Creation of Knowledge Graphs for Low-resource Language

Raunakk Banerjee, Suvarthi Sarkar, Kotha Ananya Reddy
Group No. 4
raunakk@iitg.ac.in
s.sarkar@iitg.ac.in
kotha18@iitg.ac.in

Abstract

In this paper, we have worked on two different research problems: performing cross-lingual Knowledge Graph (KG) alignment and performing entity-alignment between English and Hindi parallel corpora. We have analyzed different architectures performing cross-lingual KG alignment. Of all the models, graph convolutional networks(GCNs) are pretty intuitive due to their simple architecture and efficient output.We were able to implement crosslingual entity alignment successfully for english to chinese translation. Our goal was to do the same for a low resource Indian language (Hindi) but we were not able to find such entity aligned corpora for low resource Indian language and thus worked on making our own entity alignment from english to hindi by using Neural Machine Translation and Statistical Machine translational models like IBM Model 1 and Hidden Markov Model.

Keywords: Knowledge Graphs (KGs), Cross-lingual knowledge alignment, Graph Convolutional Networks(GCNs), Entity alignment, Hidden Markov Model, IBM Model

1 Introduction

Knowledge Graphs (KGs) have been built and applied in several domains, including question answering, recommendation systems, and information extraction. Most existing KGs are built separately using different data sources and languages due to which the same entity may exist in different KGs in different forms. On the other hand, KGs can be complementary to each other; knowledge about the same entity distributed in several KGs can help create connections spanning diverse languages. This aspect of knowledge graphs helps in handling the data sparsity of low resource languages. Low resource languages have a small corpora, which isn't sufficient enough to develop required knowledge bases for all kinds of AI applications. Hence, to handle this problem, cross-lingual knowledge graph alignment is used. Knowledge Graph (KG) alignment is done by discovering the mappings (i.e., equivalent entities, relations, and others) between two KGs. Conventionally, similarity propagation based methods [1] and probabilistic graph based methods [2] were used. These methods were dependent on machine translation technique or defining various language-independent features to discover cross-lingual links. Also, they can't be generalized to all the knowledge graphs and given the diversity of Indian languages, they are infeasible.

The concept of "attention" has gained popularity recently in training neural networks, allowing models to learn alignments between different modalities, e.g., between image objects and agent actions in the dynamic control problem, between speech frames and text in the speech recognition task , or between visual features of a picture and its text description in the image caption generation task. With simplicity and effectiveness in mind, two novel types of attention based models: a global approach in which all source words are attended and a local one whereby only a subset of source words are considered at a time.

To address the problem of bi-lingual word alignment, we used Statistical techniques like a first-order Hidden Markov model (HMM) and IBM Model 1 . The key component of HMM approach is to make the alignment probabilities dependent not on the absolute position of the word alignment, but on its relative position; i.e. we consider the differences in the index of the word positions rather than the index itself.

2 Literature Review

Given the increasing quality of output with intervention of neural networks, a number of neural network based knowledge graph embedding approaches have been proposed such as MTransE [3], JAPE [4], IPTransE [5] and GCN [6]. These approaches first embed entities in low-dimensional vector spaces, and then obtain the entity alignments by computations on their vector representations. Compared with traditional similarity-based approaches, embedding-based ones can effectively model different kinds of information in KGs and don't require any feature engineering.

MTransE learns a linear transformation between the embedding spaces of the KG using a loss func-

tion. IPTransE embeds both KGs into the same embedding space and uses a margin-based loss to enforce the embeddings of aligned entities to become similar. Unlike the aforementioned two, JAPE utilizes attribute information like the names of entities and relationships which further improve the results of KG alignment. So, with graph convolutional networks the equivalent relations between entities are modelled. With small model complexity and utilizing only pre-aligned entities dataset, GCNs result in best performance.

