CS365: Word interpretation via Global semantics, Local Context and Multiple Word Prototypes

Chandra Prakash
pchandra@iitk.ac.in

Vishal Kumar Gupta
vishalkg@iitk.ac.in

Mentor: Prof. Amit Mukerjee
Department of Computer Science and Engineering

Indian Institute of Technology, Kanpur

Project Report
April 17, 2013

ABSTRACT

The availability of large quantity of data due to the openness to the web text has led to the presence of a huge number of homonymous and polysemous words whose interpretation needs to be automated for natural language processing tasks. So, there are models that can interpret the meaning of words that have multiple meanings. But most of these models earlier focused on the local context to interpret the word meaning. The model that we are working on presented a ”neural network architecture which learns word embeddings that better capture the semantics of words by incorporating both local and global document context, and accounts for homonymy and polysemy by learning multiple embeddings per word” [3]. But the work that has been done is for English dataset. To the best of our knowledge, we could not find any work that has been done for a corpus of Hindi articles. Our contribution in this project includes the creation of a large dataset of Hindi documents and then word interpretation of these Hindi documents. So, far we have not been able to compare our results to any manually tagged words or other dataset.

TABLE OF CONTENTS

1. INTRODUCTION

One of the major goals in AI has been to automatically extract the semantic knowledge from text data, based on the relationship between concepts [1]. There have been attempts to build a question answering system by the Natural Language processing community based on the understanding of the given words. But most of these works are not scalable because the words in these attempts have been manually interpreted to identify the meaning it represents and hence a large number of words cannot be interpreted.

Developments in Machine learning techniques and modern statistical approaches and the availability of huge corpus generated from automated crawling of web pages has contributed to the building of a system that can handle huge set of data with a very sparse matrix representation. Using vector space models (VSM), a vector can be constructed for each word which takes into account the most common pattern in which a particular word occurs. This, in some way, captures the semantics of the words. But this model just cannot distinguish the meaning of the word which is polysemous because each words is represented by a single vector, so, even if the word is used with a different meaning, it is represented by the same vector in VSM. In fact there are systems which take into account the VSM of the words and the local contexts of the words to form a representation in which a word is present with its local context [4]. From this a multi-prototype representation of the word is constructed using clustering on the context vectors. But even in this case, the syntax and the semantics of the words is not taken care of, at the same time.

In this work, we have used both, the local score and the global score to build an embedding matrix which is single prototype. The representation of a word captures, using a neural network model, the semantics of the word as well as the syntax of the word. The context, for clustering the words, has an improved formulation so that the global score is taken into account, apart from the local score.

The model is evaluated on a Hindi dataset built using all the Hindi Wikipedia articles. To the best of our knowledge no work has been done with a Hindi language dataset. The dataset itself poses certain problems likes non-availability of parts of speech tagger for Hindi dataset, which would have helped in identifying the stop words.

2. RELATED WORK

A spherical k-means clustering algorithm was suggested by Dhillion and Modha [2] which can be used to cluster matrices which can be more than 99 percent sparse. Vector Space Models can be used to represent words in the form of vectors using the statistical co-occurrence of words to form syntax for the word representation [1]. The matrix build using this vector space representation takes into account the frequency of the word and its relevance with the document but it cannot represent a polysemous word in two different ways according to their meaning because it forms a single vector representation for each word. A multiple prototype approach was developed by Reisinger andMooney [4], using the local contexts as vectors and them clustering these vectors to form representation in which one word can be present in more than one cluster.

But in this approach only local context was taken into consideration and global semantics was ignored. Then we have the "word representation model using global context and multiple word prototype" [3] which will be explained in the following sections.

3. Algorithm/Our Approach:

This section is basically divided into three sub-sections:

- Objective of training
- Neural Network Architecture
- Model's training method

a) Objective of Training:

Given a document, representing *global context*, and a small sequence of words from the document, representing *local context*, our model has to learn word representations along with discriminating it with next word. One of the key features of our model is that it has to learn meaningful word representation rather than probability of a word given previous words. This feature facilitated to utilize the entire document in deciding its global context.

Let's consider a word sequence *s* taken from document *d*. The goal of the model is defined in the following manner: We have to discriminate the last word in *s* from other words. Putting it more formally, we calculated scores in the form of $g(s, d)$ and $g(s^w, d)$ ensuring $g(s, d)$ is greater than $g(s^w, d)$. In the above statement s^w represented the same sequence with the last word replaced by word *w* and $g(., .)$ represents the scoring function realized through neural network. Finally the training objective boils down to minimizing the loss of ranking corresponding to each (s, d) taken from corpus. The objective can be formulated as:

$$C_{s,d} = \sum_{w \in V} \max(0, 1 - g(s,d) + g(s^w, d)) \quad \text{... [taken from Socher, 2012]}$$

As compared to calculating log-likelihood of next word, the above approach is found to be more efficient which ensures faster training as well as formation of better word embeddings [Collobert and Weston, 2008].

b) Neural Network Architecture:

In order to calculate global and local score here we have considered two neural networks which calculates $score_l$ and $score_g$ respectively and then the final score $score_f$ for a given word is given by sum of the two scores.

$$score_f = score_l + score_g$$

The local score is known to preserve the local context in the form of syntactic information while global score tries to capture semantic knowledge.

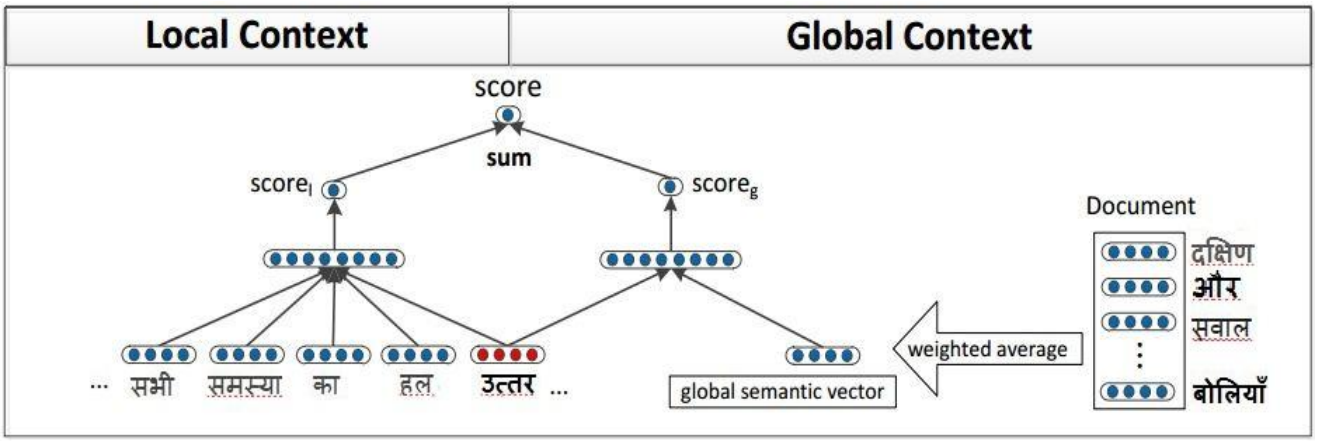

Fig. 1: Picture showing the neural network architecture for score calculation. According to figure the score of the word "**उत्तर**" should be large enough in comparison to other words. If the ambiguity for the word meaning is still unresolved, disambiguation is carried out using global context [taken from Socher, 2012].

Calculating Local Score: Word sequence *s* is represented as an ordered list of vectors $x = (x_1, x_2, \ldots, x_m)$ where x_i denotes the embedding of word *i* in the sequence. An embedding matrix $L \in R^{n \times |V|}$ is constructed. Here $|V|$ corresponds to size of the vocabulary and each column is word vector which is supposed to be learned using neural net. The eqn. representing neural network are as follow:

$$a_1 = f(W_1[x_1; x_2; \ldots; x_m] + b_1)$$
$$score_l = W_2 a_1 + b_2$$

…[taken from Socher,2021]

$[x_1; x_2; \ldots; x_m]$ is the concatenated *m* word embeddings corresponding to sequence *s*.

f is the activation function operating element wise. $a \in R^{1 \times h}$ is the activation vector for the hidden layer having *h* hidden units. $W_1 \in R^{h \times (mn)}$ and $W_2 \in R^{1 \times h}$ are 1st and 2nd layer weights respectively with b_1 and b_2 as their biases.

Calculating Global Score: For this the document is also considered as ordered list of word embeddings, $d = (d_1; d_2; \ldots; d_k)$. Weighted average document vector, *c* is calculated first and then concatenated with the vector of last word in *s* to give $[c; x_m]$.

$$c = \frac{\sum_{i=1}^{k} w(t_i) d_i}{\sum_{i=1}^{k} w(t_i)}$$
$$a_1^{(g)} = f(W_1^{(g)}[c; x_m] + b_1^{(g)})$$
$$score_g = W_2^{(g)} a_1^{(g)} + b_2^{(g)}$$

…[taken from Socher, 2012]

The final score is given by: $score = score_l + score_g$.

In both the neural networks idf-weighting is used as the weighting function.

c) ***Model's Training Method:***

For each sequence-document pair (*s, d*), a corrupt is chosen from dictionary randomly. Following this ranking loss derivative is calculated w.r.t embedding matrix *L* parameters and weights of Neural Network. Mini-batch-LBFGS resulted in formation of optimal embedding matrix as compared to stochastic gradient descent.

4. BUILDING MULTI PROTOTYPE MODEL

It's a known fact that a word is used in various contexts reflecting various meanings. Therefore a single representation can't be considered enough. Instead, if a model could be built through which we are able to represent a word in various contexts that would be much better. Multi-prototype approach for vector space models [proposed by Reisinger and Mooney, 2010b] are able to capture different senses and usages of words. Here we try to learn multiple context windows from the single prototype embedding learned previously during training phase. This is then followed by clustering for the purpose of word sense discrimination.

Initially, for a given word, fixed sized context windows are extracted for all occurrences. Using idf-weighting as the weighting function, each context is depicted as weighted average of the context words' vectors. The matrix so formed is very sparse in nature. In order to cluster them we used spherical K-means and built cluster for context representation. Finally each word in the corpus is relabeled to its corresponding cluster.

For calculating similarity between a pair of words using the above approach is given by

$$AvgSimC(w,w') = \frac{1}{K^2}\sum_{i=1}^{k}\sum_{j=1}^{k} p(c,w,i)\,p(c',w',j)\,d(\mu_i(w),\mu_j(w'))$$ . . . [taken from Mooney, 2010b]

p(c,w,i) = likelihood of word *w* lying in cluster *I* given context *c*.
$\mu_i(w)$ = vector representing the i^{th} cluster centroid of *w*.
d(*v*,*v'*) = similarity function to calculate similarity between two vectors.

5. EXPERIMENTS

a) DATASET:

The data set used by us is **Hindi data set**. We downloaded Wikipedia articles which consisted of around 1.6 lac articles related to various contexts. The python code deployed for constructing the data set is written by us. In an attempt to provide flexibility, we converted each word to ascii character code. So the methodology for building the data set is apt for any language. The concluding paragraph describes the format of dataset required for training.

For training a vocabulary file is needed whose each line is of the format: <word> <id> <frequency>. We assign an *id* to each word and frequency represents the number of times a word has occurred among all the articles. Further a document frequency file is needed whose each line tells in how many documents that word has come. A file in *.mat* format is required which contains stop words. Stop words are basically those words which are used to relate two words. For eg. In English, our stop words file will contain words like *in, from, at, to* etc. Due to absence of POS taggers in Hindi language, we constructed our stop words with those words which are most frequent. The threshold for selecting those words is fixed at 600. And at last but not the least, a corpus file is needed which accounts for all the words in the articles by their ids as included in the vocabulary file.

b) Coding Tools And Techniques:

For building the data set we wrote a python code which took all the articles (downloaded from Wikipedia) as input and constructed all the necessary files as *.txt*. Then the training is done on MATLAB. The relevant codes are available at [6]. We did the training for dataset constituted from 25 articles and 10000 articles. The embedding size is taken as 50. Number of words before the target word is 10 for describing the local context. For each of the neural nets, number of hidden units is taken as 100. In case of 25 articles all the words with frequency greater than 10 are considered as stop words and in case of 10,000 articles the frequency threshold for stop words was fixed at 600.

After training, KNN search is applied on the embedding matrix for 5 nearest neighbor to find closely associated words. The inbuilt function in MATLAB, *knnsearch* is used for this purpose using cosine distance.

For clustering, we used spherical k-means and the code is available at [5]. We produced 50 clusters from the data set of 25 articles.

6. RESULTS

देवियाँ स्मिरना कैपिटा स्त्री मिरामर्स वहीदा
नरहरि गैलोवे अंबेडकर बादलसिंह कृष्णराय दारुण
लिथुआनिया नॉर्थम्बरलैंड एमब्रोजियो पेंटागन रौवेना न्यूयॉर्क
कक्षाओं चबूतरे याज़िद गुणसूत्रों चबूतरा
कोरोनल टेरिटोरियल गुरुत्वकेंद्र पुनर्विस्तार न्यायचंद्रिका चंद्
कुकर्मो पट्ट। कुवेन्दी दर्दभरी मैली माटि

The above figure shows the results of KNN search. As we can see the 1st row of text shows words which are generally used in female context. The 2nd contains names of person. The 3rd row has words which represent names of places. The 4th row has words which are generally used in terrestrial context. These are the results obtained from word embeddings of 10,000 articles.

For clustering, given the time constraint, we were able to cluster only a set of 25 articles. We were not able to find clusters for dataset corresponding to 10,000 articles because the time complexity for clustering was very high. Also the results of clustering for 25 articles were also not good enough because of the facts mentioned in the next paragraph.

For having words falling in different clusters to develop different context and good semantic relation we must be having a lot of information about words in the form of different articles. While forming clusters for 25 articles we rejected all those words having frequency greater than 10. So, each word under consideration has a frequency less than 10. With such less information (25 articles only!!), we can't have good clusters for a word. In other way we can say that with such less number of articles it is difficult to have words falling in more than one cluster. Even if it falls for some words, visualizing good semantic relations will not be efficient.

7. CONCLUSION AND FUTURE WORKS

We conclude that the model so built is generalizable in any language. Our contribution in the project is the creation of a huge dataset in Hindi which can further be worked on. Since we haven't worked on large data set, carrying out the work on large data set would be certainly fruitful. We can expect quite good results in that case.

Future works include developing efficient method for separating stop words. With that the quality of data set would also improve and finding clusters will be efficient.

References

[1] Turney Peter D, Pantel Patrick, et al. From frequency to meaning: Vector space models of semantics. Journal of Artificial Intelligence Research, 37(1):141–188, 2010.

[2] Inderjit S Dhillon and Dharmendra S Modha. Concept decompositions for large sparse text data using clustering. Machine learning, 42(1-2):143–175, 2001.

[3] Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. Improving word representations via global context and multiple word prototypes. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1, pages 873–882. Association for Computational Linguistics, 2012.

[4] Joseph Reisinger and Raymond J Mooney. Multi-prototype vector-space models of word meaning. In Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pages 109–117. Association for Computational Linguistics, 2010.

[5] http://www.mathworks.in/matlabcentral/fileexchange/32987-the-spherical-k-means-algorithm/all_files

[6]http://www.socher.org/index.php/Main/ImprovingWordRepresentationsViaGlobalContextAndMultipleWordPrototypes