A COMPARATIVE STUDY OF WORD EMBEDDINGS FOR INDIAN LANGUAGES

May 6, 2021

Aaradhya Gupta
Aravapalli Akhilesh
Srijith Padakanti

Abstract

Dense word vectors or 'word embeddings' which encode semantic properties of words, have now become integral to NLP tasks like Machine Translation (MT), Question Answering (QA), Word Sense Disambiguation (WSD), and Information Retrieval (IR). In this paper, we use various existing approaches to create and compare multiple word embeddings for 2 Indian languages(Hindi and Telugu).

1 Introduction

This paper aims to study different word embeddings available for Indian Languages and compare them across semantic tasks for accuracy. The semantic task chosen is Analogy. Let's take an example to understand what happens, Roti : North :: Idli: ?. Human brain is capable of understanding patterns of natural language on its own but not a machine. We will discuss how a machine understands the patterns of language in this paper.

The word embeddings we chose are FastText and Word2Vec.The languages used are Telugu and Hindi. Telugu is a Dravidian free-word order agglutinative language. Hindi is an inflection-rich language.Hindi is a head-final (Subject-Object-Verb) language, with relatively free word order. The semantic task of analogy is defined as the task of finding the semantically nearer word based on definitions employed to form the comparison of the analogy. The comparison is done for accuracy where we will evaluate the results based on the gold data obtained for both the languages. To make words understood by machine learning algorithms, word embedding is used to map words into vectors of real numbers. fastText and Word2Vec are some of them.

Word2Vec is a technique used in NLP. The Word2Vec algorithm uses a neural network model to train/learn word associations from a large corpus of text. Word2vec represents each distinct word with a particular list of numbers called a vector. The vectors are chosen carefully such that a simple mathematical function (the cosine similarity between the vectors) indicates the level of semantic similarity between the words represented by those vectors. Word2Vec takes a large corpus as input and outputs a vector space, hundreds of dimensions in most cases, with each unique word in the corpus being assigned to a vector in space. The vectors are then used in various semantic tasks across Natural Language processing.

fastText is 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. The models used were trained using CBOW with position-weights, in dimension 300, with character n-grams of length 5, a window of size 5 and 10 negatives.

2 Related Work

The main contributions of our work are to the problem of the word embeddings in Indian langauge. we used fastText and Word2vec for testing the results of the existing word-embeddings.The impact of our approach that addresses these existing limitations is right now we weren't capable of have a nice corpus for Telugu or Hindi. Our data was also limited.
There have been many comparative studies before on various embeddings(Baroni et al., 2014.)[1] and some studies have even taken place comparing different word embeddings for Indian Languages(Kumar et al.,2020)[3]

GloVe

GloVe[4] is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. GloVe is another publicly available word-embedding which is available in many Indian Languages.

3 Data

The English data for the semantic task of analogy is obtained from here [5]. The English data was then manually translated into Hindi and Telugu. The translated data is processed and cleaned for it to be used as an input for the algorithm. The data can be available here.

4 Method

4.1 Data Storage

The gold data was translated into Indian languages and stored in text files. For easier usage we then transformed them into lists, which were stored as pickled files.These files are available here.

4.2 Method

We have used word-embeddings available for fastText and Word2vec for evaluating the semantic task of analogy. The algorithm takes pre-trained word embeddings, runs the embeddings against the data given as input. The algorithm used vector arithmetic for finding the nearest possible analogy based on the semantic pattern observed. This can be understood mathematically as cosine functions calculate semantic similarity of words using distance between vectors as a measure. The types of analogies chosen are antonyms,kinship relations,verbs inflected by gender and singular-plural pairs.

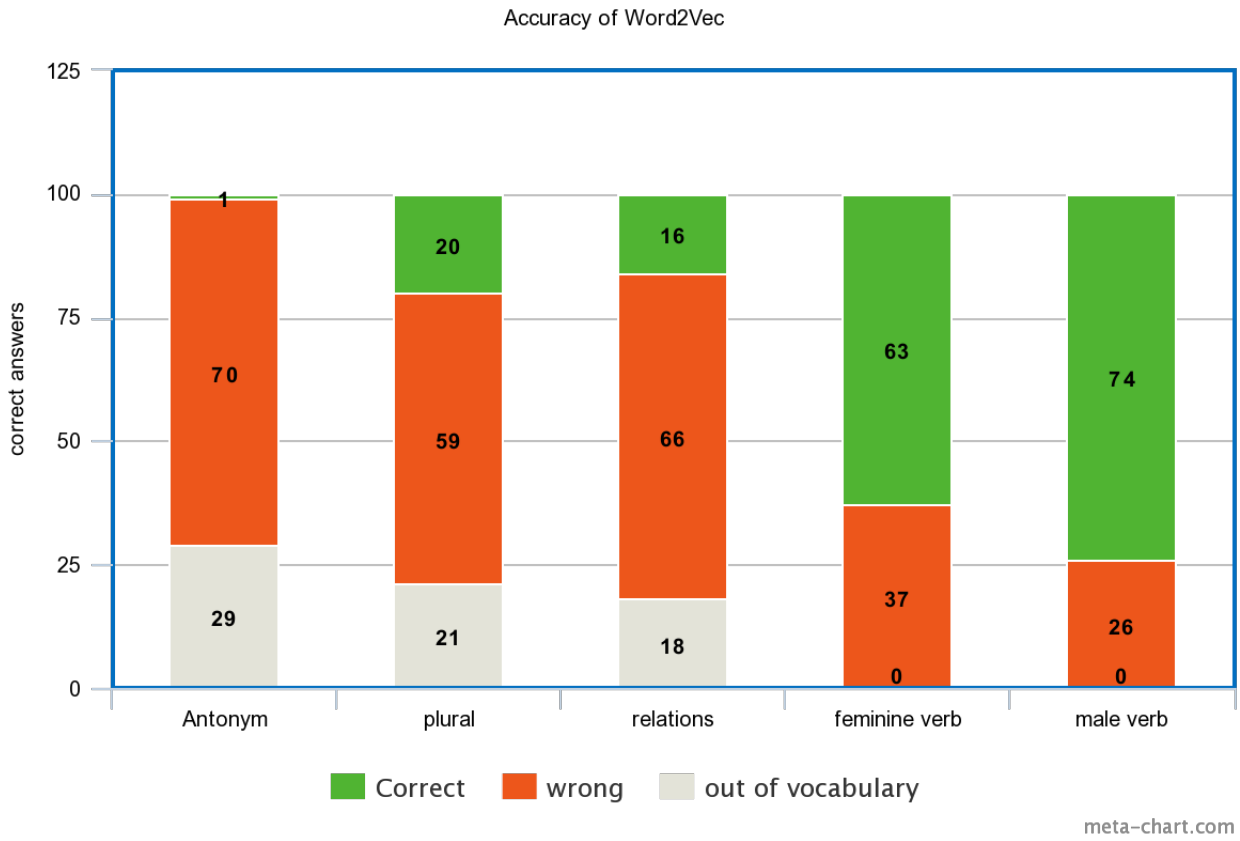
4.3 Models Used

For the fastText Models, we used the wordvectors distribute here[2]
The Word2Vec models were sourced from the LTRC server and the paper cited here [3]

