CS7015 Deep Learning
Programming Assignment 2
Word Representation for Bangla Language

CS17S008 Nitesh Methani
CS17S013 Pritha Ganguly

March 6, 2018

Contents

1 Introduction

In this assignment, we have attempted to train word vector representations for the Bangla language. For the above task, the training has been done on three models namely Continuous Bag of Words(CBOW), Skip-gram and GloVe models, out of which the first two are prediction-based models whereas the last one is a combination of both count and prediction based model. We have also designed our own test cases consisting of various analogies and similarities to evaluate the representations.

2 Makorsha

To facilitate our task of Bangla word vector representations, we have created a web-crawler named Makorsha (spider in bengali) using the python library BeautifulSoup.
A crawler should satisfy the following properties [NLP] :

- It should have the ability to execute in a distributed fashion across multiple machines.
- It's architecture should permit scaling and must be modular.
- It should make efficient use of various system resources.
- It should be biased towards fetching "useful" pages first.
- It should operate in continuous mode, i.e it should obtain fresh copies of previosuly fetched pages.
- It should be able to cope with new data formats, protocols, i.e it should be extensible.

Makorsha tries to satisfy most of the above properties.

3 Data Collection

We have crawled Bangla data from the following websites:

- https://bn.wikipedia.org/wiki/
- http://www.anandabazar.com/
- http://www.rabindra-rachanabali.nltr.org/node/1
- http://www.sarat-rachanabali.nltr.org/
- http://www.bankim.rachanabali.nltr.org/

The first link is the bangla version of Wikipedia. The second link provides data from the news domain whereas the last three links provide us with data on bangla literature.

4 Data Cleaning

We have observed that in all of the links the bengali text appears within the paragraph tag of the corresponding HTML page. Therefore, Makorsha only extracts the text content of the above tag. For the task of training, all sorts of punctuation marks, non-bengali text, special symbols from the raw data was removed; in short data was cleaned in order to build a Bangla corpus. As the bangla text is encoded as non ASCII strings, we also encoded it to bytes under the **UTF-8** encoding scheme and stored it in a single text file named "corpus.txt".

5 Corpus Description

Depending on our training model, we modify the corpus such that it either consists of newline separated unpunctuated sentences for the two prediction based model(CBOW and Skip-Gram) or a single space separated word-list for the GloVe model.
The corpus consists of a total of **3207055** sentences spanning around **40** million tokens which has the vocabulary size of **721230** words.

6 Word Embedding Models

We have used the existing models (C-code) of CBOW, Skip-Gram [word2vec] and GloVe [GloVe] to train our word embeddings. Each of the models provided some hyperparameters which we tuned as a part of our experiments. These hyper-parameters cab be categorized into the following categories [word] :

- Pre-processing Hyper-parameters : These modify the input to the algorithm, for example, context window size, removing stop words, etc.
- Association metric Hyper-parameters : These control the interaction between the word and the contexts.
- Post-processing Hyper-parameters : These modify the resulting word vectors, which includes vector normalization, etc.

(More on the above is described in section 8). For evaluation purposes, we have used the Python code provided by these models.

7 Python2 to Python3

For CBOW and Skip-gram, no modifications were required. For GloVe, the code for evaluation (evaluate.py))was written for Python 2. We made the following modifications to the existing code to make it compatible with Python 3.

- Line 16 was changed to : vectors[val[0]] = list(map(float, vals[1:]))
- Line 24 was changed to : for words,v in vectors.item():
- Line 59 was changed to : for i in range(len(filenames)):
- Line 70 was changed to : for j in range(num_iter):
- Line 78 was changed to : for k in range(len(subset)):

8 Hyper-parameter Space

The Hyperparameter Space is controlled using the following:

- **Window** - This hyperparamter controls the context window size. Our window is symmetric i.e. we are considering both the left and the right context.
- **Size** - This hyperparamter controls the embedding size of the word representation.
- **negative** - This hyperparamter controls the number of negative samples for each word in the vocabulary.
- **hs** - If this flag is set instead of negative sampling, hierarchical softmax technique is used.
- **alpha** - The negative samples in GloVe are sampled according to the smoothed unigram distribution. In order to smooth the original contexts' distribution, all context counts are raised to the power of alpha.
- **iter** - This flag is used to control the number of iterations of the gradient descent algorithm.
- **lr** - This flag specifies the learning rate used in learning algorithm. (Note that lr is getting annealed).
- **w+c** - This flag allows us to add word vectors to the context vectors as part of the embedding output. Adding context vectors effectively adds first order similarity terms to the second order similarity function.
- **nrm** - This flag allows us to normalize all vectors to unit length. Doing this makes the dot production operation equivalent to cosine similarity.
- **min-count** - This flag discards the words from the vocabulary whose count is less than the min-count.

9 Experiments

For this assignment, we have experimented with the following hyperparameters :

- Window - We have experimented with three values - 1, 5, 10, 15.
- Size - We have experimented with three values - 50, 100, 150.
- negative - We have experimented with three values - 3, 10, 25.
- hs - We have also used this flag.
- alpha - We have experimented with two values - 0.75(smoothed) and 1(unsmoothed).

Other than the above, there are many more hyperparameters which could be tuned but we couldn't use them as they were already integrated in the existing models. So we use the default values for iter = 15, lr = .05 (GloVe, CBOW), 0.025 (Skipgram), min-count = 5.

10 Output of the models

To check the output in order to evaluate the models, we have created three separate text files to test the semantic, syntactic relatedness of the word vectors.

- **semantic_analogy.txt** : Given two words which share a semantic relationship, the model tries to predict the nearest neighbor of the third word which satisfies the same/similar relationship as the first two words.
- **syntactic_analogy.txt** : Given two words which are semantically equivalent but differs only in the syntax, the model tries to predict the nearest neighbor of the third word which satisfies the same syntactic relationship as the first two words.
- **word_similarity.txt** : Given a word and some candidate synonyms, the model ranks the candidates in order of their cosine similarity with the word.

The snapshot of the Skipgram model (with default values) running on the above files is given below:

Semantic Analogy

Input			Cosine Similaritly	
A	B	C	Predicted	Expected
সবুজ	জঙ্গল	বরফের	পাহাড় (0.638119)	পাহাড় (0.638119)
শুরু	শেষ	সকাল	সোয়া (0.697746)	রাত (0.624246)
বাবা	মা	ছেলে	বেগমের (0.789611)	মেয়ে (0.730510)
ছেলে	মেয়ে	ভাই	বোনের (0.823297)	বোন (0.715920)

(a) Semantic Analogies

Syntactic Analogy

Input			Cosine Similaritly	
A	B	C	Predicted	Expected
এক	একটি	দুই	দুটি (0.759406)	দুটি (0.759406)
আমি	আমাদের	তার	মানুষের (0.649719)	তাদের (0.626114)
সালে	সালের	বছর	মাস (0.792077)	বছরের (0.761156)
সাহিত্য	সাহিত্যের	উপন্যাস	উপন্যাসটি (0.536863)	উপন্যাসের (0.486483)

(b) Syntactic Analogies

Word Similaritly

Input	Cosine Similaritly	
A	Predicted	Expected
সাথে	সাথেও (0.736058)	সংগে (0.643130)
অনেক	খুব (0.705742)	বেশি (0.692236)
কিংবা	এমনকি (0.782413)	বা (0.758016)
ব্রেকফাস্ট	পুষ্টিকর (0.582476)	সুস্বাদু (0.540185)

(c) Word Similarities

Figure 1: Output of Skipgram Model

As we can see from figure (a), if the input is (first row in the table)

Shobuj (Green) : Jongol (Jungle) :: Borofer (Snowcapped) : ?

the model predicts **Pahar(Mountain)** as expected whereas if the input is (second row in the table)

Shuru(Start) : Shesh (End):: Shokal (Day) : ?

the model predicts **Shoya (Sleep)** as opposed to the expected answer **Raat(Night)**. This suggests that some tuning is needed. Also, to compare the outputs of different models we have to define some evaluation metric.

11 Evaluation metric

Let *count* be the number of times the expected word appears in the top 10 predictions of the model. Let $\mathcal{T}$ be the total number of test cases. Therefore, we can define an evaluation metric for a model as :

$$\boxed{Accuracy(in\%) = \frac{count}{\mathcal{T}} \times 100}$$

Now to compare the performance of different models we are comparing the cosine similarity of the expected word for each model. **Higher the cosine similarity better the performance.**

12 Evaluation

The models were trained on the Bangla corpus according to the parameters specified in Section 9. Then the vector representations of the words were tested on different test cases which captures the semantic and syntactic similarity of the words. The output of the three models is then compared using the evaluation metric. The model accuracy on different set of parameters is shown below:

CBOW (cbow = 1)

Hyperparameters			Dataset		
Size	Window	negative	Word Similaritly	Semantic Analogy	Syntactic Analogy
50	1	03	93.75 %	27.20 %	85.70 %
100	1	03	62.50 %	27.20 %	100.00 %
150	1	03	87.50 %	36.30 %	100.00 %
100	5	03	81.25 %	54.50 %	100.00 %
100	15	03	87.50 %	45.40 %	85.70 %
100	10	03	87.50 %	45.40 %	85.70 %
100	10	10	75.00 %	54.50 %	100.00 %
100	10	25	87.50 %	45.40 %	100.00 %

(a) CBOW Representation

GLOVE

Hyperparameters		Dataset		
Size	Alpha	Word Similaritly	Semantic Analogy	Syntactic Analogy
50	0.75	43.75 %	36.30 %	57.10 %
100	0.75	62.50 %	36.30 %	57.10 %
150	0.75	50.00 %	27.27 %	57.10 %
50	1.00	31.25 %	36.30 %	57.10 %
100	1.00	31.25 %	36.30 %	28.57 %
150	1.00	31.25 %	45.45 %	14.20 %

(c) GloVe Representation

Skipgram (cbow = 0)

Hyperparameters				Dataset		
Size	Window	hs	negative	Word Similaritly	Semantic Analogy	Syntactic Analogy
50	1	0	03	75.00 %	45.40 %	100.00 %
100	1	0	03	81.25 %	36.30 %	85.70 %
150	1	0	03	75.00 %	36.30 %	100.00 %
100	5	0	03	50.00 %	45.40 %	100.00 %
100	15	0	03	62.50 %	36.30 %	42.80 %
100	10	0	03	68.75 %	36.30 %	42.80 %
100	10	0	10	62.50 %	45.40 %	100.00 %
100	10	0	25	68.75 %	45.40 %	57.10 %
50	1	1	0	68.75 %	27.20 %	42.80 %
100	1	1	0	68.75 %	45.40 %	42.80 %
150	1	1	0	68.75 %	45.40 %	57.10 %
100	5	1	0	62.50 %	36.30 %	57.10 %
100	15	1	0	68.75 %	54.50 %	71.40 %
100	10	1	0	68.75 %	54.50 %	57.10 %

(b) Skipgram Representation

13 Observations

- **Pre-processing helps** : Diluting the frequently occurring words (beyond some threshold) really enlarges the contexts' window size for many tokens, because they cannot reach words that were not in their original L-sized window.
- **Memory vs Training time** : We observed that GloVe takes more memory while training whereas word2vec takes longer time to train because GloVe computes the co-occurrence matrix only once (depending on the vocabulary it can take huge amount of memory) whereas word2vec parses the sentences in an online fashion handling each co-occurrence separately. As a result of which GloVe can reuse the co-occurrence matrix even if embedding size gets changed whereas word2vec has to be trained from scratch after changing its embedding dimensionality.
- **Analogy task represents mixed results** : Skipgram with negative sampling (SGNS) performs very well in syntactic analogy task (compared to CBOW) whereas CBOW outperforms SGNS in word similarity task. In case of semantic analogy task, we observe that SGNS performs considerably better than CBOW.
- GloVe identified few semantic analogies like **Abhinetri (Actress) : Sridevi :: Barrister : AtulPrasad** which neither SGNS nor CBOW was able to identify (in all our experiments).
- **Post-processing helps** : We observe that normalizing the vectors and adding context vectors to the word vectors help in predicting words which tend to appear in similar contexts and/or words which appear in the context of each other.
- Increasing the window size helps.
- Increasing the embedding size does not help much.

14 Conclusion

In this assignment we studied and compared three algorithms for vector representation of words on Bangla corpus. We explored different hyperparameters and compared models based on the evaluation metric. We realized that in case if any model performs poor, instead of changing the model or collecting more data we should tune the parameters as they have a significant impact on the performance. More data always helps but it comes at the cost of more memory and time. All models perform nearly same for different configurations. So we should try the model with different settings in case of poor performance.

References

[1] Mitesh Sir's slides, *course slides*

[2] NLP Crawler Stanford *https://nlp.stanford.edu/IR-book/html/htmledition/features-a-crawler-should-provide-1.html*

[3] Improving Distributional Similarity with Lessons Learned from Word Embeddings, Omer Levy, Yoav Goldberg, Ido Dagan Computer Science Department, Bar-Ilan University, Ramat-Gan, Israel. *https://transacl.org/ojs/index.php/tacl/article/view/570/124*

[4] Global Vectors for Word Representation *https://github.com/stanfordnlp/GloVe*

[5] Making sense of word2vec *https://rare-technologies.com/making-sense-of-word2vec/*

[6] CBOW and Skip-Gram *https://github.com/tankle/word2vec*