# PROJECT SEMINAR

—

Kousshik Raj
17CS30022

# EFFECTIVE UNSUPERVISED CROSS-DOMAIN TRANSFER FOR LOW RESOURCE SCENARIOS

# Word Embeddings

# INTRODUCTION

- The project aims to answer questions through text retrieval and reading comprehension for Indian Languages.
- Success depends on the amount of quality training data available to train the model.
- Hence, we use a cross lingual approach, where we use a language with high resource as a reference
- I focus on developing embeddings for English, Hindi, Bengali so as to improve QA results for these languages which can be extended to the other low resource languages.

# APPROACH

- I am trying to improve Monolingual embeddings for English, Hindi, Bengali, so that the mapped embeddings generated will improve the QA results.
- I have tried different ways.
- Only using the FastText Embeddings
- Only using the embeddings generated from DT dataset.
- Merging FastText and embeddings from DT.

# FastText Embeddings

- FastText is an open-source library that has pretrained models for 157 languages. We used the embeddings of length 300 for English, Hindi and Bengali.
- Using this monolingual embeddings as source, and a seed dictionary, **Vecmap** was run to generate mapped embeddings for English and Hindi.
- While testing the mapped embeddings, with a coverage of **39%**, we had an accuracy of **54.5%**.

# Embeddings from DT Dataset

- We have a knowledge graph (weighted) from the DT Dataset for English, Hindi and Bengali.
- We have to generate word embeddings of a certain length from it.
- We have 2 ways to approach this - Deepwalk and Node2Vec.
- Both use the concept of random walk and use gensim models to generate word embeddings, but Deepwalk doesn't consider the edge weights.

# Embeddings from DT Dataset (contd.)

- Node2Vec takes up lot of resources, and hence we have to trim the graph to the most common occurring words.
- From the knowledge graphs, we generate embeddings of length 128 for the languages using both Deepwalk and Node2Vec.
- Similarly, with theses embeddings as source, Vecmap was run to generated mapped embeddings.
- During the same testing, with a coverage of **39%**, we had an accuracy of **1.5%** for Node2Vec and **1 %** for Deepwalk.

# FastText and DT Embeddings

- Instead of using the individual embeddings, we can try merging the embeddings of both FastText and DT
- We retrieve the common words from both FastText and DT word embeddings, and generate word vector of length 428 by concatenating both.
- To achieve similarity across tests, we reduce the length of the embeddings to 300 by running PCA over it.
- The resulting embedding is used as a source for Vecmap to generate mapped embeddings.
- During testing, with a coverage of **39%**, we had an accuracy of **34.5%.**

# Observations

- We can see that embeddings from DT dataset tends to reduce the accuracy.
- But according to the previous works, this shouldn't happen.
- To test what happens, we checked the **cosine similarity** and **spearman's rank** between the FastText and merged embeddings.
- We found that due to the trimmed graph, there is a lack of edges between similar words which we guess to be the reason.
- Hence, we guess, not running the entire dataset for generating the embeddings to be the reason for the divergence of the results.

# Conclusion

- As of now, the best result we have is to use the FastText embeddings as the source embeddings to achieve better result.
- But theoretically, the merged embeddings from FastText and DT dataset should have a better result. This can be considered if we can process the entire dataset without trimming the knowledge graph.
- These embeddings are further passed on to train the model for QA in Hindi.

# References


- node2vec: Scalable Feature Learning for Networks
- [1403.6652] DeepWalk: Online Learning of Social Representations
- https://github.com/syedsarfarazakhtar/Word-Similarity-Datasets-for-Indian-Languages
- https://arxiv.org/pdf/1805.06297.pdf
- [1911.09483] MUSE: Parallel Multi-Scale Attention for Sequence to Sequence Learning

# THANK YOU