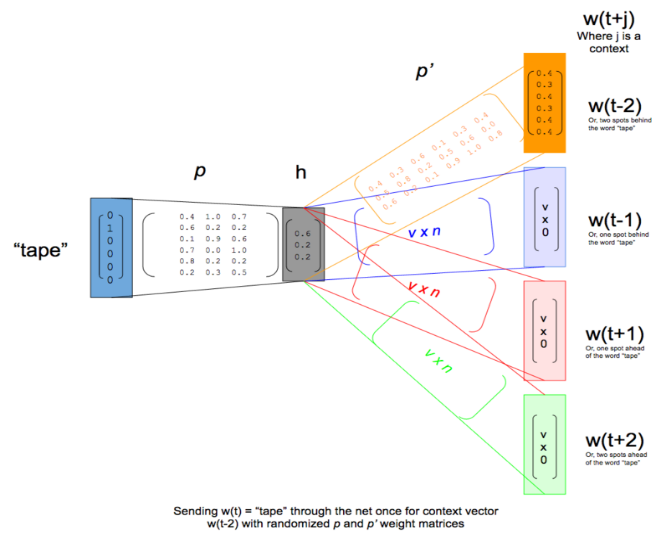
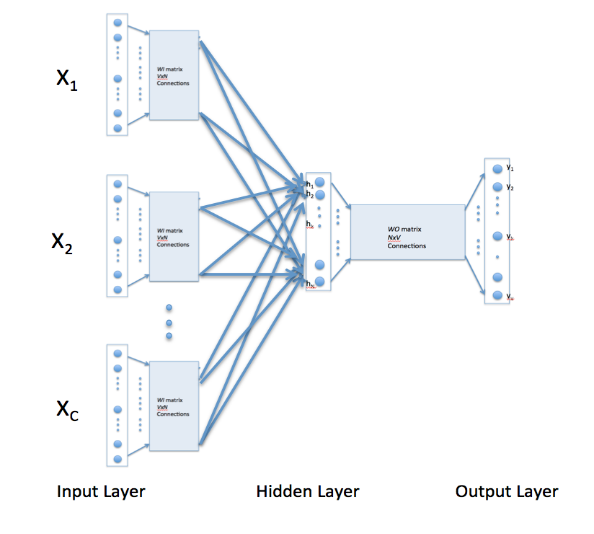
Spam Classification with GLOVE

We will discuss about different way to create word embedding’s because traditional Word2vec can utilize either of two model architectures to produce a [distributed representation](https://en.wikipedia.org/wiki/Distributed_representation) of words: [continuous bag-of-words](https://en.wikipedia.org/wiki/Continuous_bag-of-words) (CBOW) or continuous [skip-gram](https://en.wikipedia.org/wiki/Skip-gram). In the continuous bag-of-words architecture, the model predicts the current word from a window of surrounding context words. The order of context words does not influence prediction ([bag-of-words](https://en.wikipedia.org/wiki/Bag-of-words) assumption). In the continuous skip-gram architecture, the model uses the current word to predict the surrounding window of context words. The skip-gram architecture give more weight to the words which are more frequent in the context compared to the words which are less frequent. CBOW is faster while skip-gram is slower but does a better job for infrequent words.

Skip-Gram CBOW

Treadmill

Treadmill

Fig:1 Fig:2

The basic difference between CBOW and Skip-Gram is that CBOW is learning to predict the target word by the context and on the other hand Skip Gram is designed to predict the context. For better understanding I will try to explain it visually. Consider following example: -

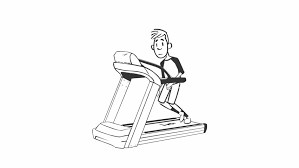


Fig: 3 Fig: 4 Fig: 5

Exapmple:-

CBOW: The boy is running on the \_\_\_\_\_. Fill in the blank, in this case, it’s “treadmill”.

Skip-gram: \_\_\_ \_\_\_ \_\_\_ treadmill. Complete the word’s context. In this case, it’s “The boy is running on the”

CBOW will learn the context as shown in fig 4 and then CBOW model will tell the most probably word is “Treadmill”,” Beach” or “Road”. The words like “Mountain” will get less attention of the model because it is designed to predict the most probable word.

Skip-gram will understand the word ‘Treadmill’ as shown in fig 5 and tell us, that there is huge probability, the context is ‘The boy is running on’, ’The girl is running on’ or some other relevant context.

But the approach used by word2vec is suboptimal since it doesn’t fully exploit statistical information regarding word co-occurrences. Jeffrey Pennington from Stanford have demonstrated a Global Vectors (GloVe) model which combines the benefits of the word2vec skip-gram model when it comes to word analogy tasks, with the benefits of matrix factorization methods that can exploit global statistical information. The GloVe model…

“GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.”

**Simple Co-occurrence Vectors**

Analyzing the context in which a word is used is a transcendental insight to attack this problem. Taking into account a word’s neighboring words is what has made NLP take a quantum leap in the most recent years.

Lets set a parameter ‘m’ which stands for the window size. In this example, we’ll use a size of 1 for educational purposes but 5–10 tends to be more common. This means that each word will be defined by its neighboring word to the left as well as the one to the right. This is modeled mathematically by constructing a co-occurrence matrix for each window. Let’s look at the following example:

I will try to explain with simple example of three sentence.

1. I love statistics.
2. I love programming.
3. I need to learn NLP.

Here the word ‘love’ is defined by the words ‘I’ and ‘Programming’, meaning that we increment the value both for the ‘I love’ and the ‘love Programming’ co-occurrence. We do that for each window and obtain the following co-occurrence matrix:

**Co-Occurrence Matrix**

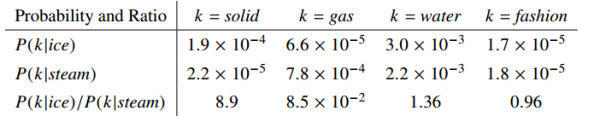
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | I | Love | Statistics | Programming | Need | To | Learn | NLP | . |
| I | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Love | 2 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| Statistics | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Programming | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Need | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| To | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| Learn | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| NLP | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| . | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |

Once we have the co-occurrence matrix filled we can plot its results into a multi-dimensional space. Since ‘Programming’ and ‘Statistics’ share the same co-occurrence values, they would be placed in the same place; meaning that in this context they mean the same thing (or ‘pretty much’ the same thing). ‘NLP would be the closest word to these 2 meaning ‘it has the closest possible meaning but it’s not the same thing’, and so on for every word.

The statistics of word occurrences in a corpus is the primary source of information available to all unsupervised methods for learning word representations, and although many such methods now exist, the question remains as to how meaning is generated from these statistics, and how the resulting word vectors might represent that meaning. And The relationship of these words can be examined by studying the ratio of their co-occurrence probabilities with various probe words, k.

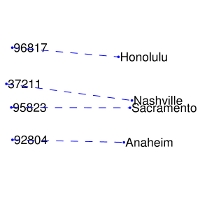
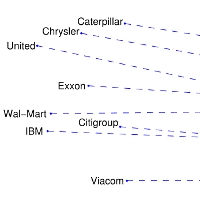
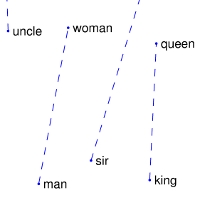
Let P(k|w) be the probability that the word k appears in the context of word w. Consider a word strongly related to ice, but not to steam, such as solid. P(solid | ice) will be relatively high, and P(solid | steam) will be relatively low. Thus the **ratio** of P(solid | ice) / P(solid | steam) will be large. If we take a word such as gas that is related to steam but not to ice, the ratio of P(gas | ice) / P(gas | steam) will instead be small. For a word related to both ice and steam, such as water we expect the ratio to be close to one. We would also expect a ratio close to one for words related to neither ice nor steam, such as fashion.

The following table shows that this does indeed pan out in practice:



The above argument suggests that the appropriate starting point for word vector learning should be with ratios of co-occurrence probabilities rather than the probabilities themselves…. Since vector spaces are inherently linear structures, the most natural way to encode the information present in a ratio in the word vector space is with vector differences because this simplicity of linearity can be problematic since two given words almost always exhibit more intricate relationships than can be captured by a single number. For example, *man* may be regarded as similar to *woman* becuase both words describe human beings; on the other hand, the two words are often considered opposites since they highlight a primary axis along which humans differ from one another.

So, to capture in a quantitative way, the nuance necessary to distinguish *man* from *woman*, it is necessary for a model to associate more than a single number to the word pair. A natural and simple candidate for an enlarged set of discriminative numbers is the vector difference between the two word vectors. GloVe is designed in order that such vector differences capture as much as possible the meaning specified by the juxtaposition of two words as shown in below.



