

word2vec Example Code

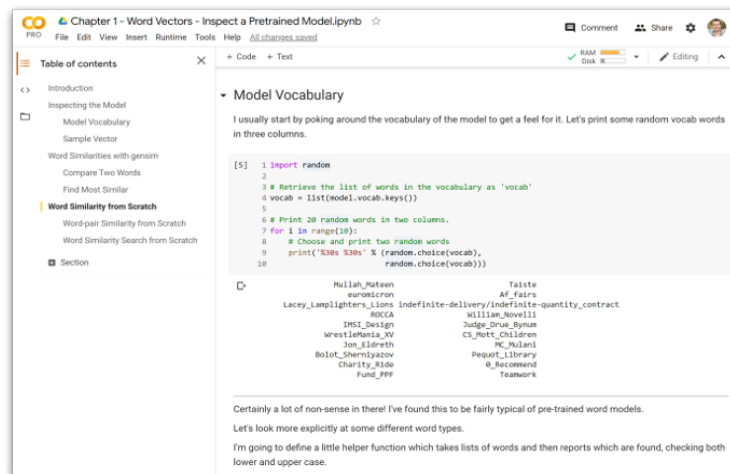
Click the [blue Notebook titles](#) to open each Notebook!

What's Colab?

Google Colab is a free service which allows you to run Jupyter Notebooks on hosted instances. It's a fantastic learning tool, in part because it eliminates any issues with setting up your own Python environment! I have a beginners Colab tutorial [here](#).

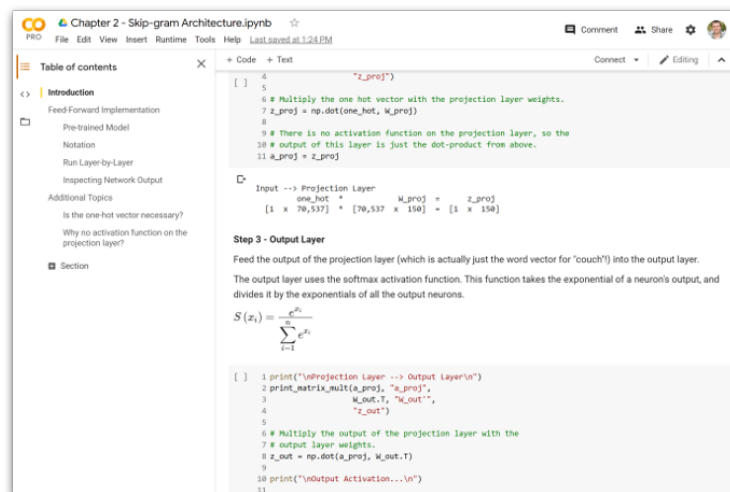
Chapter 1 - Word Vectors - Inspect a Pretrained Model.ipynb

<https://colab.research.google.com/drive/1HJpbEjh-0etoUbTjbQt36k930XY0o3UT>



Chapter 2 - Skip-gram Architecture.ipynb

https://colab.research.google.com/drive/12niolspRalsLJE6Tfee8FlZLdQ-_1_tL



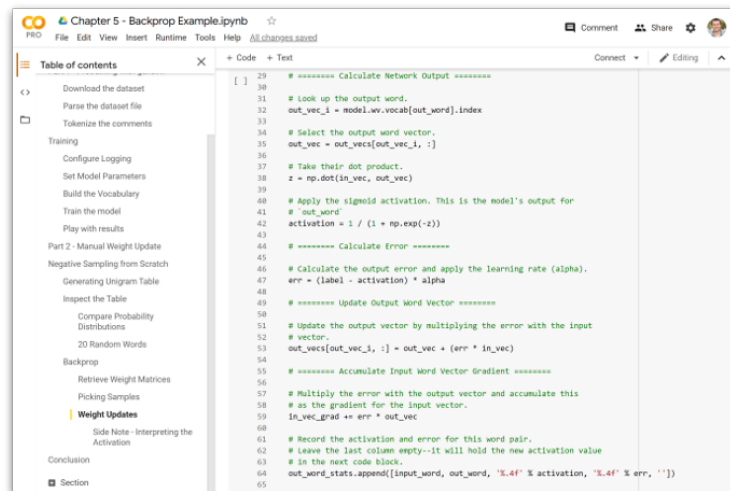
Chapter 3 - Negative Sampling.ipynb

https://colab.research.google.com/drive/1jjV9sTmC_9oY2tDLzi0-mduLeYzWun7U



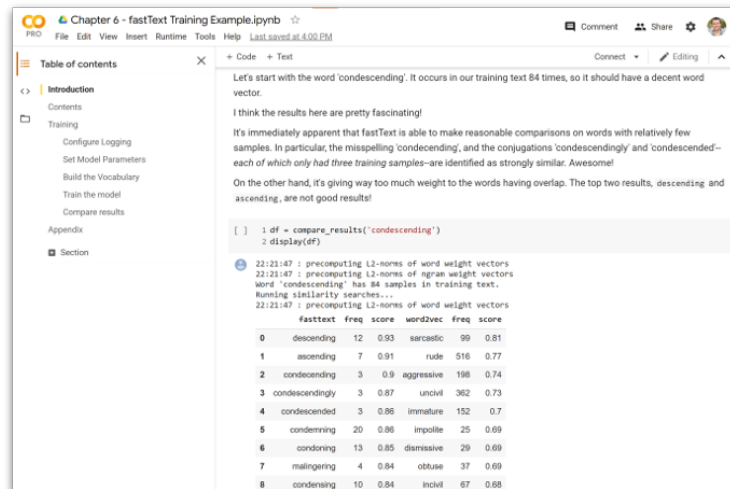
Chapter 5 - Backprop Example.ipynb

<https://colab.research.google.com/drive/1uBVAUNejTSA167WgViesUpDoTjvqimjp>



Chapter 6 - fastText Training Example.ipynb

https://colab.research.google.com/drive/1Uyjm1_PO4YqdoSr9BQ0t2ONPiF-TXuU-



Let's start with the word 'condescending'. It occurs in our training text 84 times, so it should have a decent word vector.

I think the results here are pretty fascinating!

It's immediately apparent that fastText is able to make reasonable comparisons on words with relatively few samples. In particular, the misspelling 'condescending', and the conjugations 'condescendingly' and 'condescended'—each of which *only had three training samples*—are identified as strongly similar. *Awesome!*

On the other hand, it's giving way too much weight to the words having overlap. The top two results, descending and ascending, are not good results!

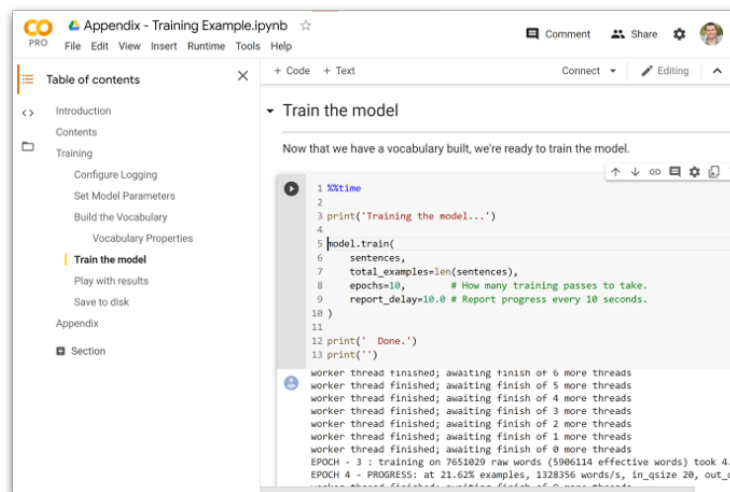
```
[ ] 1 df = compare_results('condescending')
    2 display(df)
```

```
22:21:47 : precomputing L2-norms of word weight vectors
22:21:47 : precomputing L2-norms of ngram weight vectors
Word 'condescending' has 84 samples in training text.
Running similarity searches...
22:21:47 : precomputing L2-norms of word weight vectors
```

	fasttext	freq	score	word2vec	freