An Approach for Predicting Related Word for the Hindi Language

Monika Sharma
Department of Computer Science & Engineering
Malaviya National Institute of Technology, Jaipur

Dinesh Gopalani
Department of Computer Science & Engineering
Malaviya National Institute of Technology, Jaipur

Meenakshi Tripathi
Department of Computer Science & Engineering
Malaviya National Institute of Technology, Jaipur

ABSTRACT
Without motivation, writing may be a cumbersome process. In this work, a methodology is proposed which will assist user by providing some reference information e.g. related words while composing an article or message. Smart systems with related word prediction have turned out to be extremely prevalent for English language but there is no such big efforts for Hindi language. The main goal of this dissertation work is to provide syntactically and semantically related words based on continuous feature vector representation. Continuous Bag of Words (CBOW) language model is used to get the feature vector representation of each word in training set. Cosine Distance and rule based strategy is used as measurement to find the most related word in context. In a comparative study we reasoned that our method excels in accuracy estimation than existing method. This approach will help Hindi writing in an effective and creative manner.

General Terms
Natural Language Processing, Neural Network.

Keywords
Language Modelling, Curse of Dimensionality, Distributed Representation, CBOW, POS tagging.

1. INTRODUCTION
To improve the performance of various natural language applications, statistical language modelling is required to capture the regularities of natural language. Statistical language models generate a probability distribution to assign a probability to various linguistic units such as words, sentences or whole documents.

Language modelling is critical in many NLP applications like machine translation, speech recognition, information retrieval, word sense disambiguation, POS taggers etc. Speech recognizers profit from a probability assigned to the next word in a speech sequence to be predicted. In Machine Translation systems, an LM is used to tune the probability scores of outputs by the system in order to improve for the actual grammaticality and smoothness of a translation in the target language.

While dealing with natural languages, SLM must employ techniques to estimate the large number of parameter due to categorical nature of natural languages and the large vocabulary size people naturally use. One of the standard and successful earlier technique of language modelling is n-gram model [1]. Which concatenates the short overlapping sequences seen in training set. This model take advantage of word ordering and the fact that words which are temporally closer are statistically more dependant. So, in the mathematical representation, it will take the combination of previous n-1 words in context.

$$P(w_t|w_1^{t-1}) \approx P(w_t|w_{t-n+1}^{t-1}) \quad (1)$$

For 2 grams it will calculate the probability of next word as follows:

$$P(w_t|w_{t-1}) = \frac{count(w_t, w_{t-1})}{count(w_t)} \quad (2)$$

But in recent years, there is massive amount of text of various type have become available online. As the size of data gets large, quality of language model should be improved.

N-gram language model cannot handle large amounts of data due to curse of dimensionality. Curse of Dimensionality in terms of language modelling can be defined as when number of parameters of the model increases exponentially with number of variables in input. Due to this issue sometimes n-gram model will not be able to assign the probability to a word which has not seen in training. Assigning zero probability to a word should be avoided in any efficient language model.

Although many smoothing techniques have been applied to n-gram to avoid zero probability but still they are not giving results up to our expectation while dealing with large amounts of data.

To overcome the curse of dimensionality and get better generalization, neural network techniques applied to language modelling. It has been proved that neural networks with one hidden layer works as approximator function [1]. Hidden layer in neural network represents learned non-linear combination of input features which works as an approximator. Figure 1 below shows the basic neural network architecture. This model simultaneously learns distributed representation of words as well probability function for predicting next word. Distributed representation captures syntactic and semantic features of words which helps in generalizing well to word sequences which are not seen in training set.

Advantages over n-gram model: N-gram model do not take context farther than 2 or 3 words because as it increases the context parameters would be increased exponentially which further causes more complexity. But in neural networks distributed representation of words is learned which helps in reducing total number of parameters. In neural network number of parameters grows polynomially (Square) with the number of variables in input.

Moreover, n-gram do not take into account similarity of words. It does not count syntactic or semantic features of words. For example in sentence "The cat is running in the room" and "The dog is walking in the garden" Here Dog and Cat plays similar semantical and grammatical roles. But it take both of them differently.

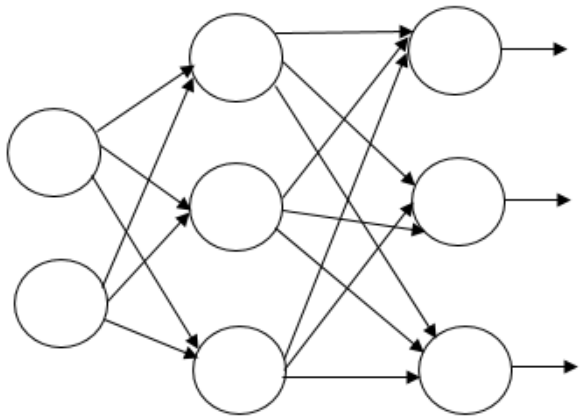

Fig 1: Basic neural network architecture (Universal Approximator)

For non-English users, Hindi input method is first tool for interacting with a computer. Quillpad [26], the first Indic transliteration solution to use statistical machine learning method for intelligently converting user entered free-style phonetic input to its accurate representation in a chosen Indian language. After that Google Input tools for Hindi Language came up. But such input methods do not provide any intelligent suggestion for writing. Antaryami [17] is first such kind of tool available for Hindi language which suggest the next word based on context. This tool is based on n-gram language model developed in 2013. This applications assists user by giving multiple suggestions for the next word possible.

The paper is organized as such that next section presents a review of existing neural network language models i.e. state of the art of the field. Details of the related work in the Hindi language is given in the next section. Subsequent section presents the proposed methodology to achieve the final objective. After this Results and Discussions are covered. Conclusion and Future work is summed up at the end.

2. NEURAL NETWORK LANGUAGE MODELS

Neural network functioning is based upon human brain. To learn the objects, our brain compare all active or inactive features of one object to others. Likewise as a language model, neural network learns distributed representation of each word by capturing all its features. In a vector space, each word is continuous valued vector representation corresponds to a point. All the word which have similar features are closer to each other in feature space and a sequence of words can be transformed into a sequence of learned feature vectors. Also, such functionally alike words can be replaced by each other in a word sequence of the same context.

This functionality helps NN to make the predictions about the next word in a given sequence and also help in predicting OOV (Out of Vocabulary) word forms.

In the early 90's, neural networks are popular learning techniques that was capable of learning syntactical and semantic meaning of words automatically during training. But they lost their popularity due to immense computation required during training phase that takes significant amount of time. But advent of faster and parallel computer architectures, made them in demand again and have been further developed into multilayer, recurrent or deep neural network architecture.

2.1 Feed Forward Neural Network Language Model

Bengio et al. [5] in 2001 proposed a feed-forward neural network language model with multiple input neurons and hidden units to capture the distributed representation of words efficiently as shown in figure 2. The methodology of this algorithm can be summarized as follows [5]:

- Associate distributional feature vector of each word in the vocabulary (a real valued vector in Rm).
- Express the joint probability function for word sequences calculated using feature vectors.
- Learn simultaneously the word feature vectors and the parameters of that probability function.

The feature vector represents different aspects of the word: each word is associated with a point in a vector space. The number of features of word like connotations, POS etc. are much smaller than the size of the vocabulary (e.g. 17,000). The probability function is expressed as a product of conditional probabilities of the next word given the previous ones.

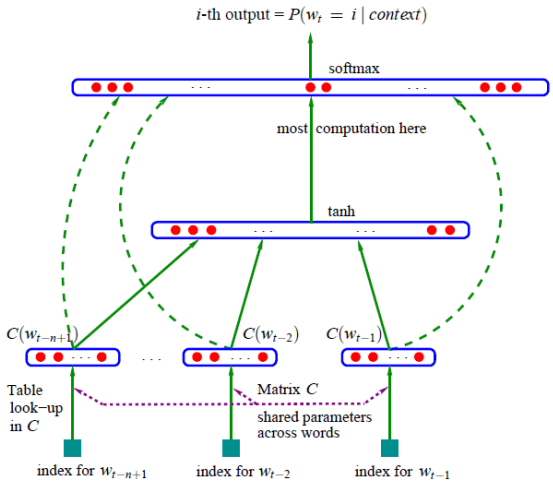

Fig 2: Feed forward neural network architecture [5]

The computational complexity of feed forward neural network per each training example is [16]:

$$Q = N * D + N * D * H + H * V \quad (3)$$

Where the dominating term is of H*V of output layer. To avoid it we may use hierarchical version of softmax layer in which total output units need to be evaluated reduced to log V [6].

Problems:

- Computation is costly.
- Training and Testing time is long.
- Use fixed length context that needs to be specified ad hoc before training.

2.2 Recurrent Neural Network Language Model

Since FNNs have fixed number of input neurons so only the words which are represented by input neurons are used to predict the next word and all words which were presented during previous iteration are forgotten. These previous words can play a significant role in learning the context and estimating suitable next word.

To overcome the issue, Elman (1990) [15] proposed variant of FNN where extra neurons are incorporated that are connected to hidden layer like other input neurons. These extra neurons are termed as context neurons and hold the contents of one of layers as it existed while the previous pattern was trained. Training of the network would be done in same way as previous.

To have the indefinite length of contexts, recurrent version of neural network is used. Micklov et al. [10] in 2010 proposed a language model based upon recurrent neural network approach with application to speech recognition. By using recurrent connections, information can cycle inside these networks for arbitrarily long time. Figure 3 depicts the process of RNNs. They have used simplest architecture of recurrent neural network, very easy to implement and train. The network consist of input layer x, hidden layer (context, shared) layer s, and output layer y. Input, hidden and output layer function are calculated as follows [10]:

$$x(t) = w(t) + s(t-1) \quad (4)$$

$$s_j(t) = f(\sum_i x_i(t)u_{ji}) \quad (5)$$

$$y_k(t) = g(\sum_j s_j(t)v_{kj}) \quad (6)$$

Where f (z) is sigmoid activation function and g (z) is softmax function:

$$f(z) = 1/1 + e^{-z} \quad (7)$$

$$g(z) = e^z / \sum_k e^{(z_k)} \quad (8)$$

Advantage over feed forward neural network approach:

- Need to define less parameters before training as in we only have to define size of hidden layer. Where as in feed forward neural network we have to define parameters for projection layer as well.
- This network do not define size of context prior. Rather they encode temporal information implicitly for context with arbitrary length.

The computational complexity of recurrent neural network per each training example is [16]:

$$Q = H * H + H * V \quad (9)$$

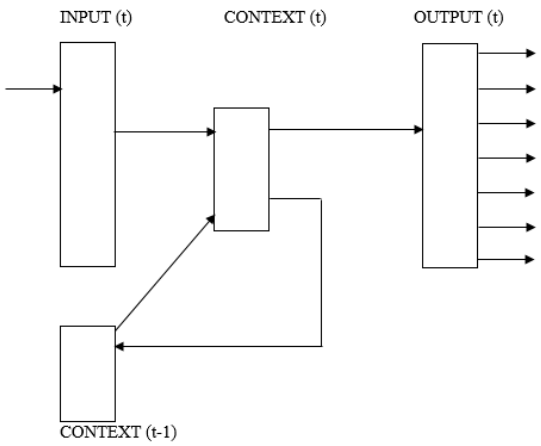

Fig 3: Architecture of Recurrent neural network language model

2.3 Continuous Bag of Words Language Model

Micklov et al. [16] in 2013 proposed a novel architecture for computing continuous vector representation of words as shown in figure 4. From the previously described model we have seen that most of the complexity is caused by the nonlinear hidden layer in the model. In this model hidden layer have been removed to reduce the complexity but they still give efficient result for large corpuses. It is alike the feed-forward NNLM, where the non-linear hidden layer is abolished and the projection layer is shared for all words (not just the projection matrix), thus, all words get projected into the same position. This architecture is called a bag-of-words model as the order of words in the history does not influence the projection. Training complexity is then [16]:

$$Q = N * D + D * \log V \quad (10)$$

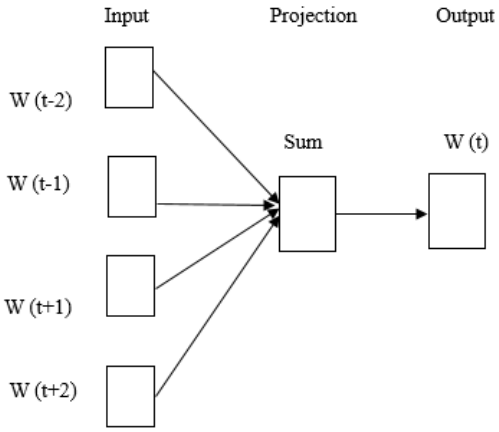

Fig 4: Architecture of continuous bag of words model (CBOW)

2.4 Continuous Skip Gram Language Model

The second log-linear architecture is identical to CBOW, but it tries to maximize classification of a word based on another word in the same sentence instead of predicting the current word based on the context [16]. In more precise manner, each current word is fed to input layer of log-linear classifier with

continuous projection layer, and predict words within a certain range (defined as parameter) before and after the current word as shown in figure 5. It is apparent that increasing the range directly impacts the quality of the resulting word vectors, but it also enhances the computational complexity. Since distance and relatedness to the current word is inversely proportional i.e. distant words are less related to the current word so less weight would be given to the distant words compare to close word by sampling less from those words in given training examples.

The training complexity of this architecture is proportional to [16]:

$$Q = C * (D + D * \log V) \qquad (11)$$

Where C is the range of context words. For example if we choose C = 4, for each training word a random number R in range < 1, C > would be selected, and then use R words from history and R words from the future of the current word as correct labels. This will require to do R * 2 word classifications, with the current word as input, and each of the R + R words as output.

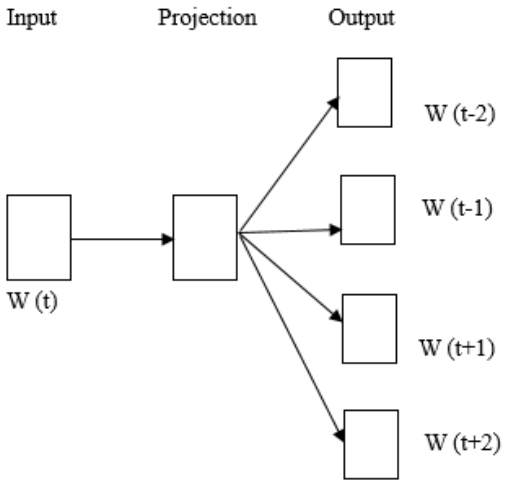

Fig 5: Architecture of continuous skip gram model

Table 1 shows comparison among language models described above:

Table 1. Comparison among state of the art language models

Parameters	N-Gram	FNLM	RNLM	CBOW	Skip-Gram
Curse of Dimensionality	Yes	No	No	No	No
Generalization	Less	High	High	High	High
Limited Context Size	Yes	Yes	No	Yes	Yes
Computational Complexity	Very Low	Very High	Very High	Low	Low
Prior Parameter Estimation	Low	High	Medium	Low	Low

2.5 Related Work

Various transliteration and synonym extraction technique have been developed in recent years for the Hindi language. In 2001 and 2002, under the leadership of Pushpak Bhattacharya, IIT Mumbai developed Hindi WordNet [28] based on Princeton University's English WordNet. Work have been done to get semantic information from Hindi word-net [8].

Similarly machine translation from English to Hindi language have been done [13]. Many transliteration techniques have been proposed. **Quillpad [26]**, the first Indic transliteration solution to use statistical machine learning method for intelligently converting user entered free-style phonetic input to its accurate representation in a chosen Indian language is developed in 2006. After that Google Input tools for Hindi Language came up. But such input methods do not provide any intelligent suggestion for writing. **Antaryami** [17] is first such kind of tool available for Hindi language which suggest the next word based on context. This tool is based on n-gram language model developed in 2013. This applications assists user by giving multiple suggestions for the next word possible.

3. PROPOSED METHODOLOGY

The proposed methodology has been depicted in figure 6 below:

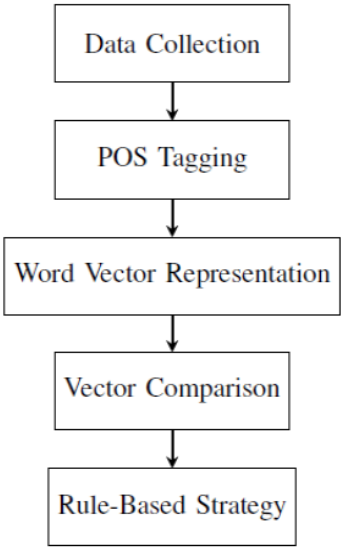

Fig 6: Proposed Methodology

3.1 Data Collection

First step is Data collection i.e. collecting Hindi data from various online resources and merge it all in one text file. Hindi data corpuses have been made available by CFITL, Mumbai [28] and Wikipedia Text Dumps available under CC-BY-SA-3.0 [32]. Moreover, data have been collected from Hindi News media sites and literature sites as well. All data files are merged into single fie. Table 2 shows data set description.

Table 2. Data Set Description

Total_Words	21635153
Vocab_size	77283
Dimensionality	200

3.2 POS Tagging

POS tagging is the process of assigning correct part of speech to each word of a given input text depending on the context. Tagging algorithm is based upon TnT tagging methodology [12]. TnT is a very efficient statistical part-of-speech tagger that is trainable on different languages and virtually any tag set. The tagger learns morphological analysis and pos tagging at the same time.

3.3 Word Vector Representation

To get the word vector, CBOW model is used. This language model is faster than others and also shows good performance in terms of perplexity and word relatedness.

3.4 Vector Comparison

After obtaining the distributed representation of each word in the vocabulary, to get the most related word in context, a comparison is performed between the input word and all other words in the vocabulary. To compare two vectors, Cosine distance is used as measurement.

After calculating the distance, Ranking of words will be organized in Descending order. Total 100 words closest words will be considered for further evaluation in next step.

3.5 Applying Rule Based Strategy

After getting the 100 closest word vectors, we re-rank them according to their part of speech [14] to maximize the accuracy. Rule-Based strategy would be as given below in table 3:

Table 3. Rule based Strategy

PoS of Original Word	PoS of Recommended Word
Adjective	Noun
Pronoun (Relative/ Reflexive/Personal)	Noun
Noun	PSP, Verb, Noun
Main Verb	Auxiliary Verb

4. RESULT EVALUATION

It has been obvious that prediction for words with initial positions would not be as accurate as it should be due to lack of context. But CBOW model has advantage that in absence of context it predict the most related word of original word. For example if we type the word "Pooja", it will give all the diverse related word of it. Table 4 below shows result.

Table 4. Predicted words for the word "Pooja"

Most Probable Predicted words	Distance	POS
upasana	0.788	NN
Aradhana	0.786	NN
Vandana	0.717	NN
Prarthana	0.681	NN
Poojan	0.678	NN
Stuti	0.661	NN

We have a separate test file to check the outcome. This test file has total 100 syntactic and semantic questions formulated. In each question we have given 4 related words of each sample word. Accuracy is measured by harnessing linguistic regularities of vectors generated by NN models [18]. Word vectors calculated by those models capture meaningful semantic and syntactic regularities in a very simple way. For example to answer the question we first have to find out the embedding vector of 3 words for each question in test set. Suppose we have $\mathbf{x_a}$, x_b and $\mathbf{x_c}$ the corresponding feature vector of a question. Then to find out the fourth word d we first calculate $\mathbf{y = x_b - x_a + x_c}$. y is the embedding vector of the word we expect to be the required answer. Of course, no word might exist at that exact position, so search for the word whose embedding vector has the greatest cosine similarity to y and match it to the fourth word d. If they match output it.

$$Accuracy = \frac{Total\ Words\ Seen}{Total\ Words} \qquad (12)$$

We compare our results with Antaryami [17] which is based on n-gram model. We can see from results in table 5 that this work stands out in giving related word because of well continuous feature vectors calculated by CBOW model.

Table 5. Comparative Study

Model_Name	Accuracy	Approach	CurseofDimensionality
Our_Model	67.92%	n-gram	No
Antaryami	52%	neural net	Yes

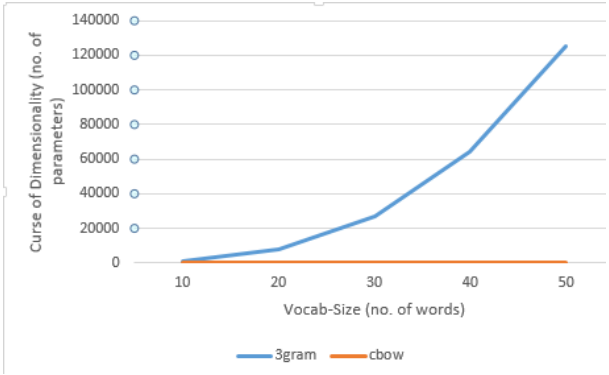

Fig 7: Comparison of both methods in terms of curse of dimensionality with context size 3. As the vocabulary size increases no. of parameters increases exponentially in 3-gram whereas polynomially (square) in CBOW.

5. CONCLUSION AND FUTURE WORK

In this paper a methodology is proposed with objective of suggesting related word based on previous context words in real time while writing in the Hindi Language. We are using CBOW architecture to get the word vector representation of Hindi word sequences. This is a novel approach based upon neural networks.

By going through the state of the art SLM that explore the use of NN's to statistical language model, we concluded that NN-LMs are very capable and timely contribution to learn generalization over a highly discrete space of natural language word sequences. When comparing to base n-gram model, they show huge reduction in perplexity and do not suffer from curse of dimensionality. Their ability to embed and cluster functionally and semantically similar words make them more useful.

In a comparative study, we observed that it is possible to train high quality word vectors using very simple architecture from a very large data set. CBOW has simple architecture compared to others (Feed forward and Recurrent NN-LMs) but exhibit better syntactic and semantic accuracy.

High quality word vectors can become a prominent building block for many NLP applications like automatic extension of facts from the knowledge bases and verifying correctness of existing facts, sentiment analysis, paraphrase detection and machine learning.

In order to make it more useful, we may opt for user modelling in which feedback from users are fed to the model in future. We can make our system better by adding it into crowd sourcing system and monitor the performance and usefulness of it to the user's writing. Furthermore, we can add a function which is based on query log of the search engine to

give more valuable suggestion to the users based on their history work. Moreover, although CBOW gives better result in short time but it suffers from proximity issues. Order information of context words doesn't matter. The ignorance of proximity causes a poorly positioned vector in feature space. Besides proximity, high quality word embedding also relies on linguistics. A single word may be belong to different multiple lexical categories. For example some words can be either a noun or verb. It would be hard to capture the syntactic regularities of such words in a single vector because the vector is required to be close to a number of nouns and verbs in the vector space. Such ambiguity must be addressed in representation learning in future.

6. ACKNOWLEDGMENTS

We would like to thanks all those who helped us in reaching our desired results directly or indirectly.

7. REFERENCES

[1] S. M. Katz. 1987 Estimation of probabilities from sparse data for the language model component of a speech recognizer. IEEE Transactions on Acoustics, Speech and Signal Processing, pp. 400–401.

[2] S. C. Douglas. 1998. Evaluation metrics for language models.

[3] S. Bengio and Y. Bengio. 2000. Taking on the curse of dimensionality in joint distributions using neural networks. Trans. Neur. Netw., vol. 11, no. 3, pp. 550–557.

[4] Y. Bengio. 2002. New distributed probabilistic language models. No. 1215.

[5] Bengio, Yoshua, Ducharme, R´ejean, P. Vincent, Janvin, and Christian, "A neural probabilistic language model," J. Mach. Learn. Res., vol. 3, pp. 1137–1155, mar 2003.

[6] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model," pp. 246–252, 2005.

[7] L. van der Plas and J. Tiedemann, "Finding synonyms using automatic word alignment and measures of distributional similarity," pp. 866–873, 2006.

[8] R. Nadig, J. Ramanand, and P. Bhattacharyya, "Automatic evaluation of wordnet synonyms and hypernyms," Proceedings of ICON-2008: 6th International Conference on Natural Language Processing., pp. 8–31, 2008.

[9] R. M. K. Sinha, "A journey from indian scripts processing to indian language processing," IEEE Annals of the History of Computing, vol. 31, no. 1, pp. 8–31, 2009.

[10] T. Mikolov, M. Karafi´at, L. Burget, J. Cernock´y, and S. Khudanpur, "Recurrent neural network based language model," pp. 1045–1048, 2010.

[11] J. Turian, D. D. Et, R. O. (diro, U. D. Montral, L. Ratinov, and Y. Bengio, "Word representations: A simple and general method for semisupervised learning," pp. 384-394, 2010.

[12] S. Reddy and S. Sharoff, "Cross language pos taggers (and other tools) for indian languages: An experiment with kannada using telugu resources," November 2011.

[13] E. Arisoy, T. N. Sainath, B. Kingsbury, and B. Ramabhadran, "Deep neural network language models," pp. 20–28, June 2012.

[14] N. Garg, V. Goyal, and S. Preet, "Rule based hindi part of speech tagger," COLING (Demos), no. 163–174, 2012.

[15] N. Pappas and T. Meyer, "A survey on language modelling using neural networks," no. Idiap-RR-32-2012, 2012.

[16] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space,"CoRR, vol. abs/1301.3781, Oct 2013.

[17] A. Das., "Antarym: The smart keyboard for indian languages,"In the Workshop on Techniques on Basic Tool Creation and Its Applications (TBTCIA 2013), ICON, no.1215, 2013.

[18] T. Mikolov, W. tau Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-2013), 2013.

[19] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," pp. 3111–3119, 2013.

[20] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," pp. 1188–1196, 2014.

[21] S. Sachdeva and B. Kastore, "Document clustering: Similarity measures," Project Report, IIT Kanpur, 2014.

[22] L. Qiu, Y. Cao, Z. Nie, Y. Yu, and Y. Rui, "Learning word representation considering proximity and ambiguity," AAAI Conference on Artificial Intelligence., June 2014.

[23] W. D. Mulder, S. Bethard, and M.-F. Moens, "A survey on the application of recurrent neural networks to statistical language modeling," Computer Speech Language, vol. 30, no. 1, pp. 61-98, 2015.

[24] D. Guthrie, B. Allison, W. Liu, L. Guthrie, and Y. Wilks, "A closer look at skip-gram modelling."

[25] Polyglot - Rami Al-Rfou - Google Sites "https://sites.google.com/site/rmyeid/projects/polyglottocdownload-wikipedia-text-dumps."

[26] Quillpad: http://www.quillpad.in/index.html.

[27] Fleksy Keyboard: http://fleksy.com/.

[28] Hindi WordNet (A Lexical Database for Hindi): http://www.cfilt.iitb.ac.in/wordnet/webhwn/.

[29] Universal approximation theorem: https://en.wikipedia.org/wiki/universalapproximationtheorem.

[30] Neural net language models: http://www.scholarpedia.org/article/neural net language models.

[31] SwiftKey: http://swiftkey.com/en/.

[32] Polyglot - Rami Al-Rfou - Google Sites "https://sites.google.com/site/rmyeid/projects/polyglottocdownload-wikipedia-text-dumps."