Natural Language Processing

Topic : Multilingual Word Embeddings

Team Name - Na Bien Na Mal

Team Members - Akshay Goindani, Zubair abid

Problem Statement

Find the vector representation of words from different languages in a common vector space. These vector representations are called multilingual word embeddings.
For this project we are focusing on bilingual word embeddings. Bilingual word embeddings represent words from two different languages. The languages chosen for this project are Hindi and English. We have chosen Hindi because it is a low resource language and therefore we want to see how the algorithm performs on low resource languages.

Basic Idea

The underlying idea is to independently train the embeddings in different languages using monolingual corpora, and then map them to a shared space through a linear transformation. The algorithm tries to learn the matrices for different languages which are used to map the monolingual word embeddings to the shared space. We use an unsupervised algorithm for learning the matrices, but in order to make improvements in the results we try to add minimal supervision in the algorithm.

Initialization

Since, this is a non convex optimization problem, initial solution is very important in order to avoid the local optima. There are few approaches which try to use some supervision by using a seed dictionary to initialize the solution but their analysis reveals that the self-learning method gets stuck in poor local optima when the initial solution is not good enough, thus failing for smaller training dictionaries.

We use an unsupervised method to build an initial solution without the need of a seed dictionary, based on the observation that, given the similarity matrix of all words in the vocabulary, each word has a different distribution of similarity values. Two equivalent words in different languages should have a similar distribution, and we can use this fact to induce the initial set of word pairings. We combine this initialization with a more robust self-learning method, which is able to start from the weak initial solution and iteratively improve the mapping.

Building an initial solution

X and Z be the word embedding matrices in two languages, so that their ith row Xi∗ and Zi∗ denote the embeddings of the ith word in their respective vocabularies.

We aim to build a dictionary between both languages, encoded as a sparse matrix D where $D_{ij} = 1$ if the jth word in the target language is a translation of the ith word in the source language.

Embedding Normalization:

- Length Normalizing the embeddings.
- Mean centering each dimension.
- Length Normalizing again in order to make all embeddings as unit vector. With unit vectors, dot products are nothing but the cosine similarity.

Fully unsupervised initialization:

- The underlying difficulty of the mapping problem in its unsupervised variant is that the word embedding matrices X and Z are unaligned across both axes: neither the ith vocabulary item Xi∗ and Zi∗ nor the jth dimension of the embeddings X∗j and Z∗j are aligned, so there is no direct correspondence between both languages.
- In order to overcome this challenge we construct 2 alternative representations X’ and Z’ that are aligned along their jth dimension. These matrices then can be used to align their respective vocabularies.
- The idea is to construct similarity matrices MX and MZ from the respective X and Z embedding matrices. The advantage of using similarity matrices is that both the axes are words and hence aligning their jth dimensions becomes easy.

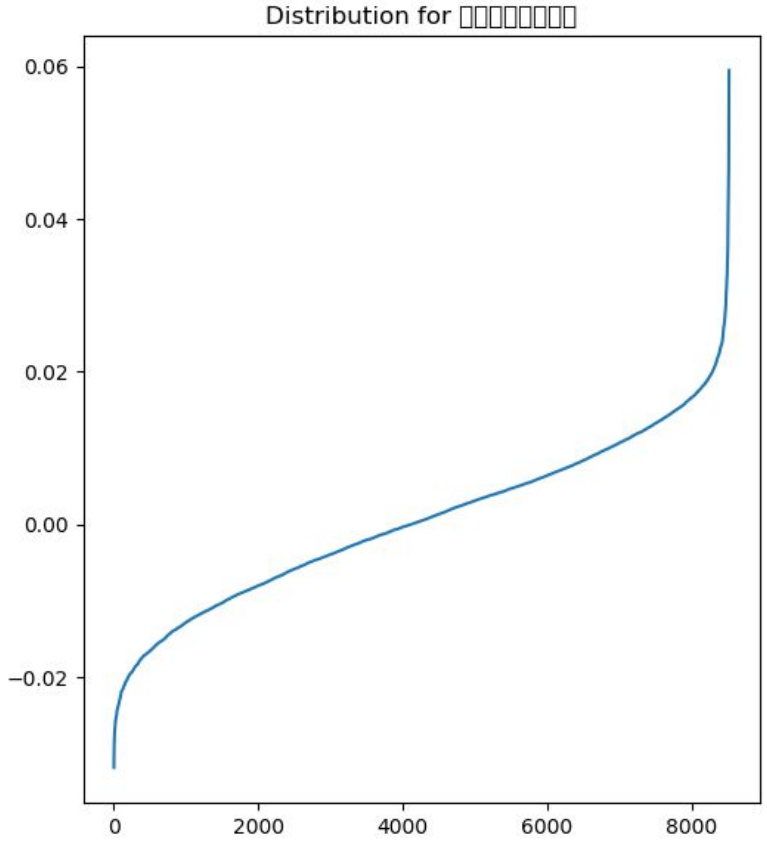
Building an Initial solution

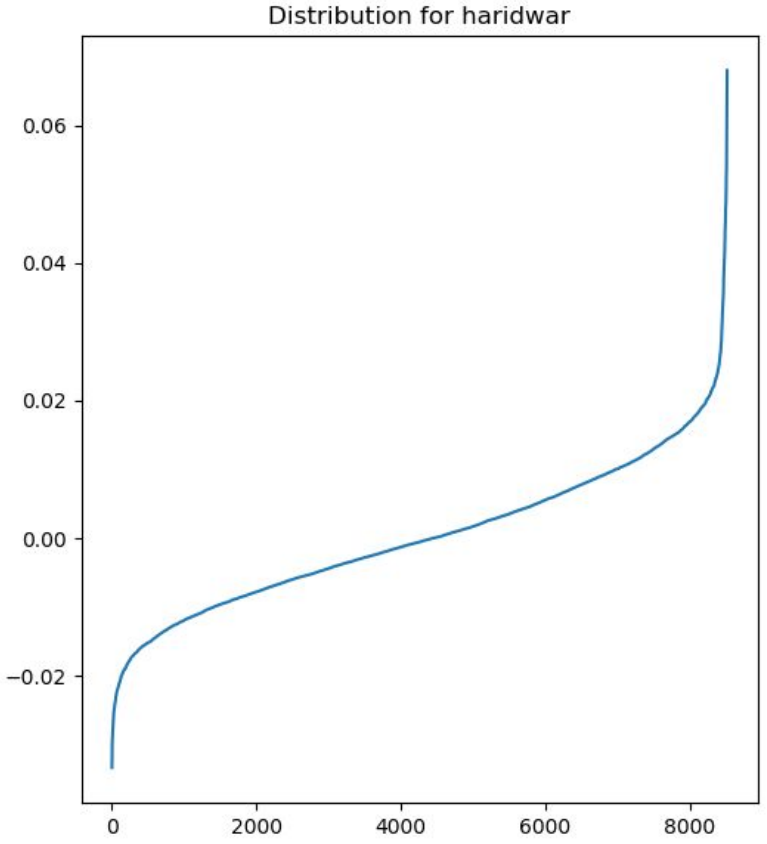
- We assume that the embedding spaces are perfectly isometric therefore the similarity matrices MX and MZ would be equivalent up to a permutation of their rows and columns. Without this assumption, mapping two monolingual embeddings into a common multilingual embedding space without supervision is hopeless.
- Using the above assumption, trying to align the matrices by trying out all the permutations of rows and columns becomes computationally intractable and hence we overcome this problem by sorting the rows of MX and MZ. The idea behind sorting is that, if two words of same meaning have similar similarity distribution then there sorted vectors will be more similar and hence by sorting we are able to align their jth dimension.
- After sorting, for each row in MX we find its nearest neighbor among the rows of MZ. If for the ith row in MX, the nearest neighbor is the jth row in MZ then we make Dij = 1 else Dij = 0. Using nearest neighbor we find the most similar word in the other language and make the initial solution as above.

Robust self learning

This self learning method is a variant of expectation maximization method.

- Expectation Step:
  - In the expectation step we compute the WX and WZ matrices which are used to project X and Z into the shared space. Wx and WZ are computed based on the dictionary D.
- Maximization Step:
  - In the maximization step we update the dictionary D based on the matrices WX and WY.
  - Dictionary D is computed using a simple nearest neighbor algorithm. For each row in X*WX we find its nearest neighbor in Y*WY and make the corresponding entries = 1 in the dictionary D.
- Joining initialization and Robust self learning:
  - It was observed that just using both of them did not work. Some other Improvements such as Cross-domain Similarity Local Scaling (CSLS), symmetric re weighting are required to improve the solution.
- Applying CSLS retrieval and symmetric reweighting:
  - We observed that by making above improvements the results were better than previous but there are some shortcomings which need to be exploited.


Distribution for □□□□□□□□□
0.06
0.04
0.02
0.00
−0.02
0
2000
4000
6000
8000


Distribution for haridwar
0.06
0.04
0.02
0.00
−0.02
0
2000
4000
6000
8000

Proposed Approach

- We observed that the dictionary D did not vary much between the iterations and hence the solution was not getting updated. In order to overcome this problem, we decided to add minimal supervision in the form of a seed dictionary. The seed dictionary contains hindi-english word aligned pairs.
- We build a dictionary in an unsupervised manner (explained above) and then start the self learning process. After every iteration we select a subset from the seed dictionary and for each pair in that subset we make the corresponding entry in the dictionary = 1.
- The hyper-parameter here is the subset size. If the subset size is too small then the dictionary wouldn't vary much and if the subset size is too big then the solution may not converge leading to divergence. By adding the supervision after each iteration, the dictionary gets updated significantly and the solution also gets updated.
- We incorporated this approach in the model with and without CSLS and found that the results are better than the previous models.

Error Analysis

- CSLS Retrieval:
  - The purpose of introducing CSLS was to remove the problem of hubness due to nearest neighbors. But through experiments we observed that introducing CSLS didn't always help. It can be made more robust by considering more number of nearest neighbors.
- Dictionary not updating:
  - Another thing which we observed was that the solution was not updating after every iteration. We found that the root cause of this problem is that the dictionary D is not getting updated after every iteration and hence to improvise we focus on varying D after every iteration.
- Same english word is always the nearest neighbor:
  - Another error is that a single english word is many times most similar to many hindi words. The root cause of this is that we are only focusing on a single direction (hindi - english) due to which the most frequent words on the english side turn out to be most similar to many hindi words.

Paper Presentation

Paper - Unsupervised Multilingual Word Embeddings