Man-Women company-ceo zip-city

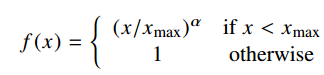
The training objective of GloVe is to learn word vectors such that their dot product equals the logarithm of the words’ probability of co-occurrence. Owing to the fact that the logarithm of a ratio equals the difference of logarithms, this objective associates (the logarithm of) ratios of co-occurrence probabilities with vector differences in the word vector space. Because these ratios can encode some form of meaning, this information gets encoded as vector differences as well. For this reason, the resulting word vectors perform very well on word analogy tasks, such as those examined in the word2vec package.

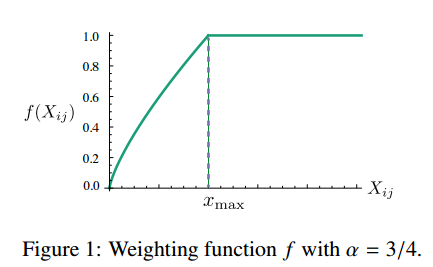
Now since our ratios are all the things we care about, so the cost function gives us the model –



Where  is ratio of cooccurrence of words and  is dot product between words.

To deal with co-occurrences that happen rarely or never – which are noisy and carry less information than the more frequent ones – the authors use a weighted least squares regression model. One class of weighting functions found to work well can be parameterized as



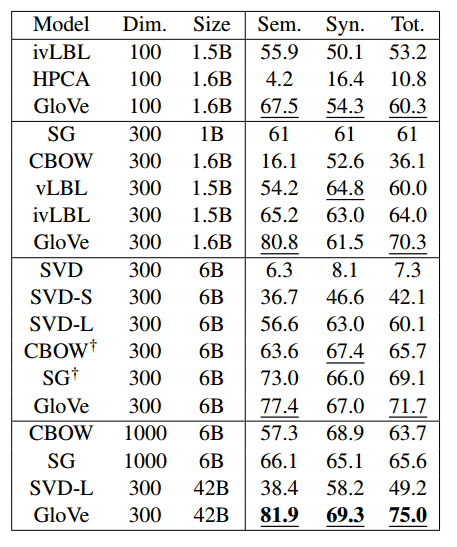


### Glove Results

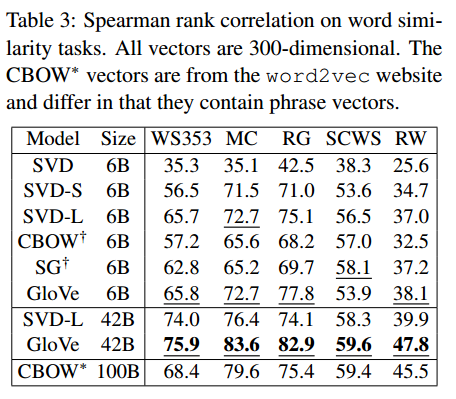
The model was trained on five corpora including a 2010 Wikipedia dump with 1 billion tokens and a 2014 Wikipedia dump with 1.6 billion tokens, Gigaword 5 with 4.3 billion tokens, a combination of Gigaword 5 and the 2014 Wikipedia dump totalling 6 billion tokens, and finally 42 billion tokens of web data from Common Crawl.

This pass can be computationally expensive, but it’s a one time up-front cost. GloVe does very well on the word analogy task, achieving class-leading combined accuracy of 75%. It also gets great results on word similarity and named entity recognition tests.

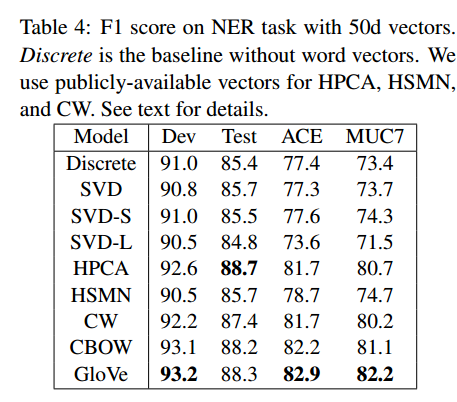
Word analogy task results:



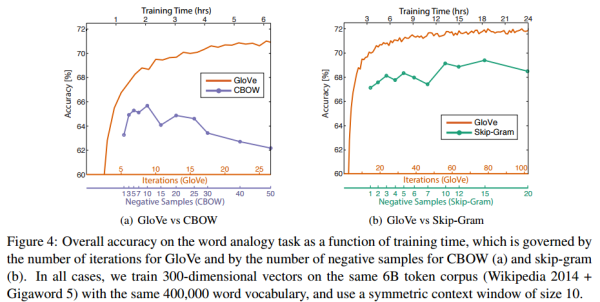
Word similarity:



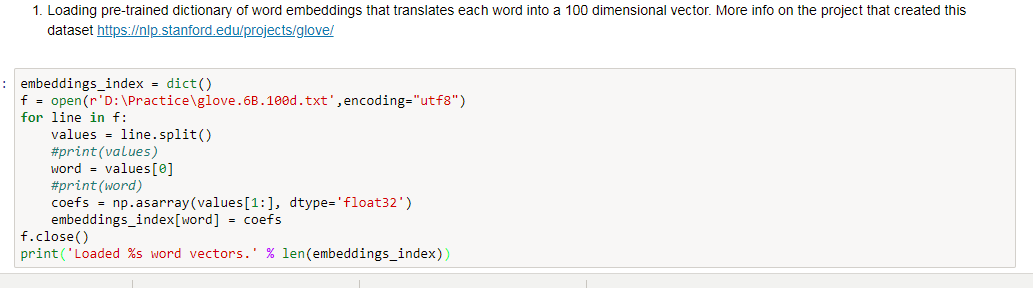
Named Entity Recognition:

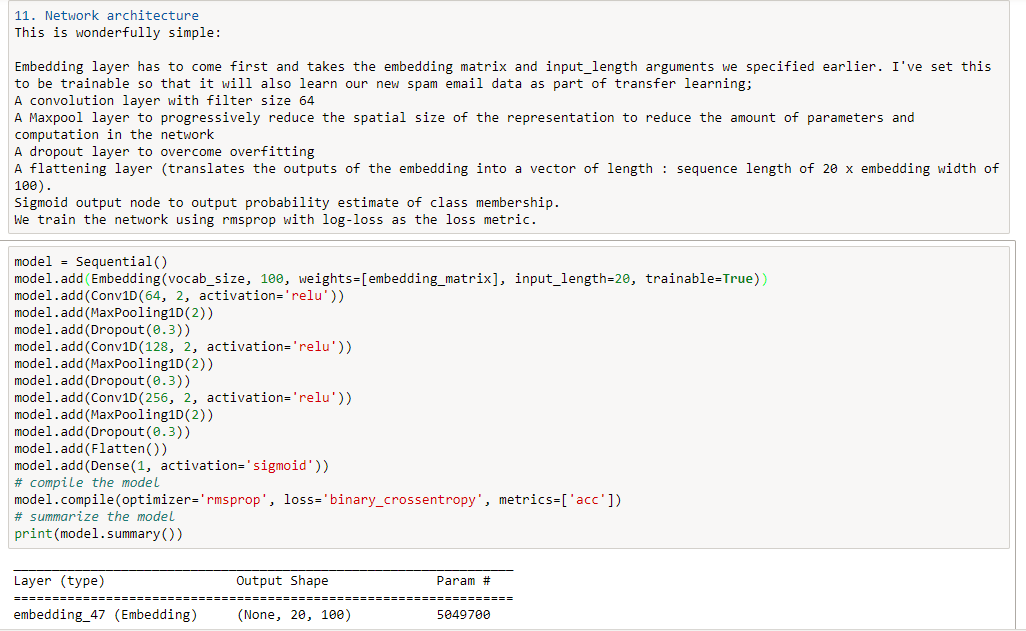


Compared to word2vec, for the same amount of training time GloVe shows improved accuracy on the word analogy task.

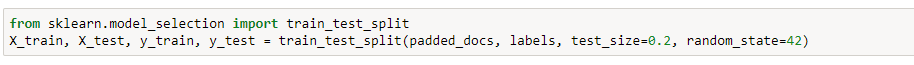


We have trained the neural network on spam classification data with pre-trained glove instead of using word2vec Skip-gram.

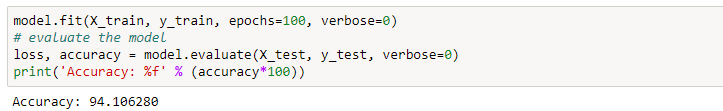




Dividing the data into train and test.



Evaluating the model



Complete code can be found on Github: