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. We propose to use graph neural networks (GNNs) and community detection algorithms to exploit the graph structure of multiparallel word alignments. Our GNN approach (i) utilizes information about the meaning, position and language of the input words, (ii) incorporates information from multiple parallel sentences, (iii) can remove 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 algorithms can provide valuable information for multiparallel word alignment. We show on three word alignment datasets and on a downstream task that our method outperforms previous work.

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; Wisniawski et al., 2014; Huck et al., 2019), bilingual lexicon induction (Lample et al., 2018; Artetxe et al., 2018; Shi et al., 2021) and creation of multilingual embeddings (Duffer et al., 2018). 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; Marchisio et al., 2021; Wu and Dredze, 2020; Imani et al., 2021).

Most prior work on word alignments uses bitext, but Imani et al. (2021) exploit the fact that many parallel corpora are multiparallel (i.e., they contain more than two parallel corpora). They introduce MPWA (MultiParallel Word Alignment), a framework that makes use of multiparallelism for better word alignments. They represent sets of bilingual word alignments as graphs and cast the word alignment task as an edge prediction problem. To exploit the graph structure, they apply two standard graph algorithms, Adamic-Adar (AdAd) and non-negative matrix factorization (NMF), and achieve improved results. 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 ad-
We use the community detection algorithms GMC (Hamaguchi et al., 2017), and recommender systems (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 the community detection algorithms GMC and LPC (see below) to find communities and show that pruning inter-community and adding intra-community edges is helpful. We use community information as node features for our GNN.

A limitation of (Imani et al., 2021) is that it only adds links and does not remove any. We address this by proposing a new method to infer alignments from the alignment probability matrix. Our method predicts new alignment links independently of initial edges. Therefore it is not limited to adding new edges to some initial bilingual alignments.

For our experiments, we follow the setup of Imani et al. (2021). We obtain bilingual alignments using the statistical word aligner Eflomal (Östling and Tiedemann, 2016). We train a GNN model on the resulting graph 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 languages to low-resource languages. We improve the performance of a part-of-speech tagger for the Yoruba language by training it over a high-quality dataset, which is created using annotation projection.

Contributions: i) We propose graph neural networks that can incorporate a diverse set of features for word alignments in multiparallel corpora and show that GNNs establish a new state of the art in word alignment. ii) We show that community detection algorithms improve multiparallel word alignment. iii) We show that the improved alignments improve performance on a downstream task. iv) We propose a new method to infer alignments from the alignment probability matrix. v) We will make our code publicly available.

2 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., the aligners Giza++ (Och and Ney, 2003), fast-align (Dyer et al., 2013) and Eflomal (Östling and Tiedemann, 2016). They use statistical similarities between word distributions in sentence aligned parallel corpora to learn word alignment models. More recent work, including SimAlign (Jalili Sabet et al., 2020), Awesome-align (Dou and Neubig, 2021), Bidir+CL (Zenkel 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. Due to its good performance, we use Eflomal as our initial bilingual aligner.

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).

Despite the usefulness of multiparallel corpora, most past work on word alignment has focused on bilingual corpora. Östling (2014)\footnote{github.com/robertostling/eflomal} proposed a word alignment method specifically designed for multiparallel corpora. However, this method is outperformed by Eflomal (Östling and Tiedemann, 2016), a “biparallel” method from the same author. Re-
cently, Imani et al. (2021) proposed MPWA (MultiParallel Word Alignment, see §3), which refines the graph structure of an initial multiparallel word alignment using standard graph algorithms. Our work improves on MPWA.

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). Their popularity increased rapidly after the efficient integration of powerful deep learning techniques, such as convolutional neural networks and attention mechanisms, into GNNs (Kipf and Welling, 2016, 2017; Velickovic et al., 2018).

GNNs achieve impressive performance in many domains, including social networks (Wu et al., 2020), natural science (Sanchez-Gonzalez et al., 2018), knowledge graphs (Hamaguchi et al., 2017), and recommender systems (He et al., 2020). This has motivated the NLP community to apply GNNs to tasks such as sentence classification (Huang et al., 2020), named entity recognition (Luo and Zhao, 2020), question generation (Pan et al., 2020) and summarization (Fernandes et al., 2019). As far as we know, our work is the first to apply GNNs to word alignment.

3 Background

3.1 MPWA

MPWA (MultiParallel Word Alignment) aims to utilize the synergy between multiple language pairs to improve bilingual word alignments (Imani et al., 2021). The rationale is that some of the missing alignment edges between a source and a target language can be recovered by 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 given multiparallel sentence are represented as a graph where words are nodes and initial word alignments are edges. Figure 1 gives an example: a bilingual alignment graph for a 12-way multiparallel corpus.

