Adaptive GloVe and FastText Model for Hindi Word Embeddings

Vijay Gaikwad
Computer Engineering
College of Engineering Pune
Pune, Maharashtra, India
gaikwadva17.comp@coep.ac.in

Yashodhara Haribhakta
Computer Engineering
College of Engineering Pune
Pune, Maharashtra, India
ybl.comp@coep.ac.in

ABSTRACT

Today, a lot of research is carried out on word embeddings in NLP domain. The algorithms like GloVe, FastText are used to develop word embeddings. However, not enough work is done on Indian languages due to lack of resource availability. The datasets required for testing word embeddings are not available for Indian languages. In this paper, two algorithms are proposed - Adaptive GloVe model (AGM) and Adaptive FastText model (AFM). Adapting to the co-occurrence matrix generation process of the original GloVe model, AGM, leverages part of speech tags, morphological knowledge of the language. Assigning higher co-occurrence weight to words with same root, AGM, significantly improved accuracy of resultant word embeddings on syntactic datasets. Whereas, AFM improves the vocabulary building process of the original FastText model. The work involves generation of word embeddings for low resource language like Hindi using AGM and AFM and creation of necessary test datasets for evaluating word embeddings. AGM word embeddings showed morphological awareness, achieving 9% increase in accuracy on syntactic word analogy task, compared to original GloVe model. AFM outperformed FastText by 1% accuracy in word analogy task and 2 Spearman rank on word similarity task, providing state-of-the-art performance.

KEYWORDS

Word Embeddings, GloVe Model, FastText Model, Adaptive GloVe Model, Adaptive FastText Model.

ACM Reference format:

Vijay Gaikwad and Yashodhara Haribhakta. 2020. Adaptive GloVe and FastText Model for Hindi Word Embeddings. In *Proceedings of CoDS COMAD 2020, January 5–7, 2020, Hyderabad, India 5 pages.* https://doi.org/10.1145/3371158.3371179

1 Introduction

Word embeddings are representation of words in vector form. Here, words are mapped to vectors of real numbers. Different methods have been proposed to generate word embeddings. Word embeddings are useful in many downstream NLP tasks like neural machine translation [18], sentiment analysis, language modeling, named entity recognition etc. Researchers have used it in improving accuracy of word sense disambiguation [5], detection of multi-word expression [22]. However, most of the work in NLP has been done for resource rich languages like English [2]. Indian languages like Hindi, Marathi lacks huge corpus. Many datasets like named entity dataset, word analogy dataset, which are useful in testing word embeddings are not available for Indian languages. Due to this reason, most of the NLP tasks have not been done for Indian languages [1] and some tasks like neural machine translation are less accurate because of the unavailability of resources for Indian languages [10]. Word embeddings are developed using standard techniques like **GloVe, Word2Vec** and **FastText**. These techniques have not been widely tested for Indian languages. Word embeddings formed using GloVe model give contextual information of words, ignoring morphology of words [6]. FastText model works on character level and has additional ability to deal with rare words. This paper works on the limitations of GloVe and FastText model, adapting original GloVe model to incorporate morphological knowledge of words and adapting original FastText model for performance improvement. This paper also talks about developing word embeddings for Hindi language and testing these word embeddings using word analogy and similarity datasets. The rest of the paper is as follows. In section 2, state-of-the-art-work on word embeddings is discussed. Proposed models are described in section 3 and section 4. Section 5 discusses about Training and Evaluation datasets. It is followed by results in section 6. Section 7 concludes this paper followed by future scope in section 8.

2 Related Work

Word embeddings are developed using two types of models, viz, Matrix Factorization Method, Shallow Window Based Methods.

2.1 Matrix Factorization Method:

Hyperspace Analogue to Language (HAL) is a technique that is based on matrix factorization method [9]. In this technique, word co-occurrence matrix is developed. Rows of this matrix represent words and columns represent context words and entries correspond to number of times the word appears with the context word. Word embeddings were tested on nearest neighbor test. As a result, it was observed that word vectors carried semantic information along with categorical information. For example, body parts, geographical locations, types of animals were represented as separate groups. However, most frequent words affect the performance of HAL severely.

[21] introduced new method called COALS. This method is an improved version of HAL. This paper introduced some normalization techniques to stop frequent words affecting co-occurrence counts. On nearest neighbor test, COALS produced reasonable results as compared to HAL method.

2.2 Shallow Window Based Method:

In shallow window-based method, word representations are learned using small local context window. Word2Vec model proposed by [15] is the best example of this method. In this work, two novel model architectures i.e., **Skip-Gram** and **Continuous Bag of Words (CBOW)** have been proposed. In skip-gram, the objective is to predict the word's context given the word itself, whereas in CBOW, word is predicted given the context. However, these methods do not efficiently use the co-occurrence statistics of corpus efficiently [17].

Further, an extension to Skip-Gram and CBOW was suggested by Mikolov and Sutskever [16]. This variant improved training speed and used negative sampling instead of hierarchical softmax to avoid noise due to overly frequent words. Al-Rfou et. al. generated word embeddings for 100 languages using their Wikipedia articles and showed improvement in performance in POS tagging using these generated embeddings [2].

[17] proposed GloVe model that combined both of these methods to compute word embeddings. GloVe model uses the Co-occurrence matrix like matrix factorization methods, and it uses local context window to count co-occurrences of words like shallow window-based methods. This model avoids shortcoming of both methods and performs better in all test datasets.

It is observed that GloVe model ignores how words are formed and do not take into consideration Morphological information of words. This knowledge was first taken into consideration by [6] with the advent of FastText model. Here, each word is represented as a bag of character n-grams and their word vectors are represented as the sum of these n-gram representations.

Further, [8] developed word embeddings for 157 languages using FastText model. This paper showed that FastText model achieved 32.1% accuracy on Hindi word analogy, which is very less than average accuracy (66.7) achieved for top ten languages. The reason for this is considerably less size of Hindi corpus as compared to other languages used for creating word vectors.

Researchers are trying to make use of language knowledge in generating better word embeddings. One such important attempt is done by [13]. In this paper, existing Skip-Gram model is adapted by making use of dependency based syntactic context derived from dependency-based parse tree instead of using linear bag of words context.

Currently, a lot of research is carried out for different Indian languages [19], [4], [3]. [20] applied existing NMT techniques on six Indian language pairs.

[11] have developed English-Hindi parallel corpus, Hindi monolingual corpus. This corpus has not yet been used for developing word embeddings. Along with that, [1] have developed word similarity datasets for Indian Languages. This dataset can be used for testing word embeddings.

3 Adaptive GloVe Model (AGM)

The original GloVe model works on word level. It creates co-occurrence matrix of words. Let C denote this matrix. In this matrix, rows and columns denote word vocabulary and the matrix entry, say C_{ij}, denote co-occurrence weight of word i and word j. Co-occurrence weight C_{ij} increases with the number of times the word i appears with word j. The higher the C_{ij}, the more similar the resultant vectors are. Let MC denote the maximum co-occurrence value in this matrix.

In FastText, words are broken into n-grams. So, as a result, vectors of words that share same root (e.g. play, played, playing) are most similar.

As an experiment, nearest neighbor test is conducted on FastText embeddings for English language available on [25] and results obtained are given in table 1.

Given Word	Most similar words as per FastText embeddings
select	('choose', 0.699), ('selected', 0.692), ('selecting', 0.688), ('Select', 0.662)
reach	('reaching', 0.771), ('reached', 0.694), ('reaches', 0.670), ('reache', 0.618)
refuse	('refusing', 0.734), ('REFUSE', 0.722), ('refuses', 0.711), ('refused', 0.708)

Table 1: Results of nearest neighbor test on FastText embeddings

It is observed that such words do not appear in same sentence. Obviously, these words do not have high co-occurrence count in GloVe model resulting in very different vectors of these words. This reduces the accuracy of GloVe model on word analogy syntactic dataset. It affects most in case of Indian languages like Hindi, Marathi, as these languages are morphologically divergent. For example, in Marathi, root word *'kar'* (Do) takes number of different forms like *'Karane'*, *'Karato'*, *'Karate'*, *'Kartat'*, *'karat'*, *'Karun'* etc. depending upon the subject, tense, type of sentence.

Again, nearest neighbor test is conducted on GloVe embeddings for English language trained on English corpus [11] and results obtained are given in table 2.

The proposed **Adaptive GloVe Model (AGM)** integrates morphological information of words in the original GloVe model. Lemmatizer and Part of Speech tagger are used to find root word and PoS tag of all words in the corpus.

Stanford CoreNLP natural language processing toolkit [14] is used for English and Lightweight Hindi Stemmer [19] and Hindi PoS tagger available on CDAC website [26] is used for Hindi.