5 Observations

5.1 Hindi

5.1.1 Word2Vec

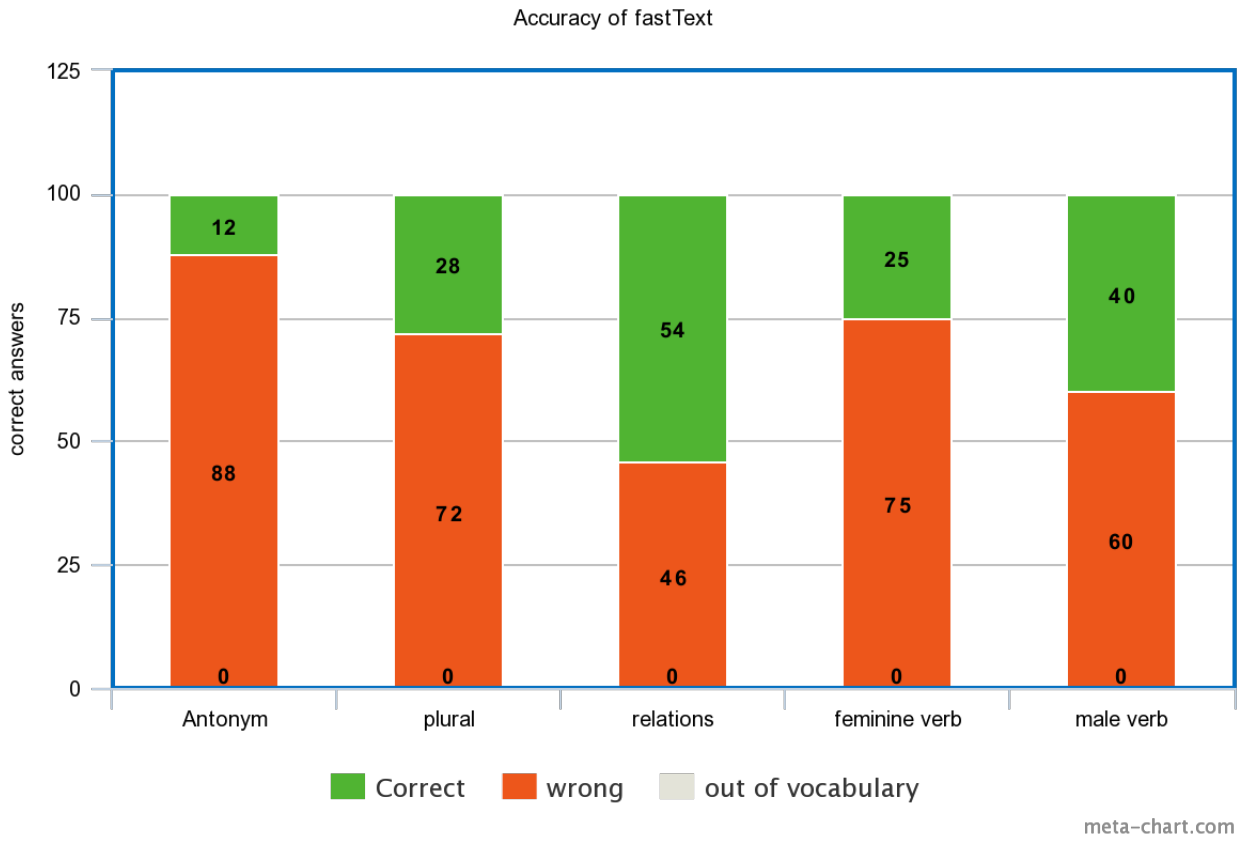


The above graph shows the performance of word2vec model on various analogy tasks for Hindi
As can be seen from the graph, for Word2Vec, there were some words that were out of vocabulary. This is because the corpus used for training had a limited size.
Word2Vec performed abysmally on antonyms and not much better on singular to plural analogy and the relationship analogies.
It beat fastText in the analogies that pertained to capture of gender information. Achieving 74% accuracy in the masculine inflected verb analogy
Word2vec captured gender information storage very well but could not handle the more semantically involved analogies like relations, antonyms, etc. This might be because of

the relatively small training corpus.

Highest accuracy achieved was 74%
Lowest accuracy recorded was 1%
The average accuracy of word2vec model comes out to be 31.8%

5.1.2 fastText



The above graph shows the performance of the fastText model across the various analogy tasks for Hindi
As we can see, there were no out-of-vocabulary words for fastText as it handles them using character-level n-grams
It performed the best on the relations analogy, while the worst in the antonym tasks.

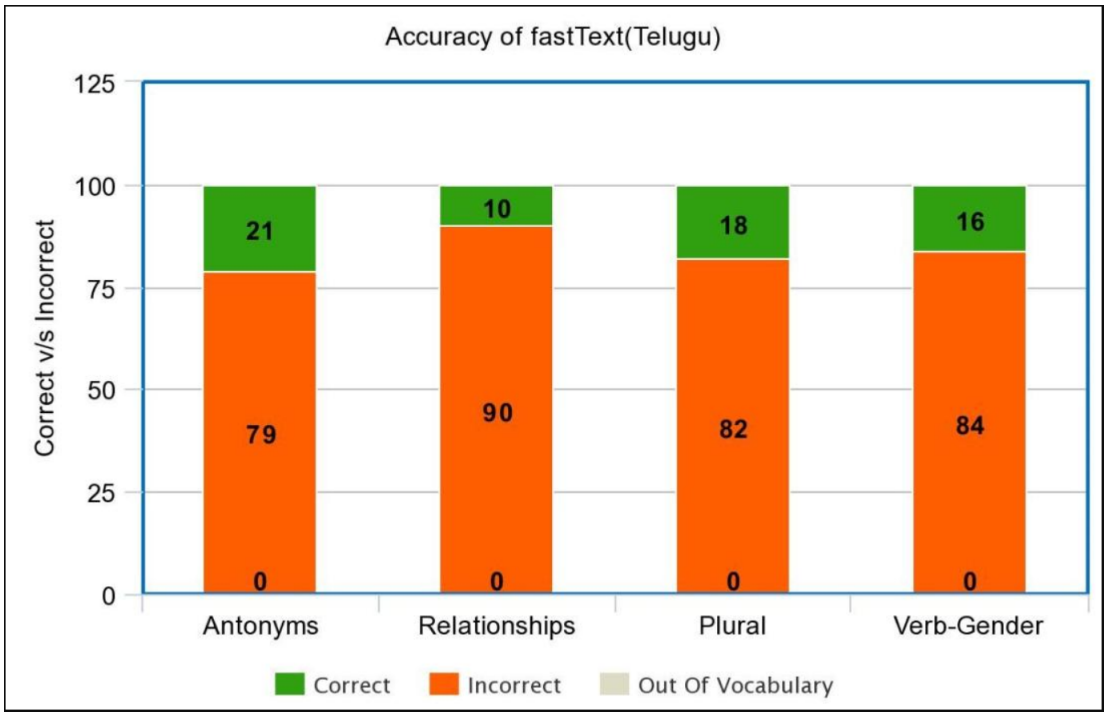
Antonym task is very difficult to handle for word embeddings like these because the context surrounding the 2 words is often similar, hence the antonyms are assigned similar vectors.

Highest accuracy achieved was 54%

Lowest accuracy recorded was 12%
The average accuracy of Word2Vec model comes out to be 34.8%

5.2 Telugu

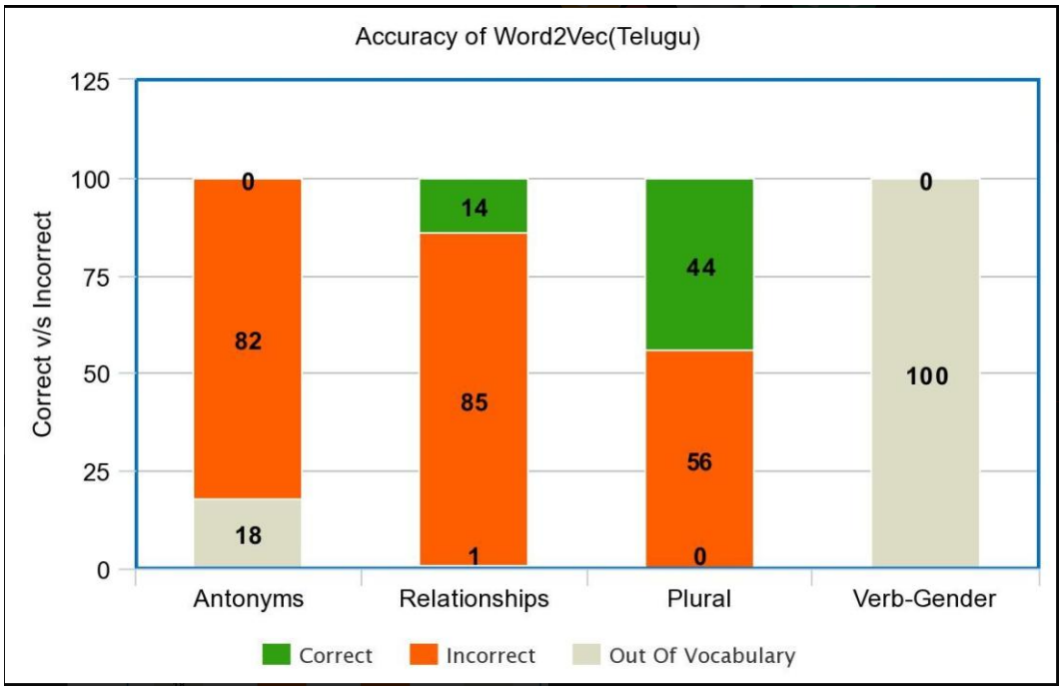
5.2.1 fastText



This graph shows the performance of the fastText model across the various analogy tasks for Telugu.
There were no out of vocabulary words as fastText handles it through character level n-grams.
The intricacies in analysing the Telugu corpus nor the vastness of the corpus for Telugu are good enough so the result is bound to be less. The fastText model has almost equal accuracy for all the types of analogies, this might be a result of richer and denser corpus on which the model was trained on leaving small vulnerabilities to inflected words in the case of agglutinative languages like Telugu.

Highest accuracy achieved was 21%
Lowest accuracy recorded was 10%
The average accuracy for fastText model came out to be 16.25%

5.2.2 Word2Vec



The above graph shows the performance of the word2vec model across the various analogy tasks for Telugu.
As you can see some of the words are out of the vocabulary. The ones in the ash colour.This is because of the size of the corpus. Agglutination of words in Telugu creates the need of richer corpus in terms of verbs as there can be many possibilities of words(verbs) which can be formed based on inflection i.e., gender,number,person,tense, etc. The model performs at its best on antonyms. In the case of Verbs based on gender, most of the words are absent in the vocabulary showing it has not been trained on a corpus which has richer verb content as discussed. This all the way brought down the accuracy of the model. Therefore, considering all types of words and vocabulary used on a corpus becomes important for obtaining higher accuracy.
The corpus on which the model has been trained on has many errors regarding spellings which can be seen in the words which are tagged as correct by the model

Highest accuracy achieved was 44%
Lowest accuracy recorded was 0%
The average accuracy for Word2Vec model came out to be 14.5%

	Telugu	Hindi	Average
Word2Vec	14.5%	31.8%	23.15%
fastText	16.25%	34.8%	25.53%

6 Results

After comparing both the above mentioned publicly available word-embeddings on both Hindi and Telugu, this experiment on semantic task of analogy concludes fastText has a higher accuracy compared to Word2Vec.

7 Challenges Faced

Having chosen languages where much more work needs to be done in the field of Computational Linguistics, obtaining different types of readily available cleaned corpus becomes an issue. Challenges we faced during our project are:

- Hindi and Telugu, despite being widely spoken languages, are low-resource languages. There is a lack of readily usable and clean corpus.

- In the case of Telugu, we faced an issue of limiting or vocabulary corpus to inflected data. This is due to the fact that it is an agglutinative language and therefore finding a word in Telugu all by itself becomes a difficult task, lack of uninflected words pushed us to collect data on semantic patterns such as gender, number and tense which are some of the features which actually cause inflection and agglutination.

- The training and running of these models was a very computationally intensive task. We faced problems running the codes, as they took a lot of time to give back results. Connectivity issues were also faced by many of our project members.

8 Future Scope

For future work, there is a lot of scope and opportunities. It can include expanding the project to include more languages. Out of the 22 scheduled languages of India, many may have decent size corpus available online.
We can also expand the scope of the project by including more word embeddings like GloVe, MUSE, ELMo and many more. We can also fine tune the parameters of the word embeddings currently used by us.
A wider variety of semantic tasks like concept categorization, synonym detection, etc. to analyze the different embeddings from different settings

References

[1] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247, 2014.

[2] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[3] Saurav Kumar, Saunack Kumar, Diptesh Kanojia, and Pushpak Bhattacharyya. "a passage to india": Pre-trained word embeddings for indian languages. In *Proceedings of the 1st Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL)*, pages 352–357, 2020.

[4] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[5] Zhe Zhao, Tao Liu, Shen Li, Bofang Li, and Xiaoyong Du. Ngram2vec: Learning improved word representations from ngram co-occurrence statistics. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, 2017.