Co-occurrences using Fasttext embeddings for word similarity tasks in Urdu

Usama Khalid  
Department of Computer Science  
AIM Lab, NUCES (FAST)  
Islamabad, Pakistan  
usama.khalid@nu.edu.pk

Aizaz Hussain  
Department of Computer Science  
AIM Lab, NUCES (FAST)  
Islamabad, Pakistan  
aizaz.hussain@nu.edu.pk

Muhammad Umair Arshad  
Department of Computer Science  
AIM Lab, NUCES (FAST)  
Islamabad, Pakistan  
umair.arshad@nu.edu.pk

Waseem Shahzad  
Department of Computer Science  
AIM Lab, NUCES (FAST)  
Islamabad, Pakistan  
waseem.shahzad@nu.edu.pk

Mirza Omer Beg  
Department of Computer Science  
AIM Lab, NUCES (FAST)  
Islamabad, Pakistan  
omer.beg@nu.edu.pk

Abstract—Urdu is a widely spoken language in South Asia. Though immoderate literature exists for the Urdu language still the data isn’t enough to naturally process the language by NLP techniques. Very efficient language models exist for the English language, a high resource language, but Urdu and other under-resourced languages have been neglected for a long time. To create efficient language models for these languages we must have good word embedding models. For Urdu, we can only find word embeddings trained and developed using the skip-gram model. In this paper, we have built a corpus for Urdu by scraping and integrating data from various sources and compiled a vocabulary for Urdu language. We also modify fasttext embeddings and N-Grams models to enable training them on our built corpus. We have used these trained embeddings for a word similarity task and compared the results with existing techniques. The datasets and code is made freely available on GitHub.

Index Terms—Word Embeddings, Ngrams, Fasttext, Urdu, Word2Vec, Skip-Gram, Low Resource

I. INTRODUCTION

Urdu language originated back in 12th with an Indo-Aryan vocabulary base and is a mixture of Arabic and Persian. Urdu language is widely spoken and written in the South Asian region with more than 170 million speakers specifically in Pakistan and India. Despite this Urdu is considered a low resourced language because of insufficient data as compared to English and other widely spoken languages. In recent times the paradigm has been shifted towards the development of efficient models for low resource languages. Many deep learning and machine learning techniques are used to train language models for the derivation of semantics from given textual data. To derive meaningful information from the text it is useful to find out the relation between words. For example, as shown in Fig. 2 words are clustered together based on their similarity. Language models store this information which can then be used for many downstream tasks.

Machines cannot understand the language in the way we do so data cannot be used as it is passed into the network, instead, each word is converted into a N dimensional vector. These representations are known as word embeddings. An example representation of these embeddings are shown in Fig. 1. The words are projected from an n dimension to 3D. The words with related meanings tend to appear close together. The word embeddings are the baseline for any natural language processing task e.g., transliteration, natural language generation, understanding user inputs, etc.
Fig. 2. An overview of the Fasttext architecture.

All these vectors together combined show how much a word is similar to others in a given vector space [10]. For Urdu, a lot of work has been done on semantic analysis and sentence classification however there are no studies that show the performance analysis of word embeddings models on the Urdu language [11]. Unlike the studies conducted for widely spoken languages, in this paper, we use different word embedding models to compute similarity scores for words in the Urdu language.

In this paper, we used a freely available Urdu news text corpus COUNTER [12], [13], which contains data from 1200 documents collected from different news agencies of Pakistan. We have then modified existing Fasttext and n-grams approaches to be applied to Urdu data and we train and provide embeddings. Additionally we compare our trained model and embeddings to previously available skip-gram [14], [15] technique on a word similarity task [15], [16].

This paper is organized in multiple sections which are as follows: In section 2, we will look into the related research. In section 3, we discuss methodologies and the experimental hypothesis. In section 4, we will look at the experimentation results. Finally, in section 5 we will summarize our work and discuss the possible contributions and future directions.

II. LITERATURE REVIEW

A lot of work have been done in Urdu language in terms of POS tagging, Sentimental Analysis, NER, Stemmer, MT, Topic Modeling [1], [6], [11], [17]–[26] but not much work has been done in word embeddings for Urdu. These words embeddings play a major role in natural language understanding. Multiple language embedding training architectures have been introduced i.e. BERT [27], Word2Vec etc.

There are many ways in which words vectors can be represented among them one is one hot encoding vector representation [28]. In one hot encoding the vectors are represented as long binary vector representations of words [24].

To formulate one hot vectors for a corpus, they can be aggregated to form the BoW (Bag of Words) representation [29], [30]. The bag of words maintain a dictionary of all possible words in the language and keep track of the frequency of the word encountered in the particular corpus.

The problem with BoW is that it doesn’t keep track of words similarity and contextual meaning. So to solve the words similarity problem TF-IDF (Term Frequency - Inverse Document Frequency) [31], [32] has been introduced. It associates each word in a document with a number which is a measure of how relevant that word is. Based on this similarity of words, one can compare the similarity of multiple documents together.

Word2Vec is a fusion of two architectures i.e. CBOW and Skip-Gram [33]. These architectures are designed to be mirror images of one another [34]. The CBOW the model tries to predict the closest context to the input word while the Skip-Gram model tries to predict the closest words to the input word.

Word embeddings help us considerably improve NLP techniques for low resourced languages. In context of Urdu, the only words embeddings present in literature are that of Skip-Gram [14]. To create a large sample word embeddings for urdu, 140 million sentences in Urdu were used. To check the accuracy of learned embeddings, the closest neighboured words were analyzed w.r.t different words in the vector space [35], context window sizes and their performance on Urdu transliteration.

The basic idea behind N-Gram language model is to assign probability to each word in a given sequence of words [36], [37]. In word embeddings the words are dissected into multiple N number of chunks and then these chunks are assigned probabilities. Using these probabilities the closest context of a word can be calculated in the vector space. By the calculation of probabilities, this model is very helpful in Natural Language Generations, sentence completion [38], sentence correction etc. The main issue of N-Gram model is that it is sensitive to the training corpus. Many models have been introduced
which combine N-Grams and neural networks to overcome the problems of N-Gram and generate more accurate results.

Fasttext is primarily an architecture developed by Facebook for text classification [39]. Fasttext works on the principal of Word2Vec and n-grams technique. In word2vec the text is feed into the Neural Network individually. However in Fasttext the words are divided into several sub words and then feed into the Neural Networks. Consider the word apple and we have to dissect this word into tri-grams then the resultant output would be app, ppl, and ple [40]. The word vector for apple will be the sum of all these tri-grams. After training the Neural Network on the training data, we get the word vector for each n-grams and later these n-grams can be used to relate other words. For rare words can be mapped as there will be many overlapping n-grams which appeared in other words.

III. METHODOLOGY

We used two methods to train our models on word embeddings, Fasttext and N-Grams [41]. Fig. 3 shows the working of modified N-Gram model used in this research. The n-gram model converts the document into tokens and stores these tokens in a dictionary based on the co-occurrences of words. That is the number of times a token $t_i$ appears next to a token $t_j$ is stored in a co-occurrence dictionary. Against each key there is a are multiple word vectors with the probability score of its occurrence. In fasttext a document is tokenized and passed through a network. The network learns weights which can be extracted as word embeddings. Fig. 2 shows how words are propagated through the network to extract embeddings for Urdu. In next sections we will discuss in detail about the dataset, experimentation and results.

A. Corpus

We have used the Urdu Monolingual corpus [42] containing 54 million sentences, 90 million tokens and 129K unique tokens. In the preprocessing step we removed all special characters such as brackets, single/double quotes and spaces [43]. All these special characters are replaced by spaces. As a second step consecutive occurring spaces of two or more are matched and replaced by a single space character.

B. Techniques

We have used two techniques for training, namely ngrams and Fasttext [45]. The ngram technique requires data to be separated sentence by sentence where each sentence is broken down into a list of words [46]. After separating into list of words we remove common stop words. Similar preprocessing is applied for Fasttext, however the fasttext python package has the tokenizer and stop word removal tool builtin. The complete architecture is given in figure 2 and 3.

C. Hyper Parameters

The Fasttext technique has four main hyper parameters [47] that we can tune. Vector dimension represents the length of the vector size to represent a single word. Larger vectors capture more information [48] but are harder to train and cost more data [49]. Epochs is the number of times the model trains on a batch of data. The larger the corpus the lesser number of times it may have to be iterated. Learning rate is a measure of how quickly the model should converge to a solution [50]. Sub words length specifies the length of substrings [51] to consider for different processing tasks like resolving out of vocabulary words.

For the current study we have used the default parameters for Fasttext which are

- Vector dimension : 100
- Epochs : 5
- Learning rate : 0.05
- Sub words length : min=3 & max=6

The ngrams technique only has a single hyper-parameter namely the number of consecutive words or grams to train.

IV. RESULTS AND DISCUSSION

For the evaluating the similarity of learned word representations We use Urdu translated version of corpora SimLex-999 [52] and WordSim-353 [53]. SimLex-999 is a gold standard dataset for evaluating word embeddings. It contains 999 noun, adjective and verb triplets in a concrete [54] and abstract form. The dataset is designed to evaluate similarity of words rather than relatedness and contains similarity score for words. The WordSim-353 dataset [55] contains relatedness scores for 353 word pairs.
These datasets have been translated to Urdu using ijunoon’s translation service\footnote{https://translate.ijunoon.com/} and made available. For calculating the similarity and relatedness of words we use the Spearman correlation coefficient $r_s$\footnote{Mirza O Beg, Mubashar Nazar Awan, and Syed Shahzaib Ali. Algorithmic machine learning for prediction of stock prices. In FinTech as a Disruptive Technology for Financial Institutions, pages 142–169. IGI Global, 2019.}. The difference between the predicted score and actual score is $d$ and $n$ is the number of examples.

$$r_s = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2 - 1)} \quad (1)$$

|                      | WordSim-353 | SimLex-999 |
|----------------------|-------------|------------|
| Fasttext             | 0.462       | **0.743**  |
| bigrams              | 0.188       | 0.156      |
| skip-gram [14]       | 0.492       | 0.293      |
| Fasttext English [57]| 0.84        | 0.76       |

Fig. 4. In this figure, some word embeddings produced by fasttext are mapped to a 2D space which shows how words are related and how they appear in close proximity to each other.

Fig. 5. We compare our results with skip-gram\cite{14} embeddings that were evaluated on the same dataset. The Fasttext embeddings are trained for 100 dimensional vectors so the results of skip-gram are also for 100 dimensional vectors for a fair comparison.

The bigrams similarity measure as expected produces a low correlation score, this is also because correlation is only computed for exact word matches from the corpus which are comparatively very less as compared to Fasttext for WordSim-353 and SimLex-999. The Fasttext technique outperforms skip-gram based technique \cite{14} for the SimLex-999 task however slightly under-performs in WordSim-353.

V. CONCLUSION

The advent of Word Embedding techniques \cite{58} was no less than a revolution in the field of NLP. It enabled the representation of words in a digital form (vectors) that computers can understand and perform mathematical calculations on, like the famous example King - Man + Woman = Queen. It also established the ground work for modern Deep attention based models and Transformers in the field of NLP.

Urdu has for long remained an Under resourced language which has caused many proposed state-of-the-art techniques to under perform when being applied to Urdu corpora. It can also be seen in Fig. 5 that performance of Fasttext on Urdu corpora is nowhere near to that of English. In this research we have proposed Word co-occurrences using bigrams and Fasttext word embeddings trained using the COUNTER corpus and have evaluated our approach on WordSim-353 and SimLex-999 similarity tasks and compared that to previously proposed skip-gram technique.

In the future work can be done on training these techniques on larger Urdu corpora and evaluate on various tasks like POS Tagging, NER, Machine Translation, sentiment analysis and dependency parsing. In addition to this large corpora have to be proposed for Urdu if we want to at least match the performance of techniques that have been proposed for High resource Languages such as English. We hope that this work will help researchers to produce better techniques in the area of Urdu NLP.

REFERENCES

[1] Mirza Beg and Mike Dahlin. A memory accounting interface for the java programming language.
[2] Bilal Naeem, Aymen Khan, Mirza Omer Beg, and Hasan Mujtaba. A deep learning framework for clickbait detection on social area network using natural language cues. Journal of Computational Social Science, pages 1–13, 2020.
[3] Abdul Rehman Javed, Muhammad Usman Sarwar, Mirza Omer Beg, Muhammad Asim, Thar Baker, and Hissam Tawfiq. A collaborative healthcare framework for shared healthcare plan with ambient intelligence. Human-centric Computing and Information Sciences, 10(1):1–21, 2020.
[4] Mirza Beg and Peter Van Beek. A graph theoretic approach to cache-conscious placement of data for direct mapped caches. In Proceedings of the 2010 international symposium on Memory management, pages 113–120, 2010.
[5] Hafiz Tayyeb Javed, Mirza Omer Beg, Hasan Mujtaba, Hammad Majeed, and Muhammad Asim. Fairness in real-time energy pricing for smart grid using unsupervised learning. The Computer Journal, 62(3):414–429, 2019.
[6] Rabail Zahid, Muhammad Owais Iidrees, Hasan Mujtaba, and Mirza Omer Beg. Roman urdu reviews dataset for aspect based opinion mining. In 2020 35th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW), pages 138–143. IEEE, 2020.
[7] Mirza Beg, Laurent Charlin, and Joel So. Maxsm: A multi-heuristic approach to xml schema matching. 2006.
[8] Aaditeshwar Seth and Mirza Beg. Achieving privacy and security in radio frequency identification. In Proceedings of the 2006 international symposium on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services, pages 1–1, 2006.
[9] Abdul Ali Bangash, Hareem Sahar, and Mirza Omer Beg. A methodology for relating software structure with energy consumption. In 2017 IEEE 17th International Working Conference on Source Code Analysis and Manipulation (SCAM), pages 111–120. IEEE, 2017.
[10] Sumanth Chandrasekaran and Mirza Omer Beg. Fairness in real-time energy pricing for smart grid using unsupervised learning. In Proceedings of the 2010 international symposium on Memory management, pages 113–120, 2010.
channel attack using smartphone keystrokes. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–14, 2020.

[56] Ch Spearman. The proof and measurement of association between two things. *International journal of epidemiology*, 39(5):1137–1150, 2010.

[57] Vitalii Zhelezniak, Aleksandar Savkov, April Shen, and Nils Y Hammerla. Correlation coefficients and semantic textual similarity. *arXiv preprint arXiv:1905.07790*, 2019.

[58] Martin Karsten, Srinivasan Keshav, Sanjiva Prasad, and Mirza Beg. An axiomatic basis for communication. *ACM SIGCOMM Computer Communication Review*, 37(4):217–228, 2007.