Using the root words and POS tag information, groups are created of word pairs, viz, {play-playing, cut-cutting}, {play-played, sing–sang}. Similar groups are created for Hindi language as well viz. {*'Kha'* (Eat) - *'Khana'* (To eat), *'Khel'* (Play) -

'Khelna' (To play)}. One distinct co-occurrence value is assigned for each group. Keeping co-occurrence values near to MC i.e., maximum co-occurrence value from the matrix C, the proposed AGM has ensured that syntactically similar words have highly similar vectors.

Given word	Most similar words returned by GloVe embeddings
select	('choose', 0.755), ('specify', 0.661), ('Wait…', 0.645), ('Please', 0.645)
reach	('reached', 0.669), ('adulthood', 0.664), ('within', 0.647), ('attain', 0.637)
refuse	('accept', 0.659), ('do', 0.650), ('believe', 0.649), ('Grosso', 0.639)

Table 2: Results of nearest neighbor test on GloVe embeddings

ALGORITHM 1: AGM Algorithm

```
1. Count distinct words from corpus.
2. While traversing through corpus, find root words and pos tags of all words.
3. Using morphological information obtained in step 2, form groups of words according to their pos tags.
4. Create co-occurrence matrix from corpus. Find max co-occurrence count value MC.
5. Using groups created in step 3, add entries to co-occurrence matrix for all words in the groups.
6. For all groups:
    For all word pairs (word k, word l) in group i:
        a. Co-occurrence value = MC * i
        b. Add entry to co-occurrence matrix:
            (word k, word l, co-occurrence value)
7. Pass updated co-occurrence matrix to original GloVe model for generating word embeddings.
```

4 Adaptive FastText model (AFM)

The original FastText model [24] is modified for better performance. The contribution involves correcting vocabulary building process of original model. Original model didn't separate punctuation marks from words which usually results in errors. This problem of not separating punctuation marks and words exists in original GloVe model as well.

Word analogy and nearest neighbor test is conducted on FastText embeddings for English and Hindi language available on [25]. Some of the results are shown below.

For example, it is observed that during word analogy testing, 'eye' was expected answer, FastText embeddings returned 'eye-'. '*chota'* (small) was expected answer and FastText embeddings returned *'1chota'*. Top ten nearest neighbors of *'ladka'* (boy) returned by FastText embeddings included erroneous words like '*ladka:'*, '*ladka00'*.

So, **Adaptive FastText Model (AFM)** is based on separating punctuation marks and words and thereby reducing errors.

5 Training and Evaluation datasets

For training purpose, Hindi monolingual corpus [11] is used for Hindi language. Statistics of training corpus is given in table 3.

As per the findings, word analogy dataset is not available for Hindi language. So, as a first task, Hindi word analogy dataset is created by taking help of domain experts and google translate to translate some sample data from English test dataset available on [23]. This dataset contains questions like:
A is to B then C is to __?
It includes syntactic questions like:
'uth' (Stand up): *'uthana'* (To stand) then *'kha'* (Eat): _?
And semantic questions like:
'Mata' (Mother): *'Pita'* (Father) then *'Beti'* (Daughter): __?
(Translations taken from translate.google.com.)

Language	Number of sentences	Number of words	Vocabulary Size
Hindi	45,075,279	844,925,684	4,099,606
English	1,492,827	20,666,377	250,782

Table 3: Statistics of training corpus

For English, test dataset available on [23] is used. The statistics of this test dataset is given in table 4.

Following Che's advise [7] four prediction choices are provided in Hindi word analogy test dataset. With typical analogy question like A:B::C:D, single prediction choice is not enough to show the quality of all four trained vectors.

Word **similarity datasets** for Indian Languages developed by [1] available on [27] is used for Hindi language and WS353 dataset [12] is used for English.

Language	Total Syntactic Questions	Total Semantic Questions
Hindi	8888	3408
English	10675	506

Table 4: Statistics of word analogy dataset

6 Results

Word embeddings formed using the proposed **AGM** is tested on word analogy and word similarity dataset and results are compared with word embeddings developed using original GloVe model, FastText model (**FastTextHin**), Adaptive FastText model (**AFM**) (trained on Hindi monolingual corpus [11] and FastText embeddings published on the website [25] (**FastTextWeb**) (trained on Wikipedia corpora and common crawl data).

6.1 Word Analogy Results:

Word analogy Results for Hindi language are given in table 5. The Adaptive GloVe model performed better than original GloVe model in syntactic analogy tests. Original GloVe model achieved 25.75% accuracy on syntactic data set. AGM achieved 34.68% syntactic accuracy improving the accuracy by **9%.** The interesting part is that, vectors created by AGM performed significantly better than FastTextWeb. On semantic data set also, AGM outperformed original GloVe model by some 0.08% accuracy. This is because AGM is based on morphological information of words which works at syntactic level not semantic level.

On specific syntactic analogy test, AGM performed exceptionally well. For example, on Hindi Plurals test, AGM achieved **89%** accuracy. On verb-forms set which include pairs like {*'karna'* (To do) – *'kar'* (do): *'Likhana'* (To write)– *'Likh'* (write)}, AGM achieved 52% accuracy. It can be observed that AGM is a high potential method and can be used in specific downstream applications.

The FastText embeddings, trained using Hindi monolingual corpus (FastTextHin), performed significantly better than FastTextWeb. The syntactic accuracy is increased by **14%** and semantic accuracy by 3%. The reason behind this is that FastTextWeb is trained using Wikipedia corpora and Common Crawl data which contains huge noise which is not the case with FastText embeddings trained on Hindi monolingual corpus. Further, AFM increased the syntactic accuracy of original FastText model by 0.8 %.

Model	Vector Size	Syntactic Accuracy	Semantic Accuracy
Original GloVe	100	25.75	20.85
AGM	100	**34.68**	**20.93**
FastTextHin	100	**40.5**	**24.44**
AFM	100	**41.3**	17
FastTextWeb	300	26.9	21.07

Table 5: Accuracy in percentage of different models on word analogy task

Similar test is conducted for English language also. AGM and original GloVe model are trained on English corpus developed by [11] and tested on syntactic word analogy test. Results are compared with FastText English embeddings available on [25]. Original GloVe model achieved 11% whereas AGM achieved 34.07% accuracy. However, FastText achieved 71% accuracy. The reason behind huge result gap is AGM and original GloVe model are trained on relatively small corpus (vocabulary 250,782). The experiment was conducted to check if AGM works better than original GloVe or not.

6.2 Word Similarity Results:

Results of different models on Hindi word similarity task are given in table 6. All word embeddings are trained on Hindi monolingual corpus [11] except FastTextWeb.

It can be observed from the table that proposed AGM reports same Spearman rank as that of original GloVe model. This shows that though AGM focuses more on making GloVe model morphologically aware, word relatedness or similarity information is not lost.

FastTextHin (vector size 300) achieved better results as compared to FastTextWeb, increasing the Spearman rank by 8. AFM (vector size 300) increased the Spearman rank by **2** than FastTextHin. AFM (vector size 100) achieved 63 Spearman rank, whereas FastTextHin (vector size 100) achieved just 45 Spearman rank. It means that simple change in vocabulary building process of FastText increased result by **18** Spearman rank. For example, for words like '*Bagh'* (Tiger) – *'Janvar'* (Animal) FastTextHin reported 5.3 similarity and AFM reported 7.5 similarity which is closer to expected similarity (7.6) according to word similarity dataset. For *'Grah'* (planet) – *'Akashganga'* (Galaxy), expected similarity was 6.6, AFM reported 6.5 whereas FastTextHin reported 5.5.

Model	Vector Size	S. R. correlation
Original GloVe	100	50
AGM	100	50
FastTextHin	100	45
FastTextHin	300	63
AFM	100	**63**
AFM	300	**65**
FastTextWeb	300	55

Table 6: Spearman rank correlation of different models on word similarity task.

7 Conclusion

In this work, word embeddings for Hindi language are developed using Adaptive GloVe model and Adaptive FastText Model.

AGM performed better than original GloVe model and FastTextWeb on word analogy and word similarity datasets.

FastText embeddings trained on Hindi monolingual corpus (FastTextHin) showed huge performance increase as compared to FastTextWeb. Adaptive FastText model increased the performance of FastText model on word analogy and similarity datasets.

8 Future Scope

Further work can be done to improve the accuracy of word embeddings developed using AGM and AFM. These word embeddings along with test datasets will be made available publicly, which can be used in improving accuracy of many downstream NLP applications. For now, AGM is dependent on the accuracy of the Stemmer. Lightweight Hindi stemmer is used in this work. Results can be improved with the use of lemmatizer.

