



IJRASET

International Journal For Research in
Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 12 **Issue:** X **Month of publication:** October 2024

DOI: <https://doi.org/10.22214/ijraset.2024.64747>

www.ijraset.com

Call: ☎ 08813907089

E-mail ID: ijraset@gmail.com

Improving Word Embedding on Malayalam Corpus

Fathima Murshida K¹, Ruby Fathima A C²

Assistant Professor, Farook College, BVOC Software Development, Kerala, India

Abstract: NLP is natural language processing or neuro linguistic programming. Natural languages like Malayalam are highly inflectional and agglutinative in nature. This is problematic when dealing with NLP based Malayalam applications. So that in order to improve performance of Malayalam NLP based applications, word embedding improvement on Malayalam corpus is needed. The improvement is based on converting the words contained in the Malayalam corpus into a standardised means removing all inflectional parts in the words in the existing Malayalam corpus i.e. taking root words only. All that is needed is a stemmer. In this project I have used a Malayalam morphological analyser for taking root words of all words in the existing Malayalam corpus. The advantage of removing inflectional parts from all words is that we can reduce the sparsity in the existing Malayalam corpus. Also there will be a high hike in frequency of words in the resulting corpus, then the space and time complexity of word embedding representation of the existing corpus will decrease. According to Zipf's law by increasing frequency of words performance of neural word embedding will increase. Zipf's Law is a discrete probability distribution that tells you the probability of encountering a word in a given corpus. By applying Zipf's law I am proposing there will be improvement on Malayalam word embedding. Here using fastText, word embeddings are performed and capture dense word vector representation of the Malayalam corpus with dimensionality reduction from the sparse word co-occurrence matrix. The improvement is mainly used for wordnet, analogy, ontology based Malayalam applications. **Index Terms**—Morphological Analyzer, Zipf's law, Preprocessing, Testing, Training

I. INTRODUCTION

NLP stands for Neuro-Linguistic Programming. Neuro refers to your neurology, Linguistic refers to language, programming refers to how that neural language functions. In other words, learning NLP is like learning the language of your own mind. It is the sub-field of AI that is focused on enabling computers to understand and process human languages. Computers can't yet truly understand Natural languages in the way that humans do — but they can already do a lot. We might be able to save a lot of time by applying NLP techniques to projects.

In this project I am using Malayalam language. Malayalam is an Dravidian Indian language spoken in the Indian state of Kerala. It is one of the 22 scheduled languages of India and was designated a Classical Language in India in 2013. The earliest script used to write Malayalam was the Vatteluttu script, and later the Kolezhuttu, which derived from it. The oldest literary works in Malayalam, distinct from the Tamil tradition, are the Paattus, folk songs, dated from between the 9th and 11th centuries. Grantha script letters were adopted to write Sanskrit loanwords, which resulted in the modern Malayalam script. Malayalam is a highly inflectional and agglutinative language. Algorithmic interpretation of Malayalam's words and their formation rules continues to be an untackled problem. The word order is generally subject-object-verb, although other orders are often employed for reasons such as emphasis. Nouns are inflected for case and number, whilst verbs are conjugated for tense, mood and causativity (and also in archaic language for person, gender, number and polarity). Being the linguistic successor of the macaronic Manipravalam, Malayalam grammar is based on Sanskrit too. Because of its high complexity nature it is challenging to work on Malayalam language.

Morphological analyzer and morphological generator are two essential and basic tools for building any language processing application. Morphological Analysis is the process of providing grammatical information of a word given its suffix. Morphological analyzer is a computer program which takes a word as input and produces its grammatical structure as output. A morphological analyzer will return its root/stem word along with its grammatical information depending upon its word category. For nouns it will provide gender, number, and case information and for verbs, it will be tense, aspects, and modularity. In my project need to remove inflections from each word in the corpus so that the frequency of words in the resulting corpus will increase. So the resulting corpus will contain only the root words. By increasing the frequency of words, complexity of word embedding will decrease.

Because sparsity is more in inflected words. Also with limited resources will get better output provided that not considering or taking any morphosyntactic information. In my project am using only the conceptual similarity, that is conceptually similar words co-occurene so that Malayalam NLP based wordnet, ontology, analogy applications can improve their performance.