Like in knowledge graphs there have been significant works in Neural Machine Translation as well. Kalchbrenner and Blunsom (2013)[8] used an RNN with the standard hidden unit for the decoder and a convolutional neural network for encoding the source sentence representation. On the other hand, both Sutskever et al. (2014)[9] and Luong et al. (2015)[10] stacked multiple layers of an RNN with a Long Short-Term Memory (LSTM) hidden unit for both the encoder and the decoder. Cho et al. (2014)[11] adopted a different version of the RNN with an LSTM-inspired hidden unit, the gated recurrent unit (GRU), for both components. Entity alignment is the task of finding entities in two knowledge bases (KBs) that represent the same real-world object. When facing KBs in different natural languages, conventional cross-lingual entity alignment methods rely on machine translation to eliminate the language barriers. These approaches often suffer from the uneven quality of translations between languages.

Among the works done in Statistical Machine translation on word alignments, C. Cherry et al. [14] used the arithmetical based approach in which they improved the performance of alignment at sentence level by computing probability. This model permits easy integration of context-specific features. The drawback of this model is that it only works for one to one alignments only and not for many to one or many to many alignments.Eknath Venkataramani et al. [15] provided the corpus augmented method of word alignment. They used two GIZA++ tool which performs word alignment statistically and NATools for providing the bilingual dictionary. Hindi is a complex language and the resources are limited, above approach helps in eliminating these limitations. The parallel corpus under goes through 5 stages process and this method help in reducing the Alignment Error Rate (AER) approximately to 5.96%. When the method was applied after the POS tagging, the correct alignments of the adjective class and noun class increases whereas in the verb class the correct alignments slightly increase.

3 Methods

3.1 Knowledge Graph

• We used GCN to get vector embeddings for entities based on their alignment.
•Knowledge graphs of different languages from DBpedia/Yago have some pre-aligned entities which are used as training data.
•Using that, the model learns to produce similar vector embeddings for similar entities of different languages.
•The degree of alignment between two entities is given by a distance measure between the entities.
•The distance is expected to be small for equivalent entities and large for non-equivalent ones.
•For a specific entity e_i in G_1, our approach computes the distances between e_i and all the entities in G_2, and returns a list of ranked entities as candidate alignments.
•We use the DBP15K datasets from DBpedia. Each dataset contains data on two KGs in different languages and 15 thousand interlanguage links connecting equivalent entities in two KGs.

$$H_s^{(l+1)}; H_s^{(l+1)} = \sigma(\hat{D}^{-1/2}\hat{A}\hat{D}^{-\frac{1}{2}}[H_s^{(l)}W_s^{(l)}; H_a^{(l)}W_a^{(l)}])$$

$$\text{fun(r)}=\frac{\#Head\_Entitiesof\_r}{\#Triples\_of\_r}$$

$$\text{ifun(r)}=\frac{No.ofTail\_Entities\_of\_r}{No.ofTriples\_of\_r}$$

where Triples_of_r is the number of triples of relation r; Head_Entitiesof_r and Tail_Entitiesof_r are the numbers of head entities of r; respectively. To measure the influence of the i-th entity over the j-th entity; we set $a_{ij} \in A$ as: $a_{ij} = \sum_{(e_i,r,ej)\in G} ifun(r) + \sum_{(e_i,r,e_j)\in G} fun(r)$

Parameters	GCN1	GCN2
Initial Structure features	$H_{s1}^{(0)} \in R^{\|E1\| \times ds}$	$H_{s2}^{(0)} \in R^{\|E2\| \times ds}$
Weight Matrix for Structure Features in Layer 1	$W_s(1) \in R^{ds \times ds}$	$W_s(1) \in R^{ds \times ds}$
Weight Matrix for Structure Features in Layer 2	$W_s(2) \in R^{ds \times ds}$	$W_s(2) \in R^{ds \times ds}$
Output Structure Embeddings	$H_{s1}^{(2)} \in R^{\|E1\| \times A1}$	$H_{s2}^{(2)} \in R^{\|E2\| \times A2}$
Initial Attribute Feature Matrices	$H_{a1}^{(0)} \in R^{\|E1\| \times ds}$	$H_{a2}^{(0)} \in R^{\|E2\| \times ds}$
Weight Matrix for Attribute Features in Layer 1	$W_{a1}^{(1)} \in R^{\|A1\| \times da}$	$W_{a2}^{(1)} \in R^{\|A2\| \times da}$
Weight Matrix for Attribute Features in Layer 2	$W_a^{(2)} \in R^{da \times da}$	$W_a^{(2)} \in R^{da \times da}$
Output Attribute Embeddings	$H_{a1}^{(2)} \in R^{\|E1\| \times da}$	$H_{a2}^{(2)} \in R^{\|E2\| \times da}$

Table 2: The paper uses two 2-layer GCNs and each GCN processes one KG to generate embeddings of its entities. Consider 2 KGs given by $G_1 = (E_1, R_1, A_1, T_1^R, T_1^A)$ and $G_2 = (E_2, R_2, A_2, T_2^R, T_2^A)$. Let GCN1 and GCN2 be used for obtaining the embeddings for the 2 languages respectively.This approach assigns two feature vectors to each entity in GCN layers, structure feature vector hs and attribute feature vector ha. In the input layer, h(0)s is randomly initialized and updated during the training process; $h_a^{(0)}$ is the attribute vectors of entities and it is fixed during the model training. Because two KGs may have different number of attributes, the dimensionalities of the input attribute feature vectors in two GCN models are different (i.e. $|A1|$ not equal to $|A2|$). Embeddings for structure features and attribute features are calculated separately and are of d_s and d_a dimensional respectively. H_s and H_a represent the matrix of features for all the entities and W is the weights matrix.

3.2 Neural Machine Translation

Neural Machine Translation consists of 2 components a) Encoder b) Decoder. We use RNN to design such an Encoder or Decoder. It is an approach to machine translation that uses an artificial neural network to predict the likelihood of a sequence of words, typically modelling entire sentences in a single integrated model.

3.2.1 Global Attention Model

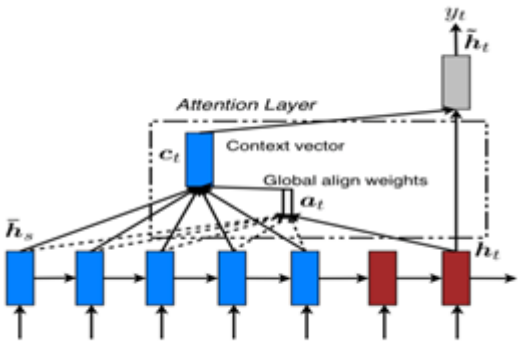

Figure 1: Global Attention Model

At each time step t, the model infers a variable-length alignment weight vector at based on the current target state h_t and all source states h_s. A global context vector c_t is then computed as the weighted average, according to a_t, over all the source states.

3.2.2 Local Attention Model

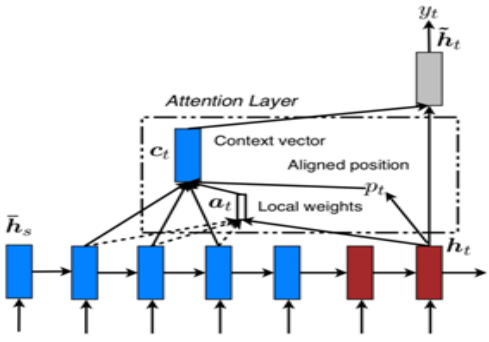

Figure 2: Local attention model

The model first predicts a single aligned position p_t for the current target word. A window centered around the source position p_t is then used to compute a context vector c_t, a weighted average of the source hidden states in the window. The weights at are inferred from the current target state h_t and those source states h_s in the window

3.2.3 Input feeding approach

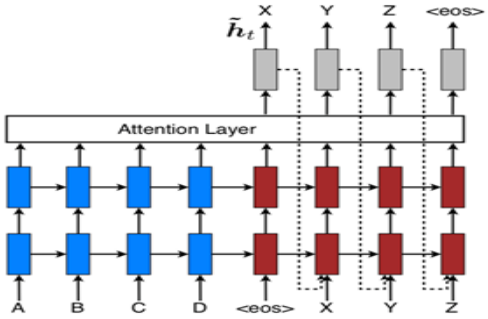

Figure 3: Input-feeding approach

In the above proposed local and global attention approaches the attentional decisions are made independently losing track of which source words got translated. So, we need to store past alignment information too and pass it on. To address this issue input feeding approach is suggested. Attentional vectors h t are fed as inputs to the next time steps to inform the model about past alignment decisions

3.3 Statistical Machine Translation model - Word Alignment

The word alignment problem is to figure out which words in the first corpus correspond to which ones in the second. There have been many methods for word alignment but the work in English-Hindi word alignment is still in development. The reason behind this is that the resources of Hindi languages are very less, and it is morphologically rich language We can find the translation model i.e. probability of f given e, where f is a foreign sentence and e is a sentence in English, by identifying alignments (correspondences) between words in f and words in e. In this report, one-one alignments are the focus. IBM Model 1 and Hidden Markov Model (HMM) model are explored.

Figure 4: one-one word alignment between Hindi and English

3.3.1 IBM Model 1

The models make direct use of the idea of alignments, and as a consequence allow us to recover alignments between Hindi and English words in the training data. The resulting alignment models are of central importance in modern Statistical Machine Translation systems. The parameters of the IBM models[12] will be estimated using the expectation-maximization (EM) algorithm. Our focus will be on modeling the distribution $p(f_1...f_m|e_1...e_l, m)$ i.e., the conditional probability of the words $f_1...f_m$, conditioned on the English string $e_1...e_l$, and the Hindi length m. We will have alignment variables $a_1...a_m$ i.e, one alignment variable for each Hindi word in the sentence, where each alignment variable can take any value in 0, 1, . . . , l. Each alignment variable a_j specifies that the Hindi word f_j is aligned to the English word e_{a_j} : We define e_0 to be a special NULL word; so $a_j = 0$ specifies that word f_j is generated from the NULL word. For each i in 1 to m: Choose a_i uniformly from 0, 1, . . . l, choose f_i by translating e_{a_i} given an English sentence $e_1, e_2, \ldots e_l$.

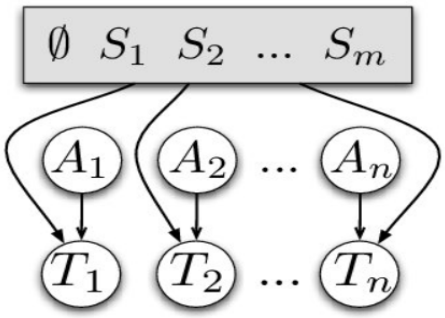

Figure 5: IBM Model 1 word alignment model : The top sentence is the source, and the bottom sentence is the target. Each target word is generated by a source word determined by the relevant alignment variable

Using the EM algorithm, start with uniform probability distribution. For each sentence pair (e, f) – For each Hindi position i ,calculate posterior probability over English positions and increment count of word f_i translating each word. Finally, re-normalize counts to give probabilities

3.3.2 HMM model

The characteristic feature of this approach is to make the alignment probabilities explicitly dependent on the alignment position of the previous word.The motivation is that typically the words are not distributed arbitrarily over the sentence positions, but tend to form clusters. In the Hidden Markov alignment model[11], we assume a first-order dependence for the alignments a_j ; also, the translation probability is assumed to be dependent on the word at position a_j and not on the one at position a_{j-1}. Putting all these together, we have the following basic HMM-based model.

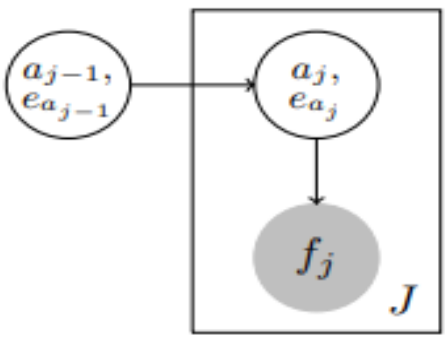

Figure 6: Graphical model of the basic HMM-based word alignment model

4 Experimental Results

4.1 Knowledge Graph

We have implemented our model on DBP15K dataset. It contains four pre-aligned entity language-specific KGs. They are extracted from English(En), Chinese (Zh), French (Fr) and Japanese (Ja) DBpedia. The number of entities varies roughly from 65k to 105k. We trained the model on DBP15k English-Chinese(en-zh) dataset which has 15000 pre-aligned entities between the two KGs. Train test split used was 70 and 30. Entities that are linked across multiple languages are used as alignments. The results obtained with en–zh dataset is shown in table2.

The results of GCN(SE) and GCN(SE+AE) are compared to see whether the attributional information is helpful in the KG alignment task. According to the results, adding attribute embeddings do lead to slightly better results.It shows that the KG alignment mainly relays on the structural information in KGs, but the attributional information is still useful.

SE : zh–en	
Hits@1	39.43%
Hits@10	70.17%
Hits@50	80.19%
Hits@100	83.00%
SE : en–zh	
Hits@1	36.86%
Hits@10	67.34%
Hits@50	78.07%
Hits@100	80.87%
SE+AE : zh–en	
Hits@1	42.85%
Hits@10	74.55%
Hits@50	85.30%
Hits@100	88.33%
SE+AE : zh–en	
Hits@1	39.45%
Hits@10	71.78%
Hits@50	82.89%
Hits@100	85.95%

Table 2: Hits@k measures the proportion of correctly aligned entities ranked in the top k candidates. SE refers to structure embedding and AE refers to attribute embedding.

4.2 Neural Machine Translation

For the translation task, we used a parallel corpus of a hindi and english. Following basic preprocessing, we employed a encoder-decoder model with attention to translate english words to corresponding hindi words, the model was trained for 100 epochs using Adam optimizer and Sparse Cross Entropy loss. Predicted translations and the actual version can be compared visually as well as with evaluation metrics. We used the Bi-Lingual Evaluation Understudy (BLEU) metric to evaluate our results. Some results obtained are shown below:

1)
   a) ACTUAL ['ब्रागा', 'कौन', 'है?']
   b) TRANSLATED ['कौन', 'हो?']
   c) BLEU SCORE
      i) Cumulative 1-gram: 0.303265
      ii) Cumulative 2-gram: 0.428882
      iii) Cumulative 3-gram: 0.482517
      iv) Cumulative 4-gram: 0.510029

2)
   a) ACTUAL ['लेकिन', 'हम', 'इंतजार', 'नहीं', 'कर', 'सकता।']
   b) TRANSLATED ['लेकिन', 'हम', 'नहीं', 'मिल', 'रही.']
   c) BLEU SCORE
      i) Cumulative 1-gram: 0.491238
      ii) Cumulative 2-gram: 0.317093
      iii) Cumulative 3-gram: 0.437775
      iv) Cumulative 4-gram: 0.509523

3)
   a) ACTUAL ['-', 'फेड', 'अप', 'है,', 'तो', 'जल्द', 'ही।']
   b) TRANSLATED ['-', 'जल्दी', 'ही', 'में', 'ले', 'आ|ओ।']
   c) BLEU SCORE
      i) Cumulative 1-gram: 0.141080
      ii) Cumulative 2-gram: 0.345575
      iii) Cumulative 3-gram: 0.468627
      iv) Cumulative 4-gram: 0.540854

Figure 7: Results Obtained using Neural Machine Translation

4.3 Statistical Machine Translation

For the task, EMILLE Corpus containing the parallel corpora of English and Hindi is used. The dataset has around 49000 sentences in both English and Hindi and the train-test split is 20%. The IBM Model 1 ran for 15 epochs and obtained precision value of 0.00578 and recall score of 0.00497. The HMM model ran for 25 epochs. The average BLEU scores obtained over 7 different smoothing techniques can be seen in table 4.3.

Method	BLEU score
Method 1	0.040453
Method 2	0.187903
Method 3	0.080428
Method 4	0.155458
Method 5	0.075804
Method 7	0.201452

Table 3: Average BLEU Score using various smoothing functions for all test samples

IBM Model 1 is weak in terms of conducting reordering or adding and dropping words. In most cases, words that follow each other in one language would have a different order after translation, but IBM Model 1 treats all kinds of reordering as equally possible. Another problem is that in most cases one input word will be translated into one single word, but some words will produce multiple words or even get dropped (produce no words at all). HMM is much easier to implement and modify and is more time-efficient to train.

4.3.1 Bi-Lingual Evaluation Understudy(BLEU)

BLEU, or the Bilingual Evaluation Understudy, is a score for comparing a candidate translation of text to one or more reference translations. This language independent metric compares n-grams of the translated sentence with the n-grams of the reference and counts the number of matches. The more the matches, the better the alignment and better the translation.

5 Conclusion

Although we have worked on a multilingual KG based Entity Alignment approach with high resource languages for ease of understanding, the main problem is finding a suitable low resource knowledge base for Hindi language. We were not able to find any suitable dataset that suits our purpose. We created our own entity alignmened data from english to hindi. But our aligned data could not be added with the GCN model as it accepts inputs from Dbpedia.

6 References

1. Wei Hu, Jianfeng Chen, Yuzhong Qu. A self-training approach for resolving object coreference on the semantic web — Proceedings of the 20th international conference on World wide web (acm.org)link

2. Wei Shen,Jianyong Wang, Ping Luo, Min Wang. Linking named entities in Tweets with knowledge base via user interest modeling — Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mininglink

3. Muhao Chen, Yingtao Tian, Mohan Yang, Carlo Zaniolo. Multilingual Knowledge Graph Embeddings for Cross-lingual Knowledge Alignmentlink

4. Zequn Sun, Wei Hu, Chengkai Li. Cross-lingual Entity Alignment via Joint Attribute-Preserving Embedding link

5. Hao Zhu, Ruobing Xie, Zhiyuan Liu, Maosong Sun. Iterative Entity Alignment via Joint Knowledge Embeddings — IJCAI link

6. Zhichun Wang, Qingsong Lv, Xiaohan Lan, Yu Zhang. Cross-lingual Knowledge Graph Alignment via Graph Convolutional Networks - ACL Anthology link

7. Effective Approaches to Attention-based Neural Machine Translation Minh-Thang Luong Hieu Pham Christopher D. Manning Computer Science Department, Stanford University, Stanford, CA 94305

8. N. Kalchbrenner and P. Blunsom. 2013. Recurrent continuous translation models. In EMNLP

9. S´ebastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In ACL.

10. M.-T. Luong, I. Sutskever, Q. V. Le, O. Vinyals, and W. Zaremba. 2015. Addressing the rare word problem in neural machine translation. In ACL.

11. Vogel, Stephan and Ney, Hermann and Tillmann, Christoph. 1996. HMM-Based Word Alignment in Statistical Translation. In ACL.

12. Aswani, Niraj and Gaizauskas, Robert. 2005. A Hybrid Approach to Align Sentences and Words in English-Hindi Parallel Corpora. Proceedings of the ACL Workshop on Building and Using Parallel Texts.

13. Nitika Nigam ,and Umesh Chandra Jaiswal. 2018. Word Alignment of English-Hindi Parallel Corpus: Relative Study. In IJARSE.

14. Cherry, C., Lin, D. (2003, July). A probability model to improve word alignment. In Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume1 (pp. 88-95). Association for Computational Linguistics.

15. Venkataramani, E., Gupta, D. (2010, December). English-Hindi automatic word alignment with scarce resources. In Asian Language Processing (IALP), 2010 International Conference on (pp. 253-256). IEEE.