Word2Vec Resources

27 Apr 2016

While researching Word2Vec, I came across a lot of different resources of varying usefullness, so I thought I'd share my collection of links and notes on what they contain.

Original Papers & Resources from Google Team

Word2Vec was presented in two initial papers released within a month of each other. The original authors are a team of researchers from Google.

Efficient Estimation of Word Representations in Vector Space

Link to paper

This was the first paper, dated September 7th, 2013.

This paper introduces the Continuous Bag of Words (CBOW) and Skip-Gram models. However, don't expect a particularly thorough description of these models in this paper…

I believe the reason for this is that these two new models are presented more as modifications to previously existing models for learning word vectors. Some of the terminology and concepts in this Word2Vec paper come from these past papers and are not redifined in Google's paper.

A good example are the labels "projection layer" and "hidden layer" which come from the "NNLM" model. The term "projection layer" is used to refer to a middle layer of the neural network *with no activation function*, whereas "hidden layer" implies a non-linear activation.

Distributed Representations of Words and Phrases and their Compositionality

Link to paper

This was a follow-up paper, dated October 16th, 2013.

This paper adds a few more innovations which address the high compute cost of training the skip-gram model on a large dataset. These added tweaks are fundamental to the word2vec algorithm, and are implemented in Google's C version as well as the Python implementation in `gensim`.

These innovations are:

1. Subsampling common words (that is, eliminating some training samples).
2. "Negative Sampling" - A modification of the optimization objective which causes each training sample to update only a small percentage of the model's weights.

Additionally, they point out the value in recognizing common "phrases" and treating them as single words in the model (e.g., "United_States" or "New_York").

Presentation on Word2Vec

Link to presentation

This was presented December 9th, 2013 at NIPS 2013 by Tomas Mikolov from Google.

I think this is mainly a re-hash of the content in the two papers. Seeing it presented differently may help you pull out some additional insights, though.

C Code Implementation

Link to code

The above link is to the home page for google's own Word2Vec implementation in C.

You can also find here some pre-trained models that they have provided. Note that it's possible to load these pre-trained models into `gensim` if you want to work with them in Python.

Tutorials

Alex Minnaar's Tutorials

The best tutorials I found online were done by Alex Minnaar.

He's since taken the tutorials down, but I have PDF copies here:

- Part I - The Skip-Gram Model
- Part II - Continuous Bag-of-Words Model

Kaggle Word2Vec Tutorial

Link to tutorial

This is pretty cool. It's a Kaggle competition that's really just a Python tutorial to teach you about using Word2Vec with `gensim` . It's well written and will walk you through all of the steps carefully. It does very little to explain the algorithms used, but is great on the practical implementation side.

It uses a sentiment analysis task (on the IMDB movie review dataset) as an example project. While the tutorial is great for showing you how to get set up with `gensim` and even train your own Word2Vec model on the data, you'll discover that it essentially fails at applying Word2Vec effectively on the example task of sentiment analysis! To get good results on the IMDB dataset, you'll want to check out Google's Doc2Vec technique (which isn't covered in this tutorial).

Here's what the tutorial covers.

Part 1:

- Cleaning and tokening the text data.
- Vectorizing the documents using word counts.
- Classification using a random forest.

Part 2:

- Setting up `gensim`
- Training a Word2Vec model (learning word vectors from the dataset) using `gensim`

Part 3:

- This section attempts two rather unsuccessful ways of applying the word vectors to create vector representations of each review. Neither manages to outperform the simpler word-count approach from part 1.
  - Creating vector representations of each movie review by averaging its word vectors.
  - Clustering the word vectors to identify sets of synonyms, then using the word-count approach, but this time combining synonyms into a single bucket.

Part 4:

- Points to Google's Doc2Vec as a superior solution to this task, but doesn't provide implementation details.

Folgert Karsdorp's Word2Vec Tutorial

Link to tutorial

I haven't read this tutorial in depth… It covers the Continuous Bag of Words model (instead of the Skip-Gram model). It even includes some of the backprop equations.

Discussions on Quora

- https://www.quora.com/What-are-the-continuous-bag-of-words-and-skip-gram-architectures-in-laymans-terms
- https://www.quora.com/How-does-word2vec-work
- https://www.quora.com/What-are-some-interesting-Word2Vec-results/answer/Omer-Levy

Implementations

The below implementations also include some tutorials; I haven't gone through them in detail yet.

- Word2Vec and Doc2Vec in Python in gensim here and here
- Word2vec in Java in deeplearning4j
- Java version from Medallia
- Word2vec implementation in Spark MLlib
- Word2Vec implementation / tutorial in Google's TensorFlow

My Own Stuff

- I have my own tutorial on the skip-gram model of Word2Vec here.
- Part 2 of my tutorial covers subsampling of frequent words and the Negative Sampling technique.
- I created a project called inspec_word2vec that uses gensim in Python to load up Google's large pre-trained model, and inspect some of the details of the vocabulary.
- I'm working on a Matlab implementation of Word2Vec, word2vec_matlab. My goal is less about practical useage and more about understanding the model. For now, it doesn't support the most important part–actually training a Word2Vec model. What it does do currently is allow you to play with a paired-down (or, really, cleaned-up!) version of Google's pre-trained model in Matlab.

16 Comments **mccormickml.com** 1 **Login**

♡ **Recommend** 4 **Share** **Sort by Best**

Join the discussion…

Tony Zhao • a month ago

I have some questions about word vector in distributed representation, the word vector is a kind of looks like this [0.792, -0.177, -0.107, 0.109, -0.542, ...]. ①Is these values are generated randomly at first ? ②Why they are all in the interval (-1,1) ? Why there have negative values ? ③What dose every value mean ? Hope you can tell me that, thanks !

Reply • Share ›

Chris McCormick Mod ➦ Tony Zhao • a month ago

1. Yes, the values are initialized in a random fashion, though with an equation that takes some of the network properties into account.
2. The weights can be positive or negative because they can either contribute positively or negatively to the output.
3. The individual values have no intuitive interpretation that I know of. Taken as a whole, though, words whose vectors are close to one another have similar meanings.

Take a look at my tutorial, it could give you some more intuition about how these values are generated.

^ | ⌄ • Reply • Share ›

Tony Zhao ➦ Chris McCormick • a month ago

Thank you very much, after reading your tutorial, I have a more clear understanding of Word2Vec.
Dose the tanh() is the equation that make values in interval (-1,1)? I saw the tanh()
appearing in a statistical language model.

^ | ⌄ • Reply • Share ›

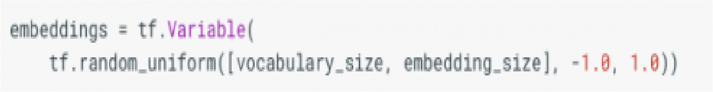
Chris McCormick Mod ➦ Tony Zhao • 18 days ago

tanh does have that shape, yes. But for Softmax, the activation function is:
1 / (1 + exp(-x))
Check this explanation out for more details

^ | ⌄ • Reply • Share ›

Tony Zhao ➦ Chris McCormick • 12 days ago

Hello Chris, I just read https://www.tensorflow.org/..., and I found the code that control the initial value in interval (-1,1).



```
embeddings = tf.Variable(
    tf.random_uniform([vocabulary_size, embedding_size], -1.0, 1.0))
```

It was generated from uniform distribution in (-1,1).

^ | ⌄ • Reply • Share ›

Tony Zhao ➦ Chris McCormick • 12 days ago

Thanks a lot, Chris !

^ | ⌄ • Reply • Share ›

Jing • 3 months ago

Thank you very much. It does help me a lot!

Thank you very much. It does help me a lot!

Reply • Share ›

Chris McCormick Mod → Jing • 3 months ago

Great, thanks!

Reply • Share ›

Labiba Jahan • 6 months ago

The document is really helpful for clear understanding. Thank you so much.

Reply • Share ›

Chris McCormick Mod → Labiba Jahan • 6 months ago

Awesome, glad it helped!

Reply • Share ›

Sam Clarke • 6 months ago

Hi Chris,
A super useful compilation of resources, thank you! Your skip-gram tutorial was also brilliant, the most clear non-mathematical explanation I have read about how it works.
Just thought I'd let you know that the final equation in the pdf that you supply of Alex Minnaar's skip-gram tutorial has a couple of typos in it, which caused me a bit of grief!
The equation

$$w_{ij}^{(new)} = w_{ij}^{(old)} - \eta \cdot \sum_{j=1}^{V} \sum_{c=1}^{C} (y_{c,j} - t_{c,j}) \cdot w'_{ij} \cdot x_j$$

should actually be

$$w_{ki}^{(new)} = w_{ki}^{(old)} - \eta \cdot \sum_{j=1}^{V} \sum_{c=1}^{C} (y_{c,j} - t_{c,j}) \cdot w'_{ij} \cdot x_k$$

if I'm not mistaken.
Also Xin Rong has written a note which explains well, and with mathematical detail, skip-gram and CBOW training as well as the hierarchical softmax and negative sampling techniques to speed up training. http://www-personal.umich.e...
Cheers,
Sam.

Reply • Share ›

Chris McCormick Mod ➦ Sam Clarke • 6 months ago − | ⚑

Cool, thank you for sharing that!

^ | v • Reply • Share ›

Sanghyeon Cho • 9 months ago − | ⚑

Very good Document! Thank you very much.

^ | v • Reply • Share ›

Chris McCormick Mod ➦ Sanghyeon Cho • 9 months ago − | ⚑

Thanks, glad it helped!

^ | v • Reply • Share ›

Vaibhav Aggarwal • 9 months ago − | ⚑

Hi
I have got a copy of Alex Minnaar's word2vec tutorial if you want to link it here.

^ | v • Reply • Share ›

Chris McCormick Mod ➦ Vaibhav Aggarwal • 9 months ago − | ⚑

Thank you. I had copies as well, but I wanted to ask Alex about it first. I haven't heard from him, though, so I went ahead and added links to my saved copies.

1 ^ | v • Reply • Share ›

ALSO ON **MCCORMICKML.COM**

SVM Tutorial - Part I

6 comments • a year ago•

Ruben Peralta — :(I really wanted to

The Gaussian Kernel

1 comment • 10 months ago•

Aharon Azulay — Thank you!

Related posts

Concept Search on Wikipedia 22 Feb 2017

Getting Started with mlpack 01 Feb 2017

Word2Vec Tutorial Part 2 - Negative Sampling 11 Jan 2017