In Natural Language Processing we want to make computer programs that understand, generate and, more generally speaking, work with human languages. But there's a challenge that jumps out: we, humans, communicate with words and sentences; meanwhile, computers only understand numbers. For this reason, we have to map those words (sometimes even the sentences) to vectors: just a bunch of numbers. That's called text vectorization. It is also termed as feature extraction. Different ways to convert text into numbers are Sparse Vector Representations and Dense Vector Representations.

II. LITERATURE SURVEY

A. State of the art

- 1) *Word2Vec* : Word2Vec is a statistical method for efficiently learning a standalone word embedding from a text corpus. It was developed by Tomas Mikolov, et al. at Google in 2013 as a response to make the neural-network-based training of the embedding more efficient and since then has become the de facto standard for developing pretrained word embedding. The model is, in contrast to other deep learning models, a shallow model of only one layer without non-linearities. The paper by Mikolov et al. introduced two architectures for unsupervisedly learning word embeddings from a large corpus of text. The first architecture is called CBOW, it tries to predict the center word from the summation of the context vectors within a specific window. The second, and more successful architecture, is called skip-gram. This architecture does the exact opposite, it tries to predict each of the context words directly from the center word. The used pretrained word2vec embeddings are trained using the Skip-gram algorithm. This algorithm is also the inspiration for the algorithms behind the GloVe and fastText embeddings.
- 2) *GloVe* : Global Vector for Word Representation by Pennington, Socher, and Manning (GloVe) was inspired by the skip-gram algorithm and tries to approach the problem from a different direction. Pennington, Socher, and Manning show that the ratio of co-occurrence probabilities of two specific words contains semantic information. The idea is similar to TF-IDF but for weighing the importance of a context word during the training of word embeddings. Classical vector space model representations of words were developed using matrix factorization techniques such as Latent Semantic Analysis (LSA) that do a good job of using global text statistics but are not as good as the learned methods like word2vec at capturing meaning and demonstrating it on tasks like calculating analogies. Their algorithm works by gathering all co-occurrence statistics in a large sparse matrix X , where each element represents the times word i co-occurs with j within a window similar to skip-gram. After which the word embeddings are defined in terms of this co-occurrence matrix.
- 3) *fastText* : fastText as a library for efficient learning of word representations and sentence classification. It is written in C++ and supports multiprocessing during training. FastText allows you to train supervised and unsupervised representations of words and sentences. These representations (embeddings) can be used for numerous applications from data compression, as features into additional models, for candidate selection, or as initializers for transfer learning. Bojanowski et al. introduced the fastText embeddings by extending the skip-gram algorithm to not consider words as atomic but as bags of character n -grams. Their idea was inspired by the work from Schütze in 1993, who learned representations of character four-grams through singular value decomposition (SVD). One of the main advantages of this approach is that word meaning can now be transferred between words, and thus embeddings of new words can be extrapolated from embeddings of the n -grams already learned. The length of n -grams you use can be controlled by the `-minn` and `-maxn` flags for minimum and maximum number of characters to use respectively. These control the range of values to get n -grams for. The model is considered to be a bag of words model because aside of the sliding window of n -gram selection, there is no internal structure of a word that is taken into account for featurization, i.e. as long as the characters fall under the window, the order of the character n -grams does not matter.
- 4) *Paragram*: Wieting et al. introduced a method to tune existing word embeddings using paraphrasing data. The focus of their paper is not on creating entirely new word embeddings from a large corpus. Instead, the authors are taking existing pretrained GloVe embeddings and tune them so words in similar sentences are able to compose in the same manner. Their training data consists of a set of P phrase pairs (p_1, p_2) , where p_1 and p_2 are assumed to be the paraphrases. The objective function they use focuses to increase cosine similarity, i.e. the similarity of the angles between the composed semantic representations of two paraphrases. Important to mention is that Wieting et al. expresses similarity in terms of angle and not in terms of actual distance. Additionally, Wieting et al. only explored one algebraic composition function, namely: averaging of the word vectors.

The data for tuning the embeddings used by the authors is the PPDB 5 or the Paraphrase Database by Ganitkevitch, Van Durme, and Callison-Burch. Specifically, they used version XXL which contains 86 million paraphrase pairs.

An example of a short paraphrase is: “thrown into jail” which is semantically similar to “taken into custody”. Wieting et al. published their pretrained embeddings called Paragram-Phrase XXL 6, which are in fact tuned GloVe embeddings, alongside their paper. These embeddings also have a dimensionality of 300 and have a limited vocabulary of 50,000. In order to apply the embeddings, according to Wieting et al., they should be combined with Paragram-SL999 which are tuned on the SimLex dataset.

III. PROPOSED METHOD

The increasing accuracy of word embedding representation of Malayalam language has a great impact on ontology, analogy representation based NLP based applications. Improved Word Vectors on Malayalam language, which increases the accuracy of pretrained Malayalam word embeddings in NLP Applications. The main aim is to improve word embeddings that they do not need any labeled data. The improvement is by converting Malayalam language into a standardised format by removing inflections from each word in Malayalam corpus.

Any large volume of text can be used to get the word embeddings by feeding it to the model, without any kind of labeling. In this project, we are using fastText word embedding. Word embeddings are derived by training a model on large text corpus.

A. System Architecture

The system architecture mainly consists of five modules. They are Corpus Creation, Data Preprocessing, Training Using Neural Networks, Model Evaluation, Dataset Visualisation.

- 1) *Corpus Creation* : Data is collected from Malayalam news articles and created a Malayalam Corpus containing thousands of sentences.
- 2) *Data Preprocessing*: Data Preprocessing is the important part of this project, because data in the real world is incomplete, noisy, inconsistent. Also Malayalam language is highly inflectional and agglutinative. In this, a Malayalam morphological analyser is used to generate root words or removing inflectional parts from the existing Malayalam corpus. By using sandhi rules Malayalam morphological analyser will generate root words of each word in the existing corpus. So in the resulting corpus frequency of Malayalam words will increase. According to Zipf's law in a large corpus of natural language like Malayalam, the frequency of any word is inversely proportional to its rank in frequency table. Frequency is number of times a word appears in a given corpus. By removing inflections frequency of words will increase, according to Zipf's law by increasing frequency of words performance of neural word embedding will increase.
- 3) *Training Using Neural Networks* : In this project we are using fastText word embedding. fastText is another word embedding method that is an extension of the word2vec model. Instead of learning vectors for words directly, fastText represents each word as an n-gram of characters. This helps capture the meaning of shorter words and allows the embeddings to understand suffixes and prefixes. Once the word has been represented using character n-grams, a skip-gram model is trained to learn the embeddings. FastText can be used both for classification and word-embedding creation. Here it will generate the improved word embedding of the new Malayalam corpus which is containing root words only.
- 4) *Model Evaluation*: Word embeddings should capture the relationship between words in natural language. In the Word Similarity and Relatedness Task, word embeddings are evaluated by comparing word similarity scores computed from a pair of words with human labels for the similarity or relatedness of the pair. Cosine similarity measures the similarity between two vectors of an inner product space. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction. It is often used to measure document similarity in text analysis. In this project cosine similarity measure is used to calculate the similar words.
- 5) *Model Visualisation*: PCA and T-sne are two techniques for visualising dataset in 2D or 3D space. PCA performs a linear mapping of the data to a lower-dimensional space. T-Distributed Stochastic Neighbor Embedding (t-SNE) is a non-linear technique for dimensionality reduction. Well suited for the visualization of high-dimensional datasets. So that t-SNE visualisation technique is used for this project. By using t-SNE visualisation it can be seen that similar words tend to be close to each other and dissimilar words tend to be far from each other.

IV. RESULTS

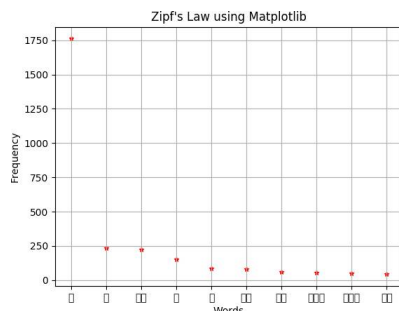


Fig. 1. Graphical result of existing dataset after applying zipfs law

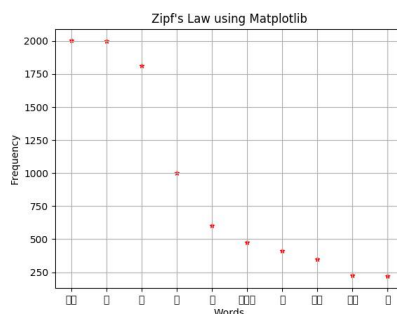


Fig. 2. Graphical result of new malayalam dataset after applying zipfs law

V. CONCLUSION AND FUTURE WORK

It can be visualise that similar malayalam words or syn- onyms occupy close to each other by using tensorflow em- bedding projector. The Euclidian distance and cosine simi- larity of similar words will be lesser than the fastText pre trained word vectors of malayalam language. It can be predict that by using zipfs law malayalam word embedding can be improved. Also sparsity problem in the existing malayalam corpus is decreased by using less resources that is conceptual similarity of words.Space and time complexity is reduced in the new malayalam corpus. The improved word vectors canbe used to all NLP malayalam applications to improve their efficiency,accuracy,speed. In future for machine translationimprovement on malayalam this model can be used as a base model and also learn morphosyntatic information on that model.

REFERENCES

- [1] Abdulaziz M. Alayba, Vasile Palade, Matthew Englandand Rahat Iqbal " Improving Sentiment Analysis inArabic Using Word Representation", IEEE 2nd International Workshop on Arabic and Derived Script Analysisand Recognition (ASAR),2018 .
- [2] Shaosheng Cao Wei Lu, "Improving Word Embeddings with Convolu- tional Feature Learning and Subword Information", Proceedings of the2017 Conference on Empirical Methods in Natural Language Processing
- [3] YuvalPinter,RobertGuthrie,Jacob Eisenstein, " Mimicking Word Embed- dings using Subword RNNs", Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. .
- [4] Vishwani Gupta,Sven Giesselbach,Stefan R"uping,Christian Bauckhage,"Improving Word Embeddings Using Kernel PCA", Proceedings of the 4th Workshop on Representation Learning for NLP.
- [5] Procheta Sen,Debasis Ganguly , "Word-Node2Vec: Improving Word Em- bedding with Document-Level Non-LocalWord Co-occurrences", Pro- ceedings of the 2019 Conference of the North American Chapterof the Association for Computational Linguistics: Human LanguageTechnologies..
- [6] Miguel Ballesteros, Chris Dyer, and Noah A. Smith, "Improved transition-based parsing by modeling characters instead of words with LSTMs", In Proc. EMNLP.2015.
- [7] Marco Baroni and Alessandro Lenci, "Distributional memory: A generalframework for corpus-based semantics", Computational Linguistics, 36(4):673–721, 2010.
- [8] Jan A. Botha and Phil Blunsom, " Compositional morphology for word representations and language modelling", In Proc. ICML, 2014.
- [9] Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huanbo Luan, "Joint learning of character and word embeddings", In Proc. IJCAI,2015.
- [10] Cicero Nogueira dos Santos and Bianca Zadrozny, "Learning character- level representations for part-ofspeech tagging", In Proc. ICML,2014.
- [11] F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida, "Detecting spammer on twitter", in Proc. 7th Annu. Collaboration, Electron. Mes- saging, Anti- Abuse Spam Conf., Jul. 2012, p. 12
- [12] J. Song, S. Lee, and J. Kim, "Spam filtering in Twitter using sender receiver relationship", in Proc. 14th Int. Conf. Recent Adv. Intrusion Detection, 2011, pp. 301317.



- [13] Chao Chen, Yu Wang, Jun Zhang, Yang Xiang, Wanlei Zhou, "Statistical Features-Based Real-Time Detection of Drifted Twitter Spam, Twitter Spam," IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 12, NO. 4, APRIL 2017.
- [14] Sayali Kamble, S.M. Sangve, "Real Time Detection of Drifted Twitter Spam Based on Statistical Features", Features 2018 International Conference on Information, Communication, Engineering and Technology (ICICET) .
- [15] Surendra Sedhai, Aixin Sun, "Semi Supervised Spam Detection Twitter Stream", "IEEE Transactions On Computational Social Systems 2017.
- [16] Hajime Watabe, Mondher Boduazizi, And Tomoaki Ohtsuki, "Hate Speech on Twitter: A Pragmatic Approach to Collect Hateful and Offensive Expressions and Perform Hate Speech Detection", "IEEE Transactions On Computational Social Systems 2018.



10.22214/IJRASET



45.98



IMPACT FACTOR:
7.129



IMPACT FACTOR:
7.429



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089  (24*7 Support on Whatsapp)