Report
In-Depth Word Vectors Analysis

Lyle He

Contents

1 Introduction

In this assignment I focused on a comprehensive understanding of word vector technologies, specifically Word2Vec and GloVe. I explored their applications, visualized the results, and analyzed the semantic and syntactic relationships they capture.

1.1 What is Word2Vec?

Word2Vec released by Google, a very popular model, is an ANN model used to learn the relationships between context words from a large corpus of text. It can transform words into vectors with a specified dimension (e.g. 100, 200, and 300).

It has two types: Continuous Bag of Words (CBOW) and Skip-Gram. CBOW predicts a target word from its context, while Skip-Gram does the inverse.

1.2 What is GloVec?

GloVe released by Stanford is a count-based model that uses matrix factorization on the word co-occurrence matrix of a corpus. The goal is to utilize statistical information by capturing global counts of word co-occurrences.

1.3 Common Test cases Design

To make the plot more readable, I picked some prepositions, conjunctions, verbs including past tense, adjectives including comparative, and some animal names.

Here are the words I chose:

- Semantically related
  - Countries: USA, Canada, Brazil, UK, France, Germany, Italy, Spain, Russia, China, India, Japan, Australia, Mexico, Egypt, South Africa, Nigeria, Kenya
  - Professions: Doctor, Engineer, Teacher, Lawyer, Nurse, Architect, Dentist, Pharmacist, Accountant, Psychologist, Veterinarian, Chef, Pilot, Journalist, Librarian, Electrician, Plumber, Carpenter
  - Aynonyms and Synonyms: Good, Bad, Happy, Sad, Smart, Stupid, Strong, Weak, Fast, Slow, Big, Small, Tall, Short, Old, Young, Rich, Poor, Beautiful, Ugly
- Syntactically related
  - Prepositions: in, on, at, by, for, with, of, to, from, through, between, among, under, over, above, below, near, far, around, about
  - Conjunctions: and, or, but, so, because, although, however, therefore, meanwhile, nevertheless, otherwise, instead
  - Verbs and Past Tense: run, ran, walk, walked, jump, jumped, eat, ate, drink, drank, sleep, slept, read, read, write, wrote, sing, sang, dance, danced

2 Building and Analyzing Word Vectors with Word2Vec

2.1 Building an own Word2Vec model

2.1.1 Corpus Selection

I chose a public domain book, Alice's Adventures in Wonderland by Lewis Carroll [1], for the model's corpus to train. There are a total of 38090 words including punctuation marks.

2.1.2 Model Training

And then trained two types of Word2Vec models, CBOW and Skip-Gram, hyperparameters are as follows:

- CBOW: window=5, min_count=1, vector_size=100
- Skip-Gram: window=5, min_count=1, vector_size=100

where window is the maximum distance between the current and predicted word within a sentence, min_count is the minimum count of words in the corpus, and vector_size is the dimensionality of the embeddings.

2.1.3 Visualization

I visualized the vectors of common test case using PCA, and the results are shown in the following figures. I picked some words frome Alice's Adventures in Wonderland to visualize the word vectors by PCA in 2D, since if I use my own words, the word vectors may not be included in the model.

