Natural Language Processing

Final Term Project

SVD Based Methods for Word Embedding

Author :
Ehsan Mahmoudi
97543012

Instructor:
Dr. Shamsfard

February 12, 2019

Contents

List of Figures

List of Tables

1 Abstract

In recent years there has been significant increase in use of word embedding methods for use in NLP applications. Amongst those methods neural network based methods are the most popular ones. Although these methods are very popular these days we believe that still SVD methods still are quiet interesting methods to be investigated. In theory, it has been shown that method like Skip Gram with Negative Sampling (SGNS) is equivalent to performing SVD on shifted PPMI matrix. Although in English language SVD based methods has underperformed compared to SGNS methods. In this project we are going to elaborate same thing in Persian language. The main ideas in this project are based on works done by [4] [2] [1].

One of the best achievements of this project was to have an end-to-end embedding method that we all understand and can debug and work with it. Unfortunately most of the times that we use embedding method is through ready made tools and this create a black box for us in embedding. Most of pre defined tools have lots of side logics that is hard to capture or alter. As result of this project we now have a white box that we have control and understanding of it.

2 Introduction

In recent years there been a significant interest in word embedding methods. Among all methods the Word2Vec [3] has gained significant attention in the community. This attention is in way that almost Word2Vec methods (both Skip Gram and CBOW) became the de-facto standards of these field. In [1] The authors show a very interesting property on SGNS, that it is equivalent of performing matrix factorization over shifted PPMI matrix. Although in practice the implicit matrix factorization never outperform over SGNS, but at least gives us the clue that, methods with matrix factorization approach are still worthwhile to be explored.

2.1 Persian Language Embeddings

Unfortunately very few is done in Word Embedding evaluation in Persian language. This includes standard datasets and benchmarks to show were we are standing, regarding the performance of embedding methods. Before this

Table 1: Best result for Analaogy Task

Category	Total Tokens	fastext-skipgram-100
semantic-capitals	4691	3514
semantic-Whole to part	420	68
semantic-family	600	349
semantic-currency	6006	1298
syntactic-Comparative Adjectives	2756	1927
syntactic-antonym	506	239
syntactic-Superlative Adjectives	2756	1611
syntactic-verb	1964	708
Total	19699	9714

project we started a class assignment project to build up the grounds for evaluating the existing models. This activity included following:

- Build standard web scale corpora
- Build datasets to assess the embedding models for various tasks (Analogy, Categorization, Word Similarity)
- Write Scripts to evaluate the models against the datasets.

As result of the same project we aimed to build models to gain the highest scores in various tasks. But unfortunately the results were significantly below what we expected.

Compared to what **CITATION NEEDED** This result is one of the best results in Persian language. In that article the highest analogy performance reported is which is around 49. Although this result has been acquired because of high tolerance value in analogy task. It accepts the answer if it is in top 50 closest neighbours of the target vector.

This means in Persian language the baseline is much lower compared to achievements reported in English literature. This might have two main reasons. Either the available resources in Persian are not as rich as resources in English and or the structure of language requires different approach, for modelling.

In this project our aim is to explore the matrix factorization methods to see, if we can reach a higher ground compared to what has been achieved in other tasks.

We will be comparing our result with fasttext based methods here.

2.2 Word Embedding as Matrix Factorization

In [1] it is shown that SGNS is equivalent to perform matrix factorization on following matrix:

$$M^{PPMI}(i,j) = max(0, PMI(w_i, w_j) - k) \tag{1}$$

Where

$$PMI(w_i, w_j) = log\left(\frac{P(w_i, w_j)}{P(w_i)P(w_j)}\right) \tag{2}$$

Once we constructed the matrix M^{PPMI} then we can perform Singular Value Decomposition on it to get the embedding vectors:

$$M^{PPMI} = U\Sigma V^T \tag{3}$$

Now we can assume that $W^{WORD} = U\Sigma$ is going to be our embedded vectors.

2.3 Advantages of SVD Methods

3 Methodology

In this section we describe a little more technical details of the project that most be considered for successful implementation.

3.1 Sparsity

One of the items that are vital for this project to be successful is to keep the sparsity of the matrixes. The start of the work is from building cooccurrence matrix. And this matrix is sparse by nature. But the way that we implement the code is important as this will significantly will impact the performance of the algorithm. And if not implemented properly it might even make it impossible to run our algorithm in such scale.

We used python language and utilised *SciPy* library because it supports implementation of sparse matrixes and also sparse algorithms of SVD methods. SciPy offers different types of sparse matrices and each of them are good for a reason. Fine tuning the type of the matrix was a big challenge.

3.2 Datasets

For this project we used the Persian Wikipedia dump. The size of the corpus is around 2GB and it contains around 17 Million sentences. One of the drawback of this corpus is that it has lots of sentences which are related to editing the pages.

If we simply tokenize the corpus based on space to space tokenization method, the will be around 120,000 unique tokens. As this is going to significantly impact our methods, we have filtered the tokens that have less than 100 times occurrence in the corpus. This filter out all of the miss detected tokens in corpus.

3.3 Hyper Parameters

One of the advantages of SVD based methods is that they have less parameters compared to neural network based methods. But still there are hyperparameters.

3.3.1 Window Size

Window size is one of the most important hyper parameters in the setup. Because of time constraints we got the chance to test for window size 5 but still we have to explore other values of window size.

3.3.2 Negative Sampling (K)

When computing the shifted PPMI matrix the value K to be decreased from PMI value is another hyper parameter that from experience it looks to have important impact on the result of embedding.

3.3.3 Embedding Dimension

One advantage of using SVD methods is that it can give us a clue about what is the optimum dimension for the embedding space. As we are extracting the eigen values of the matrix, we can set the optimum dimension where the eigen values are approaching small values or we face a plateau in values. This reduces one of the dimensions of our hyper parameter space.

4 Results

In this section although the final result. Before we deep dive into the final result, I want to talk about the journey we had so far with this project.

4.1 Practical Challenges

As we have started the project from scratch, majority of the time has been dedicated to get the pipeline ready. Lots of bits and pieces been put together to have a running pipeline. Although as we will see later the results are not as the best results we have for embedding. But at least we have paved the road for next steps that need to be taken to fine tune these methods to be form new baselines.

4.1.1 Processing Time

Although the method itself is faster compared to SGNS and CBOW methods, but in this project just building a co-occurency matrix was taking around 20 hours. This was due to use of sparse matrices. Some study been done to fine tune this we finally reach 8 hour time to build the cooccurrence matrix. Still modification can been done to reduce this time to less than hour. One of the future steps for this project is to perform such optimizations to make the research much more feasible.

Another optimization done to do this job was to store intermediate states in the files. Significant amount of time was wasted on runtime errors that happened during the execution. So from some point I decided to store every output as a file so the process can resume from where it stopped or failed.

4.1.2 Embedding Vectors

Initially we were just using the eigen vectors as embedding vectors. which meant $W^{WORD} = U$. Unfortunately this produced a very poor result. To be honest this result was very close to random.

After doing research, and look at other articles we realized that we must incorporate the eigen vectors as well. So we tried $W^{WORD} = U\Sigma$. This led to much more acceptable result in our work. In footnote the [1] suggest to explore formulations like $W^{WORD} = U(\Sigma)^{\alpha}$ To play around the impact. From our learning it look like that the eigen values have significant impact on output.

Table 2: Hyper Parameters used for baseline run

Window Size	5
Minimum Frequency	100
K (Shifted PPMI)	5
Dimension	300
Tolerance for Analogy	50

4.2 Baseline Result

Considering the practical challenges we talked about. Number of full end-to-end execution the we have for this framework is limited. Actually by the time this report is written we have only one setting run that is at least working. Here we will be discussing the result of this run. But you should have it in mind that this is the baseline run and there is a plenty of room to improve as we have control over the full process.

4.2.1 Hyper Parameters

Here are the hyper parameters used for baseline run available on Table 2

4.2.2 Running Time

For this run it took around 8 hours to build the co-occurrence matrix. After that calculation of PPMI and SVD each took around 1 hours to finish.

For test part also we have difficulty as the number of token is high running each analogy test is taking in order of seconds and we have around 19000 analogy questions. This also took a long time from us to have end-to-end outcome.

4.2.3 Analogy Result

The result of analogy task can be found in Table 3

5 Discussions and Conclusion

As you have noticed the final result is under-performing existing models that we have. But the aim of this project was not to create a state of the art embedding model. We believe we have achieved following in this project:

Table 3: Beseline result for analogy task

Category	Total Tokens	fastext-skipgram-100
semantic-capitals	4691	1842
semantic-Whole to part	420	179
semantic-family	600	196
semantic-currency	6006	534
syntactic-Comparative Adjectives	2756	944
syntactic-antonym	506	221
syntactic-Superlative Adjectives	2756	876
syntactic-verb	1964	421
Total	19699	5231

- Trying new method for embedding and make sure the basics and pipeline is working
- Having a white-box approach to embedding method
- Building up the ground for next steps for embedding
- Make sure our method will work on web-scale dataset and it is feasible on existing processing machines that we have access to.

All of the above targets were met, now it's the time to take further steps. Here are the possible next steps regarding this project:

- Explore various combination of hyper parameters (Window size, K , ...)
- Explore matrices other than PPMI matrix. The suggestion could be to use Graph Laplacian methods to extract these values
- Work on other graph based pre-processing to extend the scope of this work to other areas.

References

[1] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In Z. Ghahramani, M. Welling, C. Cortes, N. D.

Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2177–2185. Curran Associates, Inc., 2014.

[2] Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *TACL*, 3:211–225, 2015.

[3] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[4] Roser Morante and Wen-tau Yih, editors. *Proceedings of the Eighteenth Conference on Computational Natural Language Learning, CoNLL 2014, Baltimore, Maryland, USA, June 26-27, 2014*. ACL, 2014.