-source transfer parsing. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Anh Khoa Ngo Ho and François Yvon. 2019. Neural baselines for word alignment.

Rebecca Hwa, Philip Resnik, Amy Weinberg, and Okan Kolak. 2002. Evaluating translational correspondence using annotation projection. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Machine Translation Summit*, volume 5.

Philipp Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*.

Harold W Kuhn. 1955. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2).

Joël Legrand, Michael Auli, and Ronan Collobert. 2016. Neural network-based word alignment through score aggregation. In *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*.

William D Lewis and Fei Xia. 2008. Automatically identifying computationally relevant typological features. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*.

Xintong Li, Guanlin Li, Lemao Liu, Max Meng, and Shuming Shi. 2019. On the word alignment from neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Xintong Li, Lemao Liu, Zhaopeng Tu, Shuming Shi, and Max Meng. 2018. Target foresight based attention for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Jindřich Libovický, Rudolf Rosa, and Alexander Fraser. 2019. How language-neutral is multilingual bert? *arXiv preprint arXiv:1911.03310*.

Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. Neural machine translation with supervised attention. In *Proceedings of the 26th International Conference on Computational Linguistics*.

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*.

Anthony McEnery, Paul Baker, Rob Gaizauskas, and Hamish Cunningham. 2000. Emille: Building a corpus of south asian languages. *VIVEK-BOMBAY-*, 13(3).

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Supervised attentions for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Rada Mihalcea and Ted Pedersen. 2003. An evaluation exercise for word alignment. In *Proceedings of the HLT-NAACL 2003 Workshop on Building and using parallel texts: data driven machine translation and beyond*.

Jan Niehues and Stephan Vogel. 2008. Discriminative word alignment via alignment matrix modeling. In *Proceedings of the Third Workshop on Statistical Machine Translation*. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1).

Robert Östling. 2015a. *Bayesian models for multilingual word alignment*. Ph.D. thesis, Department of Linguistics, Stockholm University.

Robert Östling. 2015b. Word order typology through multilingual word alignment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

Robert Östling and Jörg Tiedemann. 2016. Efficient word alignment with markov chain monte carlo. *The Prague Bulletin of Mathematical Linguistics*, 106(1).

Sebastian Padó and Mirella Lapata. 2009. Cross-lingual annotation projection for semantic roles. *Journal of Artificial Intelligence Research*, 36.

Jan-Thorsten Peter, Arne Nix, and Hermann Ney. 2017. Generating alignments using target foresight in attention-based neural machine translation. *The Prague Bulletin of Mathematical Linguistics*, 108(1).

Mohammad Taher Pilevar, Heshaam Faili, and Abdol Hamid Pilevar. 2011. TEP: Tehran English-Persian parallel corpus. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 68–79. Springer.

Nima Pourdamghani, Marjan Ghazvininejad, and Kevin Knight. 2018. Using word vectors to improve word alignments for low resource machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Ofir Press and Noah A Smith. 2018. You may not need attention. *arXiv preprint arXiv:1810.13409*.

Lyle Ramshaw and Robert E Tarjan. 2012. On minimum-cost assignments in unbalanced bipartite graphs. *HP Labs, Palo Alto, CA, USA, Tech. Rep. HPL-2012-40R1*.

Masoud Jalili Sabet, Heshaam Faili, and Gholamreza Haffari. 2016. Improving word alignment of rare words with word embeddings. In *Proceedings of the 26th International Conference on Computational Linguistics*.

Philip Schulz, Wilker Aziz, and Khalil Simaan. 2016. Word alignment without null words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Holger Schwenk, Vishrav Chaudhary, Shuo Sun, Hongyu Gong, and Francisco Guzmán. 2019. Wikimatrix: Mining 135m parallel sentences in 1620 language pairs from wikipedia. *arXiv preprint arXiv:1907.05791*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Frank Smadja, Kathleen R McKeown, and Vasileios Hatzivassiloglou. 1996. Translating collocations for bilingual lexicons: A statistical approach. *Computational linguistics*, 22(1).

Elias Stengel-Eskin, Tzu-ray Su, Matt Post, and Benjamin Van Durme. 2019. A discriminative neural model for cross-lingual word alignment. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*.

Akihiro Tamura, Taro Watanabe, and Eiichiro Sumita. 2014. Recurrent neural networks for word alignment model. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Leila Tavakoli and Heshaam Faili. 2014. Phrase alignments in parallel corpus using bootstrapping approach. *International Journal of Information & Communication Technology Research*, 6(3).

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*.

Elena Voita, Rico Sennrich, and Ivan Titov. 2019. The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4387–4397.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the first international conference on Human language technology research*.

Thomas Zenkel, Joern Wuebker, and John DeNero. 2019. Adding interpretable attention to neural translation models improves word alignment. *arXiv preprint arXiv:1901.11359*.

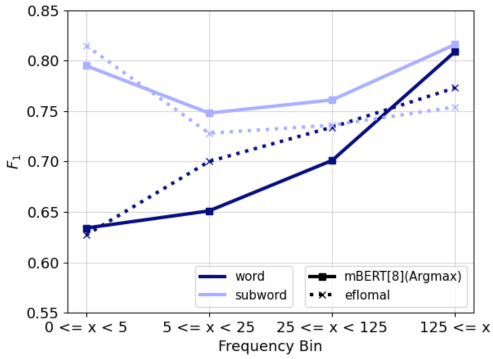

Figure 10: Results for different frequency bins. An edge in S, P, or A is attributed to exactly one bin based on the minimum frequency of the involved words (denoted by x). Eflomal is trained on 1000k parallel sentences. Word frequencies are computed on this 1000k parallel corpus.

A Further Results

The analogous numbers from Table 3 on subword-level can be found in Table 6. 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.

A more detailed version of Table 4 with precision and recall can be found in Table 5

Figure 10 shows the same as Figure 8 but now with a reference corpus of 1000K parallel sentences. The main takeaways are similar.

B Symmetrization

For asymmetric alignments different symmetrization methods exist. (Dyer et al., 2013) provide an overview and implementation (fast-align) for these methods, which we use. We compare intersection and grow-diag-final-and (GDFA) in Table 7. In terms of F1 GDFA performs better (Intersection wins four times, GDFA eleven times, three ties). As expected, Intersection yields higher precision while GDFA yields higher recall. Thus intersection is preferable for tasks like annotation projection, whereas GDFA is typically used in statistical machine translation.

C Hyperparameter Details

We provide a list of customized hyperparameters used in our computations in Table 8. For remaining hyperparameters we used default values as provided in the corresponding implementation (see respective links to the code repositories).

D Examples

We show some more alignment examples in Figure 11, Figure 12, Figure 13, and Figure 14.

		Method	ENG-CES				ENG-DEU				ENG-FAS				ENG-FRA				ENG-HIN				ENG-RON			
			Prec.	Rec.	F_1	AER	Prec.	Rec.	F_1	AER	Prec.	Rec.	F_1	AER	Prec.	Rec.	F_1	AER	Prec.	Rec.	F_1	AER	Prec.	Rec.	F_1	AER
		(Dyer et al., 2011)				.21																				
		(Tamura et al., 2014) RNN															.93									
		(Östling, 2015a)															**.94**	.06			.58	.42			**.73**	**.27**
		(Legrand et al., 2016) Neural				.16												.10								
		(Östling and Tiedemann, 2016) efmaral																.08				.47				.28
		(Zenkel et al., 2019) GIZA++								.21								.06								.28
		(Zenkel et al., 2019) fast-align								.27								.11								.32
		(Garg et al., 2019) Multitask								**.16**								**.05**								
Baselines	Word	fast-align	.71	.81	.76	.25	.70	.73	.71	.29	.48	.44	.46	.54	.77	.92	.84	.18	.34	.33	.34	.66	.69	**.67**	.68	.33
		GIZA++	.79	.86	.82	.18	.79	.75	.77	.23	.58	.56	.57	.43	.89	.95	.92	.09	.52	.44	.48	.52	.74	.64	.69	.32
		eflomal	.84	.87	.85	.15	.80	.75	.77	.23	.64	.55	.59	.41	.91	.95	.93	.08	.61	.44	.51	.49	.81	.63	.71	.29
	Subword	fast-align	.72	.84	.78	.23	.67	.74	.71	.30	.47	.44	.45	.55	.77	.91	.83	.19	.39	.37	.38	.62	.69	**.67**	.68	.32
		GIZA++	.79	.86	.82	.18	.78	.78	.78	.22	.58	.56	.57	.43	.89	.95	.92	.09	.52	.44	.48	.52	.74	.64	.69	.32
		eflomal	.80	.88	.84	.17	.74	.78	.76	.24	.66	.60	.63	.37	.89	.96	.92	.09	.58	.47	.52	.48	.78	**.67**	.72	.28
Methods	Word	fastText - Itermax	.71	.69	.70	.30	.66	.56	.61	.39	.60	.45	.52	.48	.72	.79	.75	.25	.59	.41	.48	.52	.59	.41	.48	.52
		mBERT[8] - Itermax	.85	.89	**.87**	**.13**	.83	.80	**.81**	.19	.77	.66	.71	.29	.89	.96	.92	.09	.73	.48	.58	.42	.73	.48	.58	.42
		XLM-R[8] - Itermax	.88	.87	**.87**	**.13**	.85	.76	.80	.20	.83	.64	**.73**	**.28**	.89	.94	.92	.09	.79	.49	.60	.40	.79	.49	.60	.40
		fastText - Argmax	.86	.59	.70	.30	.81	.48	.60	.40	.75	.38	.50	.50	.85	.71	.77	.22	.75	.35	.48	.52	.77	.34	.47	.53
		mBERT[8] - Argmax	.95	.80	**.87**	**.13**	.92	.69	.79	.21	.88	.54	.67	.33	**.97**	.91	.90	.06	.84	.39	.53	.47	.90	.50	.64	.36
		XLM-R[8] - Argmax	**.96**	.80	**.87**	**.13**	**.93**	.68	.79	.22	**.91**	.57	.70	.30	.96	.91	.93	.06	**.88**	.44	.58	.42	**.94**	.56	.70	.30
	Subword	fastText - Itermax	.58	.57	.58	.43	.61	.54	.57	.43	.20	.07	.11	.89	.68	.77	.72	.29	.13	.04	.06	.94	.13	.04	.06	.94
		mBERT[8] - Itermax	.83	**.90**	.86	.14	.81	**.81**	**.81**	.19	.74	.66	.70	.30	.89	**.97**	.92	.09	.70	.50	.59	.42	.70	.50	.59	.42
		XLM-R[8] - Itermax	.82	.89	.86	.15	.81	.79	.80	.20	.78	**.68**	.72	**.28**	.87	.95	.91	.10	.74	**.51**	**.61**	**.39**	.74	.51	.61	.39
		fastText - Argmax	.72	.48	.58	.42	.75	.45	.56	.44	.27	.06	.09	.91	.80	.67	.73	.26	.14	.02	.04	.96	.67	.31	.43	.58
		mBERT[8] - Argmax	.92	.81	.86	.14	.92	.72	**.81**	.19	.85	.56	.67	.33	.96	.92	**.94**	.06	.81	.41	.54	.46	.88	.51	.65	.35
		XLM-R[8] - Argmax	.92	.83	**.87**	**.13**	.92	.72	**.81**	.19	.87	.59	.71	.30	.95	.91	.93	.07	.86	.46	.60	.40	.91	.58	.71	.29

Table 5: Comparison of word and subword levels. Best overall result per column in bold.

Emb.	Method	ENG-DEU				ENG-CES				ENG-HIN			
		Prec.	Rec.	F_1	AER	Prec.	Rec.	F_1	AER	Prec.	Rec.	F_1	AER
fastText	Argmax	.75	.45	.56	.44	.72	.48	.58	.42	.14	.02	.04	.96
	+*Dist*	**.79**	**.51**	**.62**	**.38**	**.77**	**.58**	**.66**	**.34**	**.16**	**.04**	**.06**	**.94**
	+*Null*	.76	.43	.55	.45	.74	.47	.57	.42	.14	.02	.04	.96
	Itermax	.61	.54	.57	.43	.58	.57	.58	.43	.13	.04	.06	.94
	+*Dist*	**.67**	**.60**	**.64**	**.36**	**.63**	**.66**	**.65**	**.36**	**.15**	**.07**	**.09**	**.91**
	+*Null*	.64	.52	.57	.43	.62	.56	.59	.41	.14	.04	.07	.93
	Match	.51	.58	.54	.46	.44	.61	.52	.49	**.10**	.08	**.09**	**.91**
	+*Dist*	**.59**	**.66**	**.62**	**.38**	**.54**	**.71**	**.61**	**.39**	**.10**	**.09**	**.09**	**.91**
	+*Null*	.52	.57	.54	.46	.46	.60	.52	.48	**.10**	.08	**.09**	**.91**
mBERT[8]	Argmax	.92	**.72**	**.81**	**.19**	**.92**	**.81**	**.86**	**.14**	.81	**.41**	**.54**	**.46**
	+*Dist*	.90	.70	.79	.21	.91	.80	.85	.15	.65	.30	.41	.59
	+*Null*	**.93**	.70	.80	.20	**.92**	.78	.85	.15	**.82**	.40	**.54**	.47
	Itermax	.81	**.81**	**.81**	**.19**	.83	**.90**	**.86**	**.14**	.70	**.50**	**.59**	**.42**
	+*Dist*	.81	.77	.79	.21	.82	.87	.84	.16	.53	.35	.42	.58
	+*Null*	**.85**	.77	**.81**	.20	**.84**	.86	.85	.15	**.72**	.47	.57	.43
	Match	.75	**.80**	**.78**	**.23**	.76	**.90**	**.82**	**.18**	.64	**.52**	**.58**	**.43**
	+*Dist*	.72	.77	.75	.26	.74	.88	.80	.20	.45	.37	.40	.60
	+*Null*	**.77**	.78	**.78**	**.23**	**.77**	.88	**.82**	.19	**.65**	.51	.57	**.43**

Table 6: Comparison of methods for inducing alignments from similarity matrices. All results are subword-level. Best result per embedding type across columns in bold.

Method	Symm.	ENG-CES				ENG-DEU				ENG-FAS				ENG-FRA				ENG-HIN				ENG-RON			
		Prec.	Rec.	F_1	AER	Prec.	Rec.	F_1	AER	Prec.	Rec.	F_1	AER	Prec.	Rec.	F_1	AER	Prec.	Rec.	F_1	AER	Prec.	Rec.	F_1	AER
eflomal	Inters.	**.95**	.79	**.86**	**.14**	**.91**	.66	.76	.24	**.88**	.43	.58	.42	**.96**	.90	**.93**	**.07**	**.81**	.37	**.51**	**.49**	**.91**	.56	.70	.31
	GDFA	.84	**.86**	.85	.15	.80	**.75**	**.77**	**.23**	.68	**.55**	**.61**	**.39**	.91	**.94**	**.93**	.08	.61	**.44**	**.51**	**.49**	.81	**.63**	**.71**	**.29**
fast-align	Inters.	**.89**	.69	**.78**	**.22**	**.87**	.60	**.71**	**.29**	**.78**	.43	.55	.45	**.93**	.84	**.88**	**.11**	**.55**	.22	.31	.69	**.89**	.50	.64	.36
	GDFA	.71	**.81**	.76	.25	.70	**.73**	**.71**	**.29**	.60	**.54**	**.57**	**.43**	.81	**.93**	.86	.15	.34	**.33**	**.34**	**.66**	.69	**.67**	**.68**	**.33**
GIZA++	Inters.	**.95**	.60	.74	**.26**	**.92**	.62	.74	.26	**.89**	.26	.40	.60	**.97**	.89	**.93**	**.06**	**.82**	.25	.38	.62	**.95**	.47	.63	.37
	GDFA	.71	**.79**	**.75**	**.26**	.79	**.75**	**.77**	**.23**	.55	**.48**	**.51**	**.49**	.90	**.95**	.92	.09	.47	**.43**	**.45**	**.55**	.74	**.64**	**.69**	**.31**

Table 7: Comparison of symmetrization methods on word level. Best result across columns per method in bold.

System	Parameter	Value
fastText	Version	0.9.1
	Code URL	https://github.com/facebookresearch/fastText/archive/v0.9.1.zip
	Downloaded on	11.11.2019
	Embedding Dimension	300
mBERT,XLM-R	Code: Huggingface Transformer	Version 2.3.1
	Maximum Sequence Length	128
fastalign	Code URL	https://github.com/clab/fast_align
	Git Hash	7c2bbca3d5d61ba4b0f634f098c4fcf63c1373e1
	Flags	-d -o -v
eflomal	Code URL	https://github.com/robertostling/eflomal
	Git Hash	9ef1ace1929c7687a4817ec6f75f47ee684f9aff
	Flags	–model 3
GIZA++	Code URL	http://web.archive.org/web/20100221051856/http://code.google.com/p/giza-pp
	Version	1.0.3
	Iterations	5 iter. HMM, 5 iter. Model 1, 5 iter. Model3, 5 iter. Model 4 (DEFAULT)
	p0	0.98
Vecmap	Code URL	https://github.com/artetxem/vecmap.git
	Git Hash	b82246f6c249633039f67fa6156e51d852bd73a3
	Manual Vocabulary Cutoff	500000

Table 8: Overview on hyperparameters. We only list parameters where we do **not** use default values.

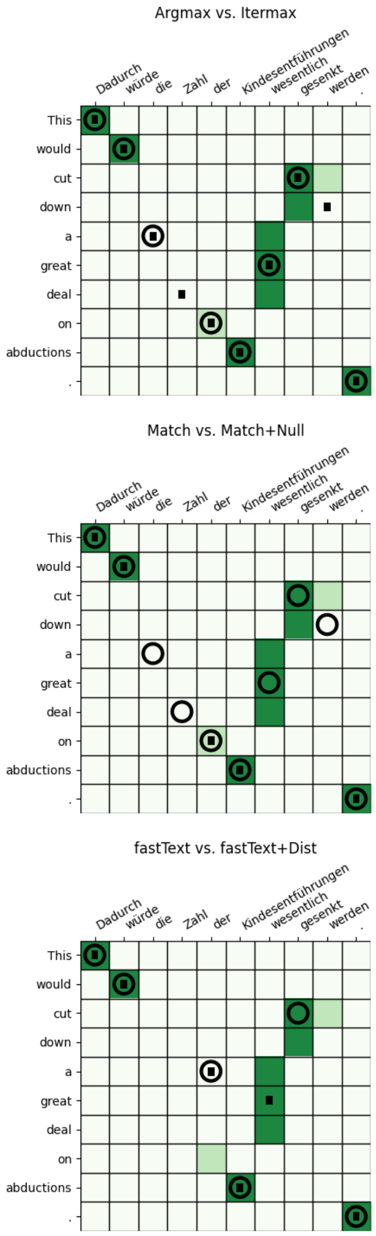

Figure 11: Comparison of alignment methods. Dark/light green: sure/possible edges in the gold standard. Circles are alignments from the first mentioned method in the subfigure title, boxes alignments from the second method.

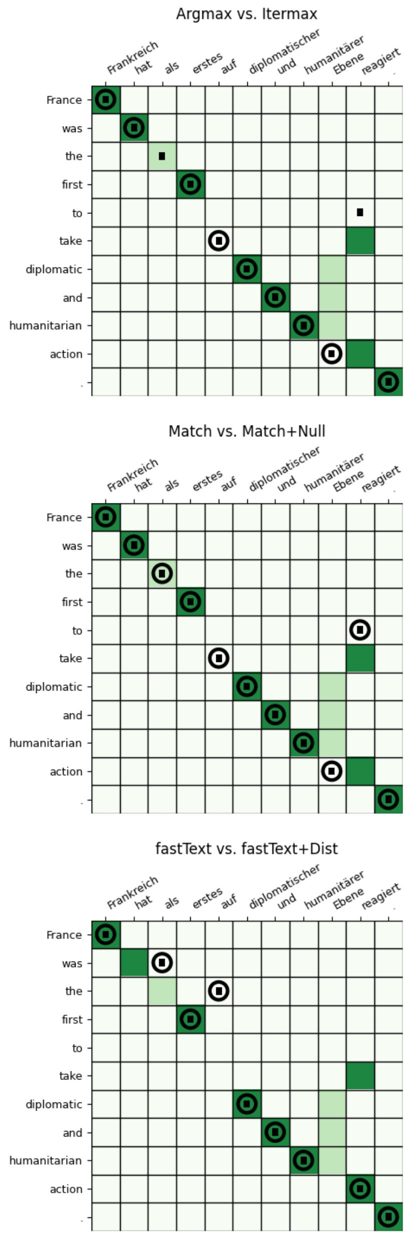

Figure 12: More examples.

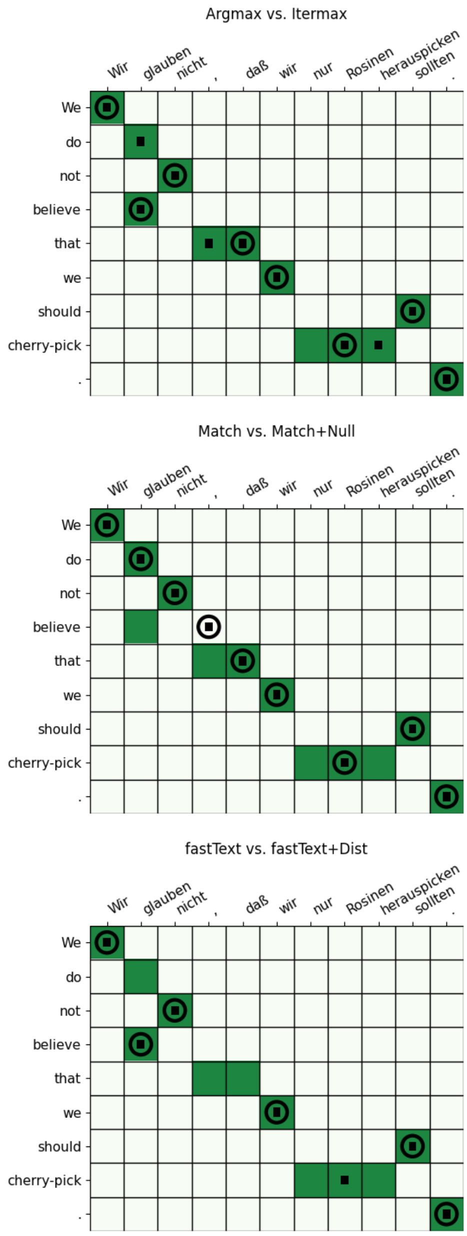

Figure 13: More examples.

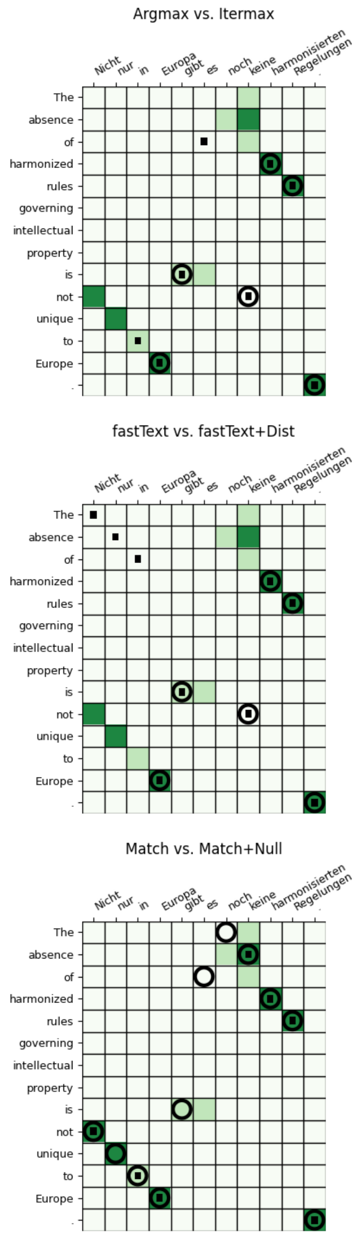

Figure 14: More examples.