MPWA tries to infer missing alignment links based on the graph structure, casting the word alignment task as an edge prediction problem. Imani et al. (2021) use two traditional graph algorithms, Adamic-Adar and non-negative matrix factorization, for edge prediction. We replace them here with more powerful GNNs.

3.2 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 in the graph; see Figure 1. We expect these communities to be 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.

To examine to what extent this expectation is met, we count the components in the original (Effomal-generated) graph. Table 1 shows that, for most sentences, the average number of components per sentence is less than three. But intuitively, the number of components (representing the concepts in the sentence) should be roughly equal to sentence length (or at least 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). These algorithms try to maximise 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:

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

Where $d_i$ is the degree of node $i$, and $I(c_i, c_j)$ is 1 if nodes $i$ and $j$ are in the same community and 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
The table detection algorithms on alignment prediction. #CC: average number of connected components. $F_1$: word alignment performance. LPC consistently increases the number of components and increases $F_1$.

<table>
<thead>
<tr>
<th></th>
<th>FIN-HEB</th>
<th>FIN-GRC</th>
<th>ENG-FRA</th>
</tr>
</thead>
<tbody>
<tr>
<td>#CC</td>
<td>F1</td>
<td>#CC</td>
<td>F1</td>
</tr>
<tr>
<td>Eflomal intersection</td>
<td>2.2 0.404</td>
<td>1.6 0.646</td>
<td>2.2 0.678</td>
</tr>
<tr>
<td>GMC</td>
<td>13.7 0.396</td>
<td>10.1 0.375</td>
<td>13.5 0.411</td>
</tr>
<tr>
<td>LPC</td>
<td>41.5 0.713</td>
<td>37.1 0.754</td>
<td>46.0 0.767</td>
</tr>
<tr>
<td>Sentence length</td>
<td>25.7</td>
<td>23.2</td>
<td>27.4</td>
</tr>
</tbody>
</table>

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

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. LPC’s semi-synchronous algorithm, which at each step performs label propagation on a portion of nodes, quickly converges to a stable labeling. After detecting communities, we link all nodes inside a community and remove all inter-community 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 running 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.

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 of that node.

4 Methods

4.1 GNN in MPWA

GNNs can be used in transductive or inductive settings. In a transductive setting, nodes are represented as node IDs, and the final model can only be used for inference over the same graph that it is trained on. In an inductive setting, nodes are represented as feature vectors, and the final model has the advantage of being applicable to a different graph in inference. We use the inductive setting. GNNs can incorporate different sources of signal in the form of node and edge features. We only use node features. All are trained (or finetuned) during GNN training.

4.1.1 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 of these features across all nodes in the graph, and train an embedding of size four for each feature.3

Community-based features. We use the community detection algorithms GMC and LPC (see §3.2) 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.

4.1.2 Model Architecture

Our model is inspired by the Graph Auto Encoder (GAE) model of Kipf and Welling (2016) for the link prediction task. The architecture consists of an encoder and a decoder.

The encoder is a graph attention network (GAT) (Veličković et al., 2018) with two GATConv layers.
followed by a fully-connected layer. Layers are connected by RELU non-linearities. A GATConv layer computes its output $x'_i$ for a node $i$ from its input $x_i$ as

$$x'_i = \alpha_{i,i}Wx_i + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j}Wx_j,$$  \hspace{1cm} (2)

where $W$ 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

$$\alpha_{i,j} = \frac{\exp(g(a^T[Wx_i \mid Wx_j]))}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp(g(a^T[Wx_i \mid Wx_k]))}$$  \hspace{1cm} (3)

where $^T$ is transposition and $\mid$ is concatenation, $g$ is the LeakyReLU non-linearity, and $a$ 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.** 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 $u_1 u_2 u_3 \ldots u_n$ in language $U$ and its translation $v_1 v_2 v_3 \ldots v_m$ 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:

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

**4.1.3 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 an alignment probability matrix \[ S^1 \] 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 (G DFA) 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 \( S^{XY} \):

\[
S^{XY} = S \ast (\text{softmax}(S) > \frac{\alpha}{m})
\] (5)

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

\[
S^{YX} = S^\top \ast (\text{softmax}(S^\top) > \frac{\alpha}{m})
\] (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).

In addition to the TGDFA alignments, we also experiment with combining them 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.

### 4.2 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 language, 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 (TG DFA) 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 Experimental Setup

#### 5.1 Word Alignment Datasets

Following Imani et al. (2021), we use PBC as our multiparallel corpus. PBC contains 1758 editions of the Bible in 1334 languages, aligned at the verse level. A verse can contain more than one sentence, but we take it as one unit since sentence level alignments are not available.

For our main evaluation, we use the two word alignment gold datasets for PBC published by

---

5https://universaldependencies.org/
We evaluate on the same subset of 84 languages

<table>
<thead>
<tr>
<th>Method</th>
<th>FIN-HEB</th>
<th>FIN-GRC</th>
<th>ENG-FRA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prec.</td>
<td>Rec.</td>
<td>F1</td>
</tr>
<tr>
<td>Effomal (interaction)</td>
<td>0.818</td>
<td>0.269</td>
<td>0.405</td>
</tr>
<tr>
<td>Effomal (GDFA)</td>
<td>0.598</td>
<td>0.465</td>
<td>0.572</td>
</tr>
<tr>
<td>WAdAd (interaction)</td>
<td>0.781</td>
<td>0.612</td>
<td>0.686</td>
</tr>
<tr>
<td>NMF (intersection)</td>
<td>0.780</td>
<td>0.576</td>
<td>0.665</td>
</tr>
<tr>
<td>WAdAd (GDFA)</td>
<td>0.546</td>
<td>0.693</td>
<td>0.611</td>
</tr>
<tr>
<td>NMF (GDFA)</td>
<td>0.546</td>
<td>0.693</td>
<td>0.611</td>
</tr>
<tr>
<td>GNN (TDGFA)</td>
<td>0.811</td>
<td>0.648</td>
<td>0.729</td>
</tr>
<tr>
<td>GNN (TG DFA+orig)</td>
<td>0.622</td>
<td>0.683</td>
<td>0.651</td>
</tr>
</tbody>
</table>

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 F1 and AER.

5.3 Training Details

We use NetworkX\(^7\) for graph structural and community-based feature extraction; PyTorch Geometric\(^8\) to construct and train the GNN, and Gensim\(^9\) to train sentence-ID embeddings. 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 §6.1.1). For training, we use a batch size of 400 and learning rate of .001 with AdamW (Loshchilov and Hutter, 2017).

6 Experiments and Results

6.1 Multiparallel corpus results

Table 2 shows results on Blinker and HELFI for our GNNs 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 F1 and AER on all three datasets, outperforming WAdAd and NMF.

GNN (TG DFA) achieves the best results on HELFI (FIN-HEB, FIN-GRC) while GNN (TG DFA+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

\(^7\)https://networkx.org/
\(^8\)https://pytorch-geometric.readthedocs.io
\(^9\)https://radimrehurek.com/gensim/

---

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. The Finnish-Hebrew dataset has word level alignments for 22,291 verses and the Finnish-Greek dataset for 7,909. 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.

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. This allows our model to learn from multiple training samples and use its accumulated knowledge on the new test samples with fast inference. We combine the verses in training sets of Finnish-Hebrew and Finnish-Greek for a combined train set size of size 24,159\(^6\).

5.2 Initial Word Alignments

We use the Effomal statistical word aligner to obtain bilingual alignments. We do not consider SimA- lign (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, which includes the target languages in the HELFI dataset (Imani et al., 2021). We evaluate on the same subset of 84 languages as Imani et al. (2021). To train Effomal for a target language pair, we use all available translations; e.g., for a language pair with two and four different versions of the Bible, Effomal is trained on all eight translation pairs.

\(^6\)Note that we don’t use any gold alignments for training the GNN. We use these sets only to ensure that our training sentences are different than test sentences.
GNN (TG DFA) and GNN (TG DFA+orig) based on the characteristics of the desired alignments.

6.1.1 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.

6.2 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 very strong signal regarding words relatedness. The novel information that is crucial is about the overall graph structure which goes beyond local word associations, which are captured by word position and word embeddings.

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

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.

Table 3: \( F_1 \) for GNNs and \( \Delta F_1 \) for five ablations

<table>
<thead>
<tr>
<th>Model</th>
<th>FIN-HEB</th>
<th>FIN-GRC</th>
<th>ENG-FRA</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNN (TG DFA)</td>
<td>0.720</td>
<td>0.780</td>
<td>0.804</td>
</tr>
<tr>
<td>GNN (TG DFA+orig)</td>
<td>0.651</td>
<td>0.758</td>
<td>0.824</td>
</tr>
<tr>
<td>~ language</td>
<td>-0.323</td>
<td>-0.280</td>
<td>-0.370</td>
</tr>
<tr>
<td>~ position</td>
<td>-0.068</td>
<td>-0.045</td>
<td>-0.066</td>
</tr>
<tr>
<td>~ centrality</td>
<td>-0.636</td>
<td>-0.730</td>
<td>-0.772</td>
</tr>
<tr>
<td>~ community</td>
<td>-0.204</td>
<td>-0.238</td>
<td>-0.253</td>
</tr>
<tr>
<td>~ word-embedding</td>
<td>-0.139</td>
<td>-0.103</td>
<td>-0.129</td>
</tr>
</tbody>
</table>

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%.
References


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 (Volume 1:..