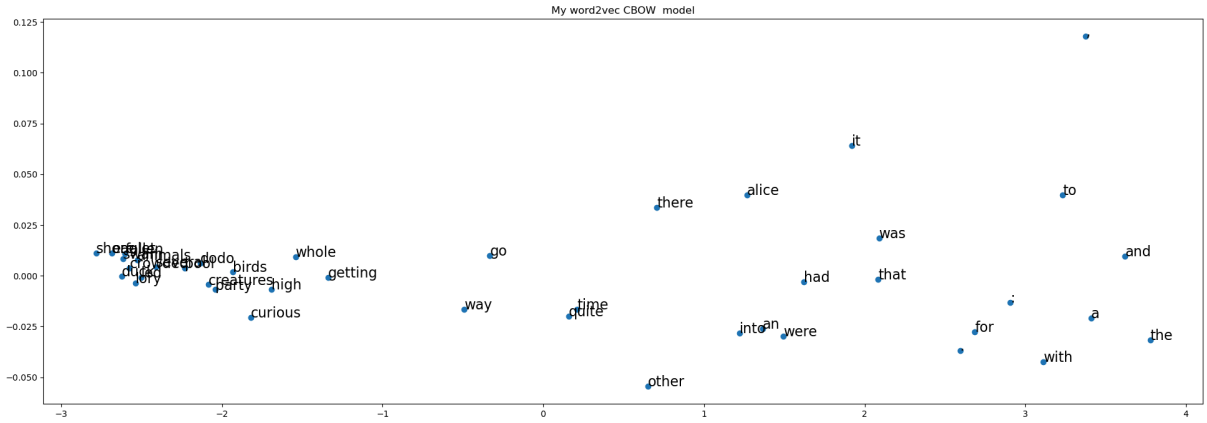

Figure 1: My word2vec CBOW model

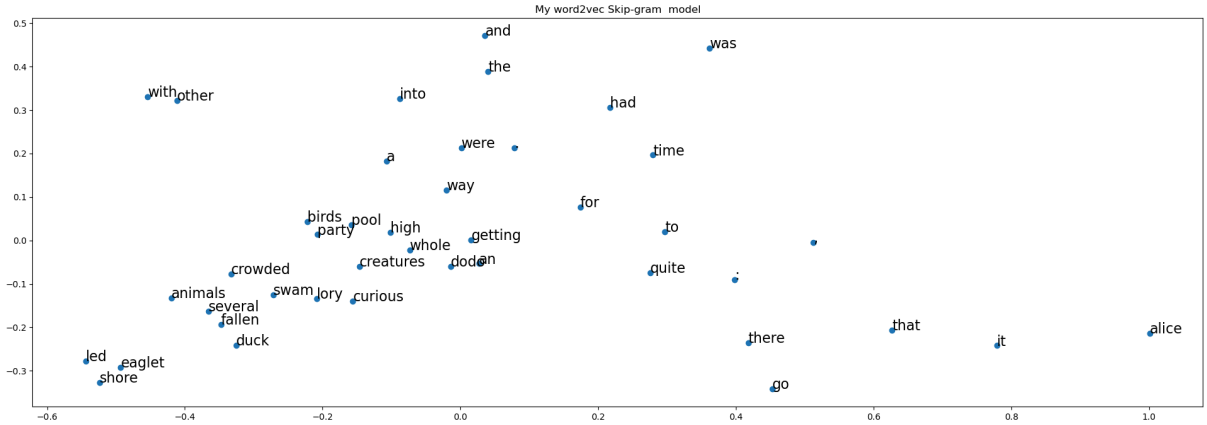

Figure 2: My word2vec Skip-gram model

2.1.4 Analysis

As we can see from figure 1 and figure 2, the distribution of the word vectors is different in the feild of semantic and syntactic.

In figure 1, the prepositions and conjunctions seems to be separated from the other words but they are not clustered together while other words are all clustered together.

In the figure 2, the obvious difference is that all the words are spread out in the 2D space, and some of the few words are clustered together a little bit.

The model is trained on a relatively small corpus. The plot results maybe means that in the small corpus, the CBOW model is better in the syntactic recognition ability than the Skip-gram model while the Skip-gram model is better in the semantic recognition ability than the CBOW model.

2.2 Use a pre-trained Word2Vec model

2.2.1 pre-trained word2vec model

I used the pre-trained word2vec model from the gensim library, which was trained on Google News data. The model has 300-dimensional vectors for 3 million words and phrases. The model was trained using the continuous bag of words (CBOW) method.

We can load the model (size is 1.64 GB.) using the following code:

```
import gensim.downloader as api
model = api.load('word2vec-google-news-300')
```

2.2.2 Visualization

I visualized the vectors of common test case using PCA in 2D, and the results are shown in the following figure 3 and figure 4. I visualized them in two categories, semantic and syntactic.

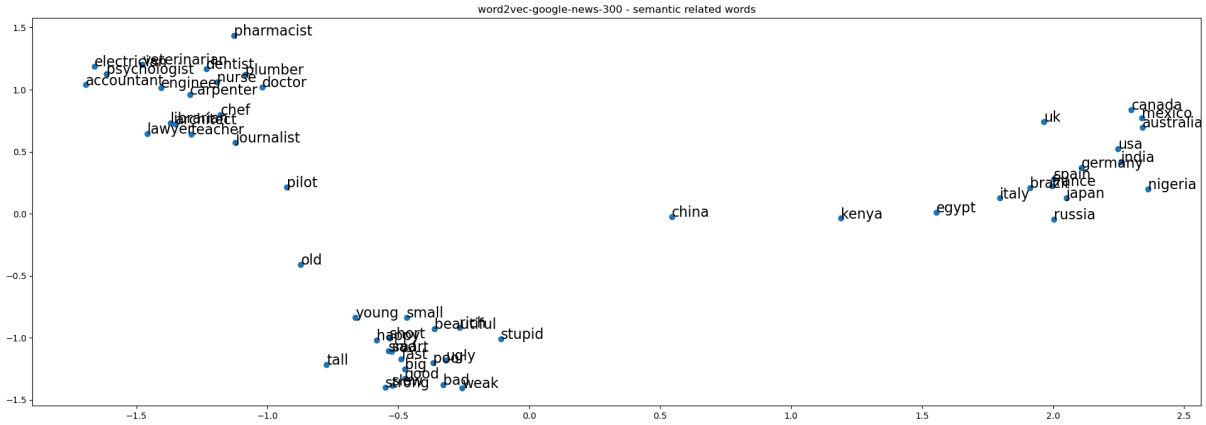

Figure 3: Word2Vec Visualization - semantic related words

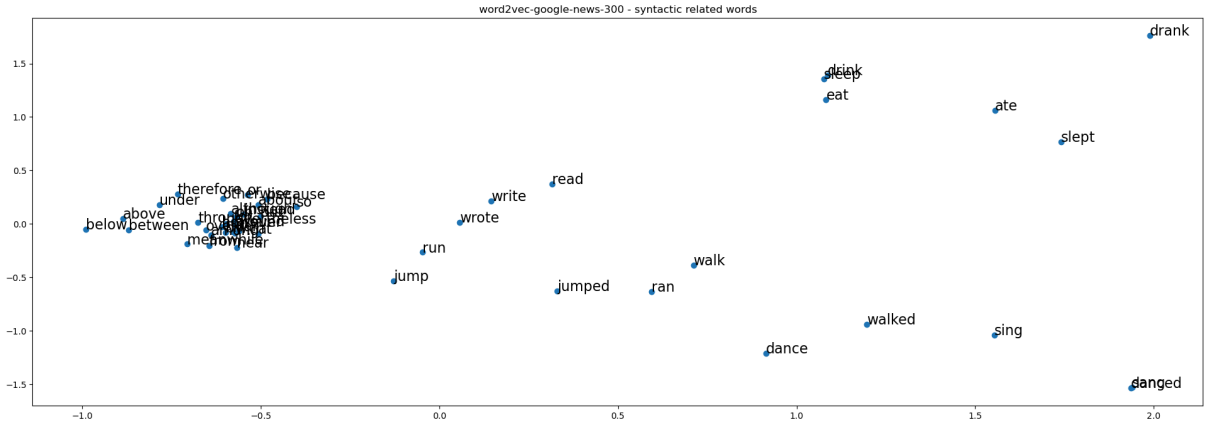

Figure 4: Word2Vec Visualization - syntactic related words

2.2.3 Analysis

As we can see, the result is very obvious, since different kinds of words cluster together respectively. It turns out that embedding words into 2D space using PCA can express the relationship between words. Word2vec model can capture the semantic and syntactic relationship between words.

In the figure 3, it is clear that different countries are clustered together, and especially some of the countries are more close to each other. The same situation happens in the professions since they are both noun words. As for the synonyms and antonyms, they are also clustered together.

In the figure 4, the prepositions and conjunctions are clustered together, some of conjunctions are separated out a little bit. The verbs and past tense are separated out from the other words and spread out in the right side of the plot.

In summary, the Word2Vec model can capture the semantic relationship between words well, but only can capture the syntactic relationship in some degree.

3 GloVe Vectors Advanced Analysis

3.1 Use a pre-trained GloVe model

3.1.1 pre-trained GloVe model

I used the pre-trained GloVe model from the gensim library, which was trained on Wikipedia 2014 + Gigaword 5 data. The model has 300-dimensional vectors for 400,000 words and phrases. The model was trained using the GloVe method.

We can load the model (size is 376 MB.) using the following code:

```
import gensim.downloader as api
model = api.load('glove-wiki-gigaword-300')

# A quick test
glove_vectors.most_similar('dog')
"""
[('dogs', 0.7888557314872742),
 ('cat', 0.6816746592521667),
 ('pet', 0.6291598081588745),
 ('puppy', 0.593606173992157),
 ('hound', 0.5468214750289917),
 ('horse', 0.5369751453399658),
 ('animal', 0.5316445827484131),
 ('cats', 0.5080744028091431),
 ('canine', 0.5038436055183411),
 ('pets', 0.5019966959953308)]
 """
```

3.1.2 Anlogy Task

As requested, I conducted an analogy task with example using the pre-trained GloVe model. I need to calculate the vector arithmetic, `king - man + woman`, and see if the result is similar to `queen`.

```
# calculate the vector of the analogy task
vector_after_calculation = glove_vectors["king"] - glove_vectors["man"] +
glove_vectors["woman"]

# find the most similar word to the vector_after_calculation
result = glove_vectors.similar_by_vector(vector_after_calculation)
print(result)

# The result is
"""
[('king', 0.8859834671020508), ('queen', 0.8609582185745239), ('daughter',
0.7684512734413147), ('prince', 0.7640699744224548), ('throne', 0.7634970545768738),
('princess', 0.7512729167938232), ('elizabeth', 0.7506489157676697), ('father',
0.7314497232437134), ('kingdom', 0.7296158671379089), ('mother', 0.728001058101654)]
"""
```

As we can see, except for the word “king”, the most similar word to the vector is “queen”, which is exactly the result we expected. Golve model is design to capture the relationship of word pairs that have some kind of relationships. This vector linear algebra calculation is a good example to show the ability of the GloVe model.

4 Semantic and Syntactic Word Relationships

4.1 Comprehensive comparison

4.1.1 Golve model Visualization supplement

I visualized the vectors of common test case using PCA through the pre-trained GloVe model, and the results are shown in the following figure 5 and figure 6. I visualized them in two categories, semantic and syntactic.

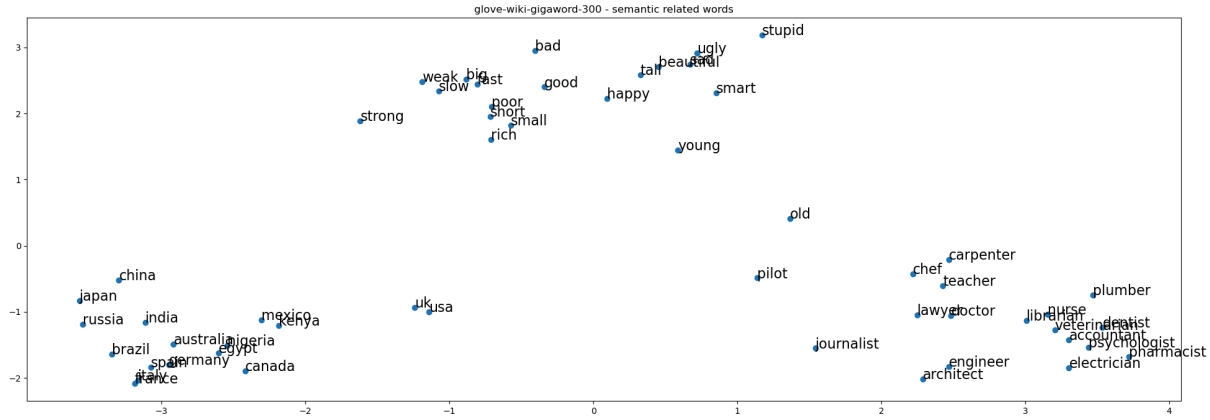

Figure 5: GloVe Vectors Visualization - semantic related words

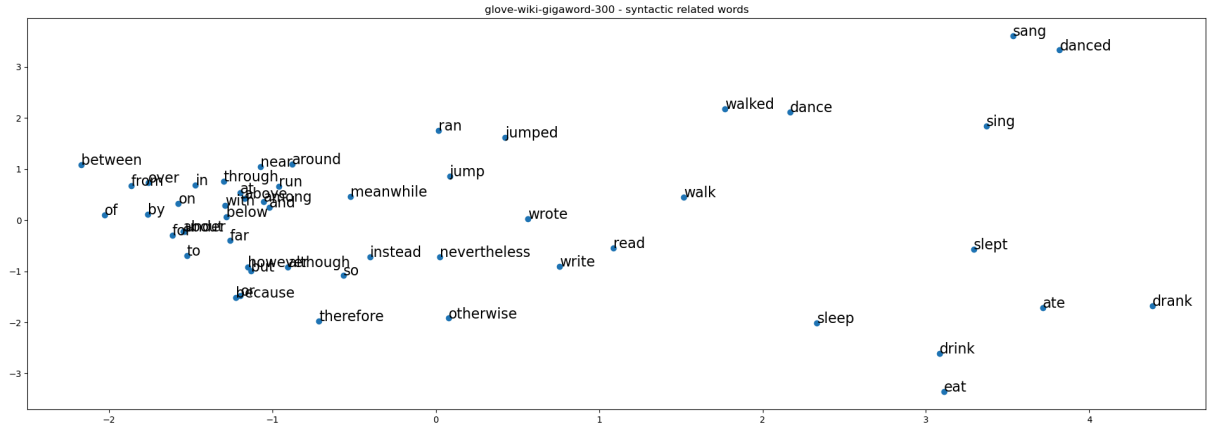

Figure 6: GloVe Vectors Visualization - syntactic related words

4.1.2 Word2Vec and GloVe comparison

As we can see from the figure 3 and figure 5, the GloVe model can capture the semantic relationship between words better than the Word2Vec model in some degree. Basically, the words are clustered together respectively by the two models, but the GloVe model can cluster the word pairs relationships more clearly. On the other hand, the Word2Vec model can separate different kinds of words better than the GloVe model. We can see the sepecific word pairs

