# Graph Neural Networks for Multiparallel Word Alignment

**Anonymous ACL submission**

## Abstract

After a period of decrease, interest in word alignments is increasing again for their usefulness in domains such as typological research, cross-lingual annotation projection and machine translation. Generally, alignment algorithms only use bitext and do not make use of the fact that many parallel corpora are multiparallel. Here, we compute high-quality word alignments between multiple language pairs by considering all language pairs together. First, we create a multiparallel word alignment graph, joining all bilingual word alignment pairs in one graph. Next, we use graph neural networks (GNNs) and community detection algorithms to exploit the graph structure. Our GNN approach (i) utilizes information about the meaning, position and language of the input words, (ii) incorporates information from multiple parallel sentences, (iii) adds and removes edges from the initial alignments, and (iv) provides a prediction model that can generalize beyond the sentences it is trained on. We show that community detection provides valuable information for multiparallel word alignment. Our method outperforms previous work on three word alignment datasets and on a downstream task.

## 1 Introduction

Word alignments are crucial for statistical machine translation (Koehn et al., 2003) and useful for many other multilingual tasks such as neural machine translation (Alkhouli and Ney, 2017; Alkhouli et al., 2016), typological analysis (Lewis and Xia, 2008; Östling, 2015; Asgari and Schütze, 2017), annotation projection (Yarowsky and Ngai, 2001; Fossum and Abney, 2005; Wisniewski et al., 2014; Huck et al., 2019). The rise of deep learning initially led to a temporary plateau, but interest in word alignments is now increasing, demonstrated by several recent publications (Jalili Sabet et al., 2020; Chen et al., 2020; Dou and Neubig, 2021)

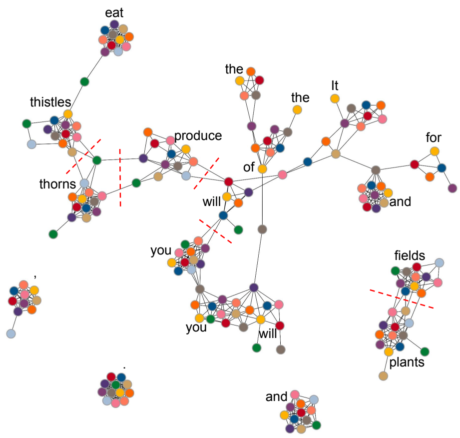



Figure 1: Alignment graph for the verse "It will produce thorns and thistles for you, and you will eat the plants of the field." in a 12-way multiparallel corpus. Colors represent languages. Each English (yellow) node is annotated with its word. Red dashed lines sever links that incorrectly connect distinct concepts. We exploit community detection algorithms to detect distinct concepts. This provides valuable information for our GNN model and improves word alignments.

Multiparallel corpora contain sentence level parallel text in more than two languages, e.g., JW300 (Agić and Vulić, 2019), PBC (Mayer and Cysouw, 2014) and Tatoeba.[1] While the amount of data provided by multiparallel corpora is less than bilingual corpora, this type of corpus is essential to study very low-resource languages. There are thousands of languages in the world a very small portion of which is covered by language technologies (Joshi et al., 2020). Recent work (Bird, 2020) suggests a number of approaches to develop technologies for indigenous languages. Multiparallel corpora are a valuable (and arguably complementary) resource for this aim. We use the PBC corpus since it covers more than 1300 languages.

Most prior work on word alignment uses bitext, with one notable exception: (Imani et al., 2021).

[1] https://tatoeba.org

They introduce MPWA (MultiParallel Word Alignment), a framework that utilizes the synergy between multiple language pairs to improve bilingual word alignments. The rationale is that some of the missing alignment edges between a source and a target language can be recovered using their alignments with words in other languages.

The first step in MPWA is to create bilingual alignments for all language pairs in a multiparallel corpus using a bilingual word aligner. Then the bilingual alignments for a multiparallel sentence are represented as a graph where words are nodes and initial word alignments are edges. Figure 1 gives an example: a multiparallel alignment graph for a 12-way multiparallel corpus. MPWA infers missing alignment links based on the graph structure in a postprocessing step, casting the word alignment task as an edge prediction problem. They use two traditional graph algorithms, Adamic-Adar and non-negative matrix factorization, for edge prediction. However, these standard graph algorithms are applied to individual multiparallel sentences independently and therefore cannot accumulate knowledge from multiple sentences. Moreover, their edge predictions are solely based on the structure of the graph and do not take advantage of other beneficial signals such as a word's language, relative position and word meaning. Another limitation is that it only adds links and does not remove any, which is important to improve precision.

In this paper, we propose to use graph neural networks (GNNs) to exploit the graph structure of multiparallel word alignments and address the limitations of prior work. GNNs were proposed to extend the powerful current generation of neural network models to processing graph-structured data (Scarselli et al., 2009) and they have gained increasing popularity in many domains (Wu et al., 2020; Sanchez-Gonzalez et al., 2018; He et al., 2020). In contrast to other graph algorithms, GNNs can incorporate heterogeneous sources of signal in the form of node and edge features.

Since the nodes in the graph are words that are translations of each other, we expect them to create densely connected regions or *communities*. Our analysis of the structure of the multiparallel alignment graph confirms this intuition; see Figure 1. We use community detection algorithms to find communities. We show that pruning inter-community edges and adding intra-community edges is helpful. We use community information as node features for our GNN.

We enable the removal of alignment edges from initial alignments by inferring alignments from the alignment probability matrix. Our method predicts new alignment links independently of initial edges. Therefore it is not limited to adding edges wrt initial bilingual alignments, it can also remove them.

For our experiments, we follow the setup of Imani et al. (2021). We train a GNN model with a link prediction objective. We show improved results for three language pairs on word alignment (English-French, Finnish-Hebrew and Finnish-Greek). As a demonstration of the importance of high-quality alignments, we use our word alignments to project annotations from high-resource to low-resource languages. We improve a part-of-speech tagger for Yoruba by training it over a high-quality dataset, which is created using annotation projection. We show that our model is especially helpful for distant languages.

**Contributions: i)** We propose a graph neural network model that incorporates a diverse set of features for word alignments in multiparallel corpora and an elegant way of training it efficiently and effectively. **ii)** We show that community detection improves multiparallel word alignment. **iii)** We show that the improved alignments improve performance on a downstream task for a low resource language. **iv)** We propose a new method to infer alignments from the alignment probability matrix. **v)** We will make our code publicly available.

## 2 Graph Analysis with Community Detection (CD)

The nodes in the alignment graph are words that are translations of each other. If the initial bilingual alignments are of good quality, we expect these translated words to form densely connected regions or *communities*; see Figure 1. We expect these communities to be genarally disconnected, each corresponding to a distinct connected component. In other words, ideally, words representing a concept should be densely connected, but there should be no links between different concepts. Clearly, this intuition will not be true for all concepts between all possible language pairs. Nonetheless, we hypothesize that identifying distinct concepts in a multiparallel word alignment graph can provide useful information.

To examine to what extent this expectation is met, we count the components in the original

Eflomal-generated (Östling and Tiedemann, 2016) graph. Table 1 shows that the average number of components per sentence is less than three ("Eflomal intersection", columns #CC). But intuitively, the number of components should roughly correspond to sentence length (i.e., the number of content words). This indicates that there are many links that incorrectly connect different concepts. To detect such links, we use community detection (CD) algorithms.

CD algorithms find subnetworks of nodes that form tightly knit groups that are only loosely connected with a small number of links (Girvan and Newman, 2002). CD algorithms maximize the modularity measure (Newman and Girvan, 2004). Modularity measures how beneficial a division of a community into two communities is, in the sense that there are many links within communities and only a few between them. Given a graph $G$ with $n$ nodes and $m$ edges and $G$'s adjacency matrix $A \in \mathbb{R}^{n \times n}$, modularity is defined as:

$$mod = \frac{1}{2m} \sum_{ij} \left( A_{ij} - \gamma \frac{d_i d_j}{2m} \right) I(c_i, c_j) \quad (1)$$

$d_i$ is the degree of node $i$. $I(c_i, c_j)$ is 1 if nodes $i$ and $j$ are in the same community, 0 otherwise.

We experiment with two CD algorithms:

- Greedy modularity communities (GMC). This method uses Clauset-Newman-Moore greedy modularity maximization (Clauset et al., 2004). GMC begins with each node in its own community and greedily joins the pair of communities that most increases modularity until no such pair exists.

- Label propagation communities (LPC). This method finds communities in a graph using label propagation (Cordasco and Gargano, 2010). It begins by giving a label to each node of the network. Then each node's label is updated by the most frequent label among its neighbors in each iteration. It performs label propagation on a portion of nodes at each step and quickly converges to a stable labeling.

After detecting communities, we link all nodes inside a community and remove all intercommunity links. GMC (LPC) on average removes 3% (7%) of the edges. Table 1 reports the average number of graph components per sentence before and after runing GMC and LPC, as well as the corresponding $F_1$ for word alignment. We see that the number of communities found is lower for GMC than for LPC; therefore, LPC identifies more candidate links for deletion.[2] Comparing the number of communities detected with the average sentence length, GMC seems to have failed to detect enough communities to split different concepts properly. The $F_1$ scores confirm this observation and show that LPC performs well at detecting the communities we are looking for.

| | FIN-HEB | | FIN-GRC | | ENG-FRA | |
|---|---|---|---|---|---|---|
| | #CC | $F_1$ | #CC | $F_1$ | #CC | $F_1$ |
| Eflomal intersection | 2.2 | 0.404 | 1.6 | 0.646 | 2.2 | 0.678 |
| GMC | 13.7 | 0.396 | 10.1 | 0.375 | 13.5 | 0.411 |
| LPC | 41.5 | 0.713 | 37.1 | 0.754 | 46.0 | 0.767 |
| Sentence length | 25.7 | | 23.2 | | 27.4 | |

Table 1: Effect of community detection algorithms on alignment prediction. #CC: average number of connected components. $F_1$: word alignment performance.

These results indicate that CD algorithms can provide valuable information. To exploit this in our GNN model, we add a node's community information as a GNN node feature; see §3.1.2.

## 3 Methods

### 3.1 GNN in MPWA

GNNs can be used in transductive or inductive settings. Transductively, the final model can only be used for inference over the same graph that it is trained on. In an inductive setting, which we use here, nodes are represented as feature vectors, and the final model has the advantage of being applicable to a different graph in inference.

#### 3.1.1 Model Architecture

Our model is inspired by the Graph Auto Encoder (GAE) model of Kipf and Welling (2016b) for link prediction. The architecture consists of an encoder and a decoder. We make changes to this model to improve the model's quality and reduce its computation cost. We use GATConv layers (Veličković et al., 2018) for encoder instead of GCNConv (Kipf and Welling, 2016a)and a more sophisticated decoder instead of simple dot product for a stronger model. We also introduce a more efficient training procedure.

The **encoder** is a graph attention network (GAT) (Veličković et al., 2018) with two GATConv layers

[2]LPC may detect more communities than average sentence length because of null words: words that have no translation in the other languages, giving rise to separate communities.

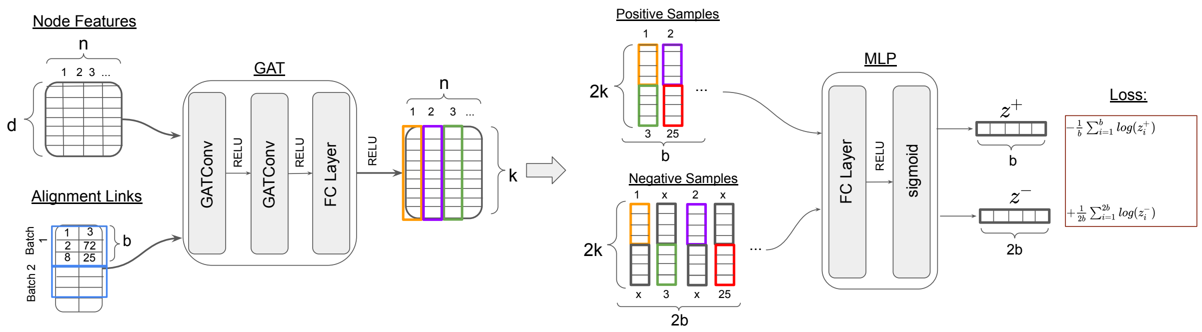



Figure 2: GNN training. At each training step, node features and links of a multiparallel sentence are fed to a graph attention network (GAT) that creates hidden representations for all nodes. On the decoder side, at each step, one batch of alignment links and hidden node representations is used to create positive and negative samples, which are then processed and classified by a multi-layer perceptron (MLP). Parameters of GAT and MLP are updated for each batch. FC = fully connected.

followed by a fully connected layer. Layers are connected by RELU non-linearities. A GATConv layer computes its output $\mathbf{x}'_i$ for a node $i$ from its input $\mathbf{x}_i$ as

$$\mathbf{x}'_i = \alpha_{i,i}\mathbf{W}\mathbf{x}_i + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j}\mathbf{W}\mathbf{x}_j, \tag{2}$$

where $\mathbf{W}$ is a weight matrix, $\mathcal{N}(i)$ is some *neighborhood* of node $i$ in the graph, and $\alpha_{i,j}$ is the attention coefficient indicating the importance of node $j$'s features to node $i$. $\alpha_{i,j}$ is computed as

$$\alpha_{i,j} = \frac{\exp\left(\mathrm{g}\left(\mathbf{a}^\top[\mathbf{W}\mathbf{x}_i \parallel \mathbf{W}\mathbf{x}_j]\right)\right)}{\sum_{k \in \mathcal{N}(i)\cup\{i\}} \exp\left(\mathrm{g}\left(\mathbf{a}^\top[\mathbf{W}\mathbf{x}_i \parallel \mathbf{W}\mathbf{x}_k]\right)\right)} \tag{3}$$

where $\parallel$ is concatanation, $g$ is LeakyReLU, and $\mathbf{a}$ is a weight vector. Given the features for the nodes and their alignment edges, the encoder creates a contextualized hidden representation for each node.

Based on the hidden representations of two nodes, the **decoder** predicts whether a link connects them. The decoder architecture consists of a fully connected layer, a RELU non-linearity and a sigmoid layer.

**Training.** By default, GAE models are trained using full batches with random negative samples. This approach requires at least tens of epochs over training dataset to converge and a lot of GPU memory for graphs as big as ours. We train our model using mini-batches and an adversarial loss to decrease memory requirements and improve the performance. Using our training approach the model converges after one epoch. The negative samples are selected more elegantly, as described below. Figure 2 displays our GNN model and the training process. The outer loop iterates over the multiparallel sentences in the training set. The training set contains one graph for each sentence; the graph is constructed using the bilingual alignment edges between all language pairs.

Each graph is divided into multiple batches. Each batch contains a random subset of the graph's edges as positive samples. The negative samples are created as follows. Given a sentence $u_1u_2\ldots u_n$ in language $U$ and its translation $v_1v_2\ldots v_m$ in language $V$, for each alignment edge $u_i$:$v_j$ in the current batch, two negative edges $u_i$:$v'_j$ and $u'_i$:$v_j$ ($j' \neq j$, $i' \neq i$) are randomly sampled.

For each training batch, the encoder takes the batch's whole graph (i.e., node features for all graph nodes and all graph edges) as input and computes hidden representations for the nodes. On the decoder side, for each link of the batch, the hidden representations of the attached nodes are concatenated to create the decoder's input. The decoder's target is the link's class: 1 (resp. 0) for positive (resp. negative) links. We train with a binary classification objective:

$$\mathcal{L} = -\frac{1}{b}\sum_{i=1}^{b}\log(p_i^+) + \frac{1}{2b}\sum_{i=1}^{2b}\log(p_i^-) \tag{4}$$

where $b$ is the batch size and $p_i^+$ and $p_i^-$ are the model predictions for the $i^{th}$ positive and negative samples within the batch. Parameters of the encoder and decoder as well as the node-embedding feature layer are updated after each training step.

### 3.1.2 Node Features

We use three main types of node features: (i) graph structural features, (ii) community-based features and (iii) word content features.

**Graph structural features.** We use *degree, closeness* (Freeman, 1978) , *betweenness* (Brandes, 2001) , *load* (Newman, 2001) and *harmonic centrality* (Boldi and Vigna, 2014) features as additional information about the graph structure. These features are continuous numbers, providing information about the position and connectivity of the nodes within the graph. We standardize (i.e., z-score) each feature across all nodes, and train an embedding of size four for each feature.[3]

**Community-based features.** One way to incorporate community information into our model is to train the model based on the refined edges after the community detection step. This approach hobbles the GNN model by making decisions about many of the edges before the GNN gets to see them. Our initial experiments also confirmed that training the GNN over CD refined edges does not help. Therefore, we add community information as node features and let the GNN use them to improve its decisions. We use the community detection algorithms GMC and LPC (see §2) to identify communities in the graph. Then we take the community membership information of the nodes as one-hot vectors and learn an embedding of size 32 for each of the two algorithms.

**Word content features.** We train embeddings for *word position* (size 32) and *word language* (size 20). We learn 100-dimensional multilingual *word embeddings* using Levy et al. (2017)'s sentence-ID method on the 84 PBC languages selected by Imani et al. (2021). Word embeddings serve as initialization and are updated during GNN training.

After concatenating these features, each node is represented by a 236 dimensional vector that is then fed to the encoder.

#### 3.1.3 Inducing Alignment Edges

When our trained GNN model is used to predict alignment edges between a source sentence $\hat{x} = x_1, x_2, \ldots, x_m$ in language $X$ and a target sentence $\hat{y} = y_1, y_2, \ldots, y_l$ in language $Y$, it produces a symmetric alignment probability matrix $S$[4] of size $m \times l$ where $S_{ij}$ is the predicted alignment probability between words $x_i$ and $y_j$. Using these values directly to infer alignment edges is usually suboptimal; therefore, more sophisticated methods have been suggested (Ayan and Dorr, 2006; Liang et al., 2006). Here we propose a new approach: it combines Koehn et al. (2005)'s Grow-Diag-Final-And (GDFA) with Dou and Neubig (2021)'s probability thresholding. We modify the latter to account for the variable size of the probability matrix (i.e., length of source/target sentences). Our method is not limited to adding new edges to some initial bilingual alignments, a limitation of prior work. As we predict each edge independently, some initial links can be discarded from the final alignment.

We start by creating a set of *forward* (source-to-target) alignment edges and a set of *backward* (target-to-source) alignment edges. To this end, first, inspired by probability thresholding (Dou and Neubig, 2021), we apply softmax to $S$, and zero out probabilities below a threshold to get a source-to-target probability matrix $S^{XY}$:

$$S^{XY} = S * (\text{softmax}(S) > \frac{\alpha}{l}) \qquad (5)$$

Analogously, we compute the target-to-source probability matrix $S^{YX}$:

$$S^{YX} = S^\top * (\text{softmax}(S^\top) > \frac{\alpha}{m}) \qquad (6)$$

where $\alpha$ is a sensitivity hyperparameter, e.g., $\alpha = 1$ means that we pick edges with a probability higher than average. We experimentally set $\alpha = 2$. Next, from each row of $S^{XY}$ ($S^{YX}$), we pick the cell with the highest value (if any exists) and add this edge to the *forward* (*backward*) set.

We create the final set of alignment edges by applying the GDFA symmetrization method (Koehn et al., 2005) to *forward* and *backward* sets. The gist of GDFA is to use the intersection of *forward* and *backward* as initial alignment edges and add more edges from the union of *forward* and *backward* based on a number of heuristics. We call this method *TGDFA* (Thresholding GDFA).

We also experiment with combining TGDFA with the original bilingual GDFA alignments. We do so by adding bilingual GDFA edges to the union of *forward* and *backward* before performing the GDFA heuristics. We refer to these alignments as *TGDFA+orig*.

We evaluate the resulting alignments using $F_1$ score and alignment error rate (AER), the standard evaluation measures in the word alignment literature.

[3]Learning a size-four embedding instead of a single number gives the feature a weight similar to other features – which have a feature vector of about the same size.

[4]For inference, we feed all possible alignment links between source and target to the decoder.

| Method | FIN-HEB | | | | FIN-GRC | | | | ENG-FRA | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Prec. | Rec. | $F_1$ | AER | Prec. | Rec. | $F_1$ | AER | Prec. | Rec. | $F_1$ | AER |
| Eflomal (intersection) | **0.818** | 0.269 | 0.405 | 0.595 | **0.897** | 0.506 | 0.647 | 0.353 | **0.971** | 0.521 | 0.678 | 0.261 |
| Eflomal (GDFA) | 0.508 | 0.448 | 0.476 | 0.524 | 0.733 | 0.671 | 0.701 | 0.300 | 0.856 | 0.710 | 0.776 | 0.221 |
| WAdAd (intersection) | 0.781 | 0.612 | 0.686 | 0.314 | 0.849 | 0.696 | 0.765 | 0.235 | 0.938 | 0.689 | 0.794 | 0.203 |
| NMF (intersection) | 0.780 | 0.576 | 0.663 | 0.337 | 0.864 | 0.669 | 0.754 | 0.248 | 0.948 | 0.624 | 0.753 | 0.245 |
| WAdAd (GDFA) | 0.546 | **0.693** | 0.611 | 0.389 | 0.707 | **0.783** | 0.743 | 0.257 | 0.831 | **0.796** | 0.813 | 0.186 |
| NMF (GDFA) | 0.548 | 0.646 | 0.593 | 0.407 | 0.720 | 0.759 | 0.739 | 0.261 | 0.844 | 0.767 | 0.804 | 0.195 |
| GNN (TGDFA) | 0.811 | 0.648 | **0.720** | **0.280** | 0.845 | 0.724 | **0.780** | **0.220** | 0.926 | 0.711 | 0.804 | 0.192 |
| GNN (TGDFA+orig) | 0.622 | 0.683 | 0.651 | 0.349 | 0.738 | 0.780 | 0.758 | 0.242 | 0.863 | 0.789 | **0.824** | **0.174** |

Table 2: Word alignment results on PBC for GNN and baselines. The best result in each column is in bold. GNN outperforms the baselines as well as the graph algorithms WAdAd and NMF on $F_1$ and AER.

### 3.2 Annotation Projection

Annotation projection automatically creates linguistically annotated corpora for low-resource languages. A model trained on data with "annotation-projected" labels can perform better than full unsupervision. Here, we focus on universal part-of-speech (UPOS) tagging (Petrov et al., 2012) for the low resource target language Yoruba; this language only has a small set of annotated sentences in Universal Dependencies (Nivre et al., 2020) and has poor POS results in unsupervised settings (Kondratyuk and Straka, 2019).

The quality of the target annotated corpus depends on the quality of the annotations in the source languages and the quality of the word alignments between sources and target. We use the Flair (Akbik et al., 2019) POS taggers for three high resource languages, English, German and French (Akbik et al., 2018), to annotate 30K verses whose Yoruba translations are available in PBC. We then transfer the POS tags from source to target using three different approaches: (i) We directly transfer annotations from English to the target. (ii) For each word in the target, we get its alignments in the three source languages and predict the majority POS to annotate the target word. (iii) We repeat (ii) using alignments from our GNN (TGDFA) model instead of the original bilingual alignments. In all three approaches, we discard any target sentence from the POS tagger training data if more than 50% of its words are annotated with the "X" (other) tag.

We train a Flair SequenceTagger model on the target annotated data using mBERT embeddings (Devlin et al., 2019) and evaluate on Yoruba test from Universal Dependencies.[5]

[5]https://universaldependencies.org/

## 4 Experimental Setup

### 4.1 Word Alignment Datasets

Following Imani et al. (2021), we use PBC, a multiparallel corpus of 1758 sentence-aligned editions of the Bible in 1334 languages.

**Evaluation data.** For our main evaluation, we use the two word alignment gold datasets for PBC published by Imani et al. (2021): Blinker (Melamed, 1998) and HELFI (Yli-Jyrä et al., 2020). **The HELFI dataset** contains the Hebrew Bible, Greek New Testament and their translations into Finnish. For HELFI, we use Imani et al. (2021)'s train/dev/test splits. **The Blinker dataset** provides word level alignments between English and French for 250 Bible verses.

**Training data.** The graph algorithms used by Imani et al. (2021) operate on each multiparallel sentence separately. In contrast, our approach allows for an inductive setting where a model is trained on a training set and then evaluated on a separate test set. We combine the verses in the training sets of Finnish-Hebrew and Finnish-Greek for a combined train set size of 24,159.

### 4.2 Initial Word Alignments

We use the Eflomal statistical word aligner to obtain bilingual alignments. We train it for every language pair in our experiments. We do not consider SimAlign (Jalili Sabet et al., 2020) since it is shown to perform poorly for languages whose representations in the multilingual pretrained language model are of low quality. We use Eflomal asymmetrical alignments post-processed with the intersection heuristic to get high precision bilingual alignments as input to the GNN. We use the same subset of 84 languages as Imani et al. (2021).

### 4.3 Training Details

We use PyTorch Geometric[6] to construct and train the GNN. The model's hidden layer size is 512 for both GATConv and Linear layers. We train for one epoch on the train set – a small portion of the train set is enough to learn good embeddings (see §5.1.1). For training, we use a batch size of 400 and learning rate of .001 with AdamW (Loshchilov and Hutter, 2017). The whole training process takes less than 4 hours on a GeForce GTX 1080 Ti and the inference time is on the order of milliseconds per sentence.

## 5 Experiments and Results

### 5.1 Multiparallel corpus results

Table 2 shows results on Blinker and HELFI for our GNNs and the baselines: bilingual alignments and the traditional graph algorithms WAdAd and NMF from Imani et al. (2021). Our GNNs provide a better trade-off between precision and recall, most likely thanks to their ability to remove edges, and achieve the best $F_1$ and AER on all three datasets, outperforming WAdAd and NMF.

GNN (TGDFA) achieves the best results on HELFI (FIN-HEB, FIN-GRC) while GNN (TGDFA+orig) is best on Blinker (ENG-FRA). As argued in Imani et al. (2021), this is mostly due to the different ways these two datasets were annotated. Most HELFI alignments are one-to-one, while many Blinker alignments are many-to-many: phrase-level alignments where every word in a source phrase is aligned with every word in a target phrase. This suggests that one can choose between GNN (TGDFA) and GNN (TGDFA+orig) based on the characteristics of the desired alignments.

#### 5.1.1 Effect of Training Set Size

To investigate the effect of training set size, we train the GNN on subsets of our training data with increasing sizes. Figure 3 shows results. Performance improves fast until around 2,000 verses; then it stays mostly constant. Indeed, using more than 6,400 samples does not change the performance at all. Therefore, in the other experiments we use 6,400 randomly sampled verses from the training set to train GNNs.

#### 5.1.2 Ablation Experiments

To examine the importance of node features, we ablate language, position, centrality, community and word embedding features. Table 3 shows that removal of graph structural features drastically reduces performance. Community features and language information are also important. Removal of word position information and word embeddings – which store semantic information about words – has the least effect. Based on these results, it can be argued that the lexical information contained in the initial alignments and in the community features provides a strong signal regarding word relatedness. The novel information that is crucial is about the overall graph structure which goes beyond the local word associations that are captured by word position and word embeddings.

#### 5.1.3 Effect of Word Frequency

We investigate the effect of word frequency on alignment performance where frequency is calculated based on the source word in the PBC; the first bin has the highest frequency. Figure 4 shows that the performance of Eflomal drops with frequency and it struggles to align very rare words. In contrast, GNN is not affected by word frequency as severely and its performance gains are even greater for rare words. WAdad which is the multilingual baseline from (Imani et al., 2021) has the same trend as GNN method, but GNN is more robust.

### 5.2 Annotation Projection

Table 4 presents accuracies for POS tagging in Yoruba. Unsupervised baseline performance is 50.86%. Supervised training using pseudo-labels mostly outperforms the unsupervised baseline. Projecting the majority POS labels to Yoruba improves over projecting English labels. Using the GNN model to project labels works best and outperforms

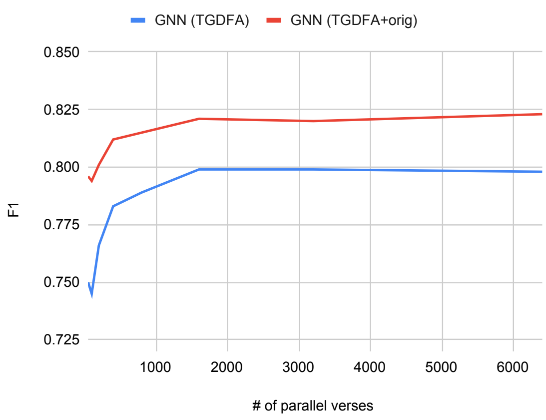
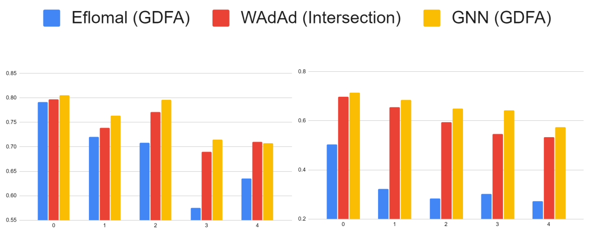



Figure 3: $F_1$ of GNN (TGDFA) and GNN (TGDFA+orig) on Blinker as a function of train size

[6] pytorch-geometric.readthedocs.io




(a) ENG-FRA (b) FIN-HEB

Figure 4: $F_1$ for different frequency bins.

| | FIN-HEB | FIN-GRC | ENG-FRA |
|---|---|---|---|
| GNN (TGDFA) | 0.720 | 0.780 | 0.804 |
| ¬ language | -0.323 | -0.280 | -0.370 |
| ¬ position | -0.068 | -0.045 | -0.066 |
| ¬ centrality | -0.636 | -0.730 | -0.772 |
| ¬ community | -0.204 | -0.238 | -0.253 |
| ¬ word-embedding | -0.139 | -0.103 | -0.129 |
| GNN (TGDFA+orig) | 0.651 | 0.758 | 0.824 |
| ¬ language | -0.238 | -0.077 | -0.162 |
| ¬ position | -0.088 | +0.029 | -0.032 |
| ¬ centrality | -0.556 | -0.530 | -0.617 |
| ¬ community | -0.156 | -0.039 | -0.083 |
| ¬ word-embedding | -0.135 | +0.002 | -0.058 |

Table 3: $F_1$ for GNNs and $\Delta F_1$ for five ablations

Eflomal-GDFA-majority (the unsupervised baseline) by 5% (15%) absolute improvement.

## 6 Related Work

**Bilingual Word Aligners.** Much work on bilingual word alignment is based on probabilistic models, mostly implementing variants of the IBM models of Brown et al. (1993): e.g., Giza++ (Och and Ney, 2003), fast-align (Dyer et al., 2013) and Eflomal (Östling and Tiedemann, 2016). More recent work, including SimAlign (Jalili Sabet et al., 2020) and SHIFT-ATT/SHIFT-AET (Chen et al., 2020), uses pretrained neural language and machine translation models. Although neural models achieve superior performance compared to statistical aligners, they are only applicable for less than two hundred high-resource languages that are supported by multilingual language models like BERT (Devlin et al., 2019) and XLM-R (Conneau et al., 2020). This makes statistical models the only option for the majority of the world's languages.

**Multiparallel Corpora.** Prior applications of using multiparallel corpora include reliable translations from small datasets (Cohn and Lapata, 2007), and phrase-based machine translation (PBMT) (Kumar et al., 2007). Multiparallel corpora are also used for language comparison (Mayer and Cysouw, 2012), typological studies (Östling, 2015; Asgari and Schütze, 2017) and PBMT (Nakov and Ng, 2012; Bertoldi et al., 2008; Dyer et al., 2013).

| Model | Yoruba YTB |
|---|---|
| Unsupervised (Kondratyuk and Straka, 2019) | 50.86 |
| Eflomal Inter - eng | 43.45 |
| Eflomal GDFA - eng | 55.13 |
| Eflomal Inter - majority | 54.13 |
| Eflomal GDFA - majority | 60.27 |
| GNN (TGDFA) - majority | **65.74** |
| GNN (TGDFA+orig) - majority | 64.55 |

Table 4: POS tagging with annotation projection for Yoruba. Apart from "Unsupervised", all lines show a sequence tagger trained on pseudo-labels induced by word alignments. GNN-based pseudo-labels outperform prior work by 5%.

To the best of our knowledge Östling (2014)[7] is the only word alignment method designed for multiparallel corpora. However, this method is outperformed by Eflomal (Östling and Tiedemann, 2016), a "biparallel" method from the same author. Recently, Imani et al. (2021) proposed MPWA, which we use as our baseline.

**Graph Neural Networks (GNNs)** have been used to address many problems that are inherently graph-like such as traffic networks, social networks, and physical and biological systems (Liu and Zhou, 2020). GNNs achieve impressive performance in many domains, including social networks (Wu et al., 2020) and natural science (Sanchez-Gonzalez et al., 2018) as well as NLP tasks like sentence classification (Huang et al., 2020), question generation (Pan et al., 2020), and summarization (Fernandes et al., 2019).

## 7 Conclusion and Future Work

We introduced graph neural networks and community detection algorithms for multiparallel word alignment. By incorporating signals from diverse sources as node features, including community features, our GNN model outperformed the baselines and prior work, establishing new state-of-the-art results on three PBC gold standard datasets. We also showed that our GNN model improves downstream task performance in low-resource languages through annotation projection.

We have only used node features to provide signals to GNNs. In the future, other signals can be added in the form of edge features to further boost the performance.

[7] github.com/robertostling/eflomal

## References


Željko Agić and Ivan Vulić. 2019. Jw300: A wide-coverage parallel corpus for low-resource languages. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3204–3210.

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, Minneapolis, Minnesota. Association for Computational Linguistics.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

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: Volume 1, Research Papers*, pages 54–65, Berlin, Germany. Association for Computational Linguistics.

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*, pages 108–117, Copenhagen, Denmark. Association for Computational Linguistics.

Ehsaneddin Asgari and Hinrich Schütze. 2017. Past, present, future: A computational investigation of the typology of tense in 1000 languages. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 113–124, Copenhagen, Denmark. Association for Computational Linguistics.

Necip Fazil Ayan and Bonnie J. Dorr. 2006. A maximum entropy approach to combining word alignments. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 96–103, New York City, USA. Association for Computational Linguistics.

Nicola Bertoldi, Madalina Barbaiani, Marcello Federico, and Roldano Cattoni. 2008. Phrase-based statistical machine translation with pivot languages. In *International Workshop on Spoken Language Translation (IWSLT) 2008*.

Steven Bird. 2020. Decolonising speech and language technology. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3504–3519.

Paolo Boldi and Sebastiano Vigna. 2014. Axioms for centrality. *Internet Mathematics*, 10(3-4):222–262.

Ulrik Brandes. 2001. A faster algorithm for betweenness centrality. *Journal of mathematical sociology*, 25(2):163–177.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Yun Chen, Yang Liu, Guanhua Chen, Xin Jiang, and Qun Liu. 2020. Accurate word alignment induction from neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 566–576, Online. Association for Computational Linguistics.

Aaron Clauset, Mark EJ Newman, and Cristopher Moore. 2004. Finding community structure in very large networks. *Physical review E*, 70(6):066111.

Trevor Cohn and Mirella Lapata. 2007. Machine translation by triangulation: Making effective use of multi-parallel corpora. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 728–735, Prague, Czech Republic. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Gennaro Cordasco and Luisa Gargano. 2010. Community detection via semi-synchronous label propagation algorithms. In *2010 IEEE international workshop on: business applications of social network analysis (BASNA)*, pages 1–8. IEEE.

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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Zi-Yi Dou and Graham Neubig. 2021. Word alignment by fine-tuning embeddings on parallel corpora. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2112–2128, Online. 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*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

Patrick Fernandes, Miltiadis Allamanis, and Marc Brockschmidt. 2019. Structured neural summarization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Victoria Fossum and Steven Abney. 2005. Automatically inducing a part-of-speech tagger by projecting from multiple source languages across aligned corpora. In *Second International Joint Conference on Natural Language Processing: Full Papers*.

Linton C Freeman. 1978. Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239.

Michelle Girvan and Mark EJ Newman. 2002. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. 2020. *LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation*, page 639–648. Association for Computing Machinery, New York, NY, USA.

Lianzhe Huang, Xin Sun, Sujian Li, Linhao Zhang, and Houfeng Wang. 2020. Syntax-aware graph attention network for aspect-level sentiment classification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 799–810, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Matthias Huck, Diana Dutka, and Alexander Fraser. 2019. Cross-lingual annotation projection is effective for neural part-of-speech tagging. In *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 223–233, Ann Arbor, Michigan. Association for Computational Linguistics.

Ayyoob Imani, Masoud Jalili Sabet, Lütfi Kerem Şenel, Philipp Dufter, François Yvon, and Hinrich Schütze. 2021. Graph algorithms for multiparallel word alignment.

Masoud Jalili Sabet, Philipp Dufter, François Yvon, and Hinrich Schütze. 2020. SimAlign: High quality word alignments without parallel training data using static and contextualized embeddings. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1627–1643, Online. Association for Computational Linguistics.

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. The state and fate of linguistic diversity and inclusion in the nlp world. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6282–6293.

Thomas N Kipf and Max Welling. 2016a. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Thomas N Kipf and Max Welling. 2016b. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.

Philipp Koehn, Amittai Axelrod, Alexandra Birch, Chris Callison-Burch, Miles Osborne, and David Talbot. 2005. Edinburgh system description for the 2005 iwslt speech translation evaluation. *International Workshop on Spoken Language Translation*.

Philipp Koehn, Franz J Och, and Daniel Marcu. 2003. Statistical phrase-based translation. Technical report, UNIVERSITY OF SOUTHERN CALIFORNIA MARINA DEL REY INFORMATION SCIENCES INST.

Dan Kondratyuk and Milan Straka. 2019. 75 languages, 1 model: Parsing Universal Dependencies universally. 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 2779–2795, Hong Kong, China. Association for Computational Linguistics.

Shankar Kumar, Franz J. Och, and Wolfgang Macherey. 2007. Improving word alignment with bridge languages. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 42–50, Prague, Czech Republic. Association for Computational Linguistics.

Omer Levy, Anders Søgaard, and Yoav Goldberg. 2017. A strong baseline for learning cross-lingual word embeddings from sentence alignments. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 765–774, Valencia, Spain. Association for Computational Linguistics.

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*.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 104–111, New York City, USA. Association for Computational Linguistics.

Zhiyuan Liu and Jie Zhou. 2020. Introduction to graph neural networks. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(2):1–127.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Thomas Mayer and Michael Cysouw. 2012. Language comparison through sparse multilingual word alignment. In *Proceedings of the EACL 2012 Joint Workshop of LINGVIS & UNCLH*, pages 54–62, Avignon, France. Association for Computational Linguistics.

Thomas Mayer and Michael Cysouw. 2014. Creating a massively parallel bible corpus. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 3158–3163.

I. Dan Melamed. 1998. Manual annotation of translational equivalence: The blinker project. *CoRR*, cmp-lg/9805005.

Preslav Nakov and Hwee Tou Ng. 2012. Improving statistical machine translation for a resource-poor language using related resource-rich languages. *Journal of Artificial Intelligence Research*, 44:179–222.

Mark EJ Newman. 2001. Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality. *Physical review E*, 64(1):016132.

Mark EJ Newman and Michelle Girvan. 2004. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France. European Language Resources Association.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Robert Östling. 2014. Bayesian word alignment for massively parallel texts. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 123–127, Gothenburg, Sweden. Association for Computational Linguistics.

Robert Östling. 2015. 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 (Volume 2: Short Papers)*, pages 205–211, Beijing, China. Association for Computational Linguistics.

Robert Östling and Jörg Tiedemann. 2016. Efficient word alignment with Markov Chain Monte Carlo. *The Prague Bulletin of Mathematical Linguistics*, 106(1).

Liangming Pan, Yuxi Xie, Yansong Feng, Tat-Seng Chua, and Min-Yen Kan. 2020. Semantic graphs for generating deep questions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1463–1475, Online. Association for Computational Linguistics.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2089–2096, Istanbul, Turkey. European Language Resources Association (ELRA).

Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. 2018. Graph networks as learnable physics engines for inference and control. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4470–4479. PMLR.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Guillaume Wisniewski, Nicolas Pécheux, Souhir Gahbiche-Braham, and François Yvon. 2014. Cross-lingual part-of-speech tagging through ambiguous learning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1779–1785, Doha, Qatar. Association for Computational Linguistics.

Yongji Wu, Defu Lian, Yiheng Xu, Le Wu, and Enhong Chen. 2020. Graph convolutional networks with markov random field reasoning for social spammer detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01):1054–1061.

David Yarowsky and Grace Ngai. 2001. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Second Meeting of the North American Chapter of the Association for Computational Linguistics*.

Anssi Yli-Jyrä, Josi Purhonen, Matti Liljeqvist, Arto Antturi, Pekka Nieminen, Kari M. Räntilä, and Valtter Luoto. 2020. HELFI: a Hebrew-Greek-Finnish parallel Bible corpus with cross-lingual morpheme alignment. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4229–4236, Marseille, France. European Language Resources Association.

# A Appendix

## A.1 Languages

| | | | | | |
|---|---|---|---|---|---|
| Afrikaans | Albanian | Arabic | Armenian | Azerbaijani | Bashkir |
| Basque | Belarusian | Bengali | Breton | Bulgarian | Burmese |
| Catalan | Cebuano | Chechen | Chinese | Chuvash | Croatian |
| Czech | Danish | Dutch | English | Estonian | Finnish |
| French | Georgian | German | Greek | Gujarati | Haitian |
| Hebrew | Hindi | Hungarian | Icelandic | Indonesian | Irish |
| Italian | Japanese | Javanese | Kannada | Kazakh | Kirghiz |
| Korean | Latin | Latvian | Lithuanian | Low Saxon | Macedonian |
| Malagasy | Malay | Malayalam | Marathi | Minangkabau | Nepali |
| Norwegian (Bokmal) | Norwegian (Nynorsk) | Punjabi | Persian | Polish | Portuguese |
| Punjabi | Romanian | Russian | Serbian | Slovak | Slovenian |
| Spanish | Swahili | Sundanese | Swedish | Tagalog | Tajik |
| Tamil | Tatar | Telugu | Turkish | Ukrainian | Urdu |
| Uzbek | Vietnamese | Waray-Waray | Welsh | West Frisian | Yoruba |

Table 5: List of the 84 languages we used in our experiments.