It is needed to check if further improvement is possible in AFM embeddings by focusing on meaning of shorter words that may show up as n-grams of other words. Inherently, this would allow for capture meaning for suffixes / prefixes.

The present work is limited to Hindi language only and in future, it can be extended to other Indian languages.

REFERENCES

[1] Akhtar, S.S., Gupta, A., Vajpayee, A., Srivastava, A. and Shrivastava, M., 2017, April. Word similarity datasets for indian languages: Annotation and baseline systems. In Proceedings of the 11th Linguistic Annotation Workshop (pp. 91-94).
[2] Al-Rfou, R., Perozzi, B. and Skiena, S., 2013. Polyglot: Distributed word representations for multilingual nlp. arXiv preprint arXiv:1307.1662.
[3] Bhattacharyya, P., 2010. IndoWordNet. In In Proc. of LREC-10.
[4] Bhattacharyya, P., Bahuguna, A., Talukdar, L. and Phukan, B., 2014. Facilitating multi-lingual sense annotation: human mediated lemmatizer. In Proceedings of the Seventh Global Wordnet Conference (pp. 224-231).
[5] Bhingardive, S., Singh, D., Rudramurthy, V. and Bhattacharyya, P., 2015. Using Word Embeddings for Bilingual Unsupervised WSD. In Proceedings of the 12th International Conference on Natural Language Processing (pp. 59-64).
[6] Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T., 2017. Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, 5, pp.135-146.
[7] Che, X., Ring, N., Raschkowski, W., Yang, H. and Meinel, C., 2017, September. Traversal-free word vector evaluation in analogy space. In Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP (pp. 11-15).
[8] Grave, E., Bojanowski, P., Gupta, P., Joulin, A. and Mikolov, T., 2018. Learning word vectors for 157 languages. arXiv preprint arXiv:1802.06893.
[9] Lund, K. and Burgess, C., 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. Behavior research methods, instruments, & computers, 28(2), pp.203-208.
[10] Koehn, P. and Knowles, R., 2017. Six challenges for neural machine translation. arXiv preprint arXiv:1706.03872.
[11] Kunchukuttan, A., Mehta, P. and Bhattacharyya, P., 2017. The iit bombay english-hindi parallel corpus. arXiv preprint arXiv:1710.02855.
[12] Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G. and Ruppin, E., 2002. Placing search in context: The concept revisited. ACM Transactions on information systems, 20(1), pp.116-131.
[13] Levy, O. and Goldberg, Y., 2014. Dependency-based word embeddings. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (Vol. 2, pp. 302-308).
[14] Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. and Mc- Closky, D., 2014. The Stanford CoreNLP natural language processing toolkit. In Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations (pp. 55-60).
[15] Mikolov, T., Chen, K., Corrado, G. and Dean, J., 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.
[16] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J., 2013. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems (pp. 3111-3119).
[17] Pennington, J., Socher, R. and Manning, C., 2014. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).
[18] Qi, Y., Sachan, D.S., Felix, M., Padmanabhan, S.J. and Neubig, G., 2018. When and why are pre-trained word embeddings useful for neural machine translation? arXiv preprint arXiv:1804.06323.
[19] Ramanathan, A. and Rao, D.D., 2003, April. A lightweight stemmer for Hindi. In the Proceedings of EACL.
[20] Revanuru, K., Turlapaty, K. and Rao, S., 2017, November. Neural Machine Translation of Indian Languages. In Proceedings of the 10th Annual ACM India Compute Conference on ZZZ (pp. 11-20). ACM.
[21] Rohde, D.L., Gonnerman, L.M. and Plaut, D.C., 2006. An improved model of semantic similarity based on lexical co-occurrence. Communications of the ACM, 8(627-633), p.116.
[22] Singh, D., Bhingardive, S., Patel, K. and Bhattacharyya, P., 2015. Detection of multiword expressions for hindi language using word embeddings and wordnet-based features. In Proceedings of the 12th International Conference on Natural Language Processing (pp. 295-302).
[23] Glove Code https://github.com/stanfordnlp/GloVe
[24] FastText Code https://github.com/facebookresearch/fastText
[25] FastText Vectors https://fasttext.cc/docs/en/crawl-vectors.html
[26] cdac http://kbcs.in/tools.php
[27] Word Similarity Dataset https://github.com/syedsarfarazakhtar/Word-Similarity-Datasets-for-Indian-Languages