CSE490UA Winter 2017
Assignment 2

Jason You [ID: 1427878]

February 2, 2017

Contents

1 Analogy Task

1.1 Introduction

In this assignment I used the pre-trained word embedding from Word2vec [Mikolov et al., 2013a,b] and GloVe [Pennington et al., 2014] to accomplish the word analogy task. In this writeout I want to reveal that these two word embeding approach works well in some corpus but not so well in the others. The Google question-words file is used to evaluate how each model works in different contexts.

1.2 Data Preparation

The training corpus for the Word2Vec model is the First billion characters from wikipedia (use the pre-processing perl script from the bottom of Matt Mahoney's page) [enwik9].

The word vectors for GloVe model is from the GloVe: Global Vectors for Word Representation, Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 50d, 100d, 200d, and 300d vectors).

Testing data provided from the course site.

1.3 Tools

All the coding part is done by the Python programming language, and the word2vec model in the Gensim package[Gensim Word2Vec] was used for the implementation of Word2Vec. The GloVe model is Based on the pre-trained word embeding vectors: `http://nlp.stanford.edu/projects/glove/`. The Numpy package in Python was used for vector manipulation.

1.4 Method

The main focus of this section is of the GloVe word embeding. I use

$$\mathbf{v}_1 - \mathbf{v}_2 \approx \mathbf{v}_3 - \mathbf{v}_4$$

to find the cosine similarity. The pattern of the question is a sequence of four words, and the mission is the predict the fourth word based of the vector/cosine similarity found based on the first three words.

1.5 Evaluation

The accuracy is based on each topic:

$$\frac{\text{Correct prediction}}{\text{Total word sequences}}$$

1.6 Result

The performance in differen model and contexts gives very different results, but some of them are acceptable.

Word2Vec:

capital-world	31.9% (53/166)
currency	11.1% (2/18)
city-in-state	6.2% (39/626)
family	74.6% (179/240)
gram1-adjective-to-adverb	26.3% (145/552)
gram2-opposite	55.3% (73/132)
gram3-comparative	80.3% (747/930)
gram6-nationality-adjective	56.7% (697/1229)

GloVe:

capital-world	21.93% (992/4524)
currency	2.54% (22/866)
city-in-state	26.92% (664/2467)
family	35.97% (182/506)
gram2-opposite	3.20% 26/812
gram3-comparative	21.85% 291/1332

My tests (Both Word2Vec and GloVe):

my-test-one-sport	0.0%
my-test-two-food	0.0%

1.7 Antonyms issue

Since the word embedding is mapping words to vectors, many cases can cause antonyms have very small cosine difference (angles bewteen vectors):

Case 1: Since word by itself don't have a direction, if we define a word to be a vector by some parameters, then it's very likely that we have same magnitude for antonyms, but the direction is not take into account. Thus though some words have totally opposite meanings, they appear in similar contexts, so their vector is almost pointing to the same direction

Case 2: Though this one might not be an implicit causation, the subtraction of different vectors might cause reverse of the direction of the result. For example:

$$\mathbf{a} - \mathbf{b} = -(\mathbf{b} - \mathbf{a})$$

2 Sentiment Classification

2.1 Different word embeddings

Log-Linear Model

Word Embedding	Adenoid (pos.)	aPhone (pos.)	Accuracy	MFC
glove.6B.50d.pkl	26.20%	24.03%	0.7419	0.6211
glove.6B.100d.pkl	29.66%	25.65%	0.7629	0.6211
glove.6B.200d.pkl	31.62%	28.71%	0.7764	0.6211
glove.6B.300d.pkl	32.26%	29.06%	0.7846	0.6211
GoogleNews-300.pkl	27.90%	24.55%	0.7689	0.6211
glove.twitter.27B.200d.pkl	5.97%	5.23%	0.6332	0.6211

Two Layers Neural Network

Word Embedding	Adenoid (pos.)	aPhone (pos.)	Accuracy	MFC
glove.6B.50d.pkl (dim1=2, dim2=2)	26.20%	24.03%	0.7419	0.6211
glove.6B.300d.pkl (dim1=3, dim2=3)	34.61%	31.41%	0.7877	0.6211
glove.6B.300d.pkl (dim1=4, dim2=3)	36.84%	32.97%	0.7880	0.6211

Note: Here the **dim1** and **dim2** represent the *number of columns of the weight matrixes* in **h1** and **h2** repectively.

Conclusion From the result got from different word embedding, we can clearly see the tranding of better performance (accuracy) as the size of dimension increase. This can possibly caused by the fact that increasing the dimension can decrease the bias in testing.

2.2 Comparing performances of one hot

Word Embedding	Adenoid (pos.)	aPhone (pos.)	Accuracy	MFC
Log-Linear one_hot	8.65%	7.91%	0.6827	0.6211
NN one_hot dim1:2, dim2:2	28.94%	24.48%	0.7785	0.6211
NN one_hot dim1:10, dim2:10	0.00%	0.00%	0.6211	0.6211

Note: Here the **dim1** and **dim2** represent the *number of columns of the weight matrixes* in **h1** and **h2** repectively.

Comparing with the best word embedding, one-hot doesn't have any advantage in accuracy, and the positive prediction in one-hot is significantly lower than most of the other word embeddings, except the glove.twitter.27B word embeddings.

2.3 Comparing Neural Network and Log-linear classifiers

The accuracy of these two classifier is similar when running on the same corpuses, but the Neural Network gives more positive prediction than the Log-linear model generally.

2.4 Best model report

After tuning the parameters and testing corpus, I choose to the model bellow as the best model:

Two layer neural network

- Corpus: glove.6B.300d.pkl
- Accuracy: **0.7890**
- Adenoid positive tweets: 32.32%
- aPhone positive tweets: 29.51%
- EPOCHS: 6
- Mini-batch size: 10
- One-hot vocab size: 10000
- Hidden size: 200

3 Conclusion

From the results we can find that even though the word embedding is a practical way to interpretate words in NLP, we cannot be too confident about the decisions made based on these word embeddings. Since the performance can vary significantly through different parameters and contexts.

But one good news is that the more data and more dimensions for each word we feed into the model (eight log-linear or neural-networds), the better outcomes we can expect.

4 References

1. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013a. URL https://arxiv.org/pdf/1301.3781.pdf. arXiv:1301.3781.

2. Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global vectors for word repre- sentation. In Proc. of EMNLP, 2014. URL http://www.aclweb.org/anthology/D14-1162.

3. Gensim Word2Vec Model URL https://radimrehurek.com/gensim/models/word2vec.html

4. About the Test Data by Matt Mahoney URL http://mattmahoney.net/dc/textdata.html

5. GloVe Project URL http://nlp.stanford.edu/projects/glove/