In the figure 4 and figure 6, the situation is similar, the GloVe model can cluster the word pairs relationships more clearly than the Word2Vec model. The Word2Vec model can separate different kinds of words better than the GloVe model.

We can see the sepecific word pairs, jump and jumped, eat and ate, drink and drank, and so on. There are more clear linear relationships between the word pairs in the GloVe model. As for the prepositions and conjunctions, the separate with each other wider in the GloVe model.

4.2 calculate the similarity between word pairs

Here I chose some word pairs:

- Semantic relationship:
  - Happy - Sad, Good - Bad, Strong - Weak, Big - Small, Rich - Poor
- Syntactic relationship:
  - in - on, run - ran, walk - walked, but - so

The result is shown in the figure 7 below.

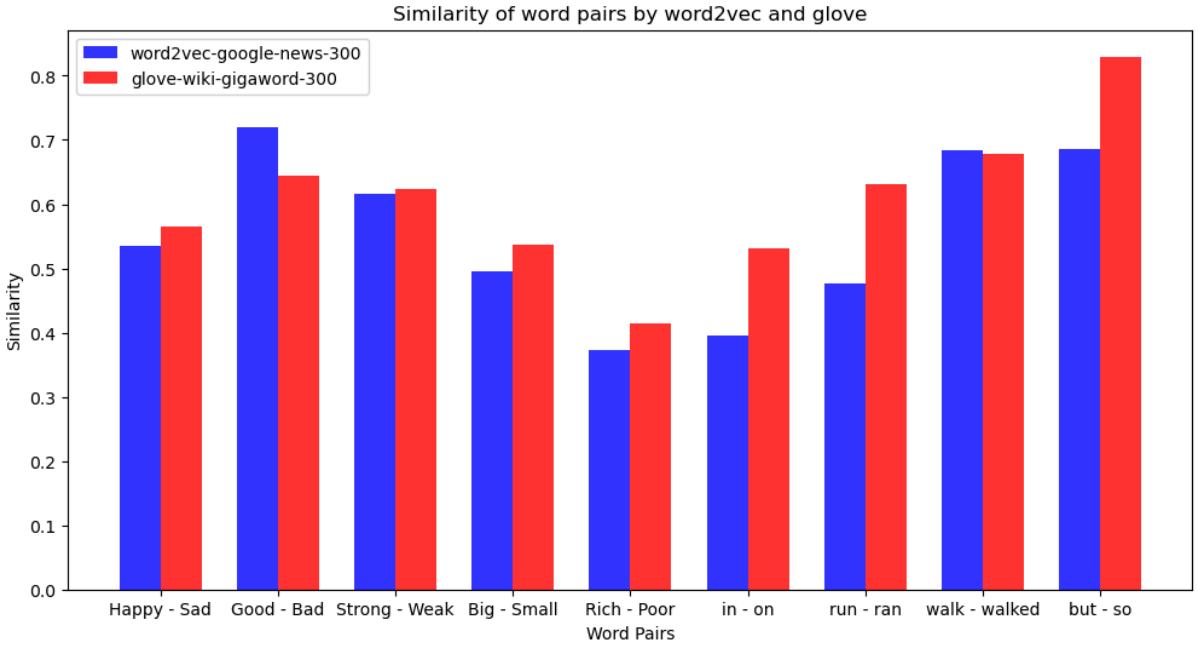

Figure 7: Word2Vec and GloVe comparison

As we can see, the similarity between word pairs in the GloVe model is higher than the Word2Vec model generally. Maybe the GloVe model can capture the relationship between word pairs better than the Word2Vec model.

5 Reference

- [1] https://www.gutenberg.org/files/11/11-0.txt
- [2] https://radimrehurek.com/gensim/models/word2vec.html
- [3] https://nlp.stanford.edu/projects/glove/
- [4] https://www.geeksforgeeks.org/python-word-embedding-using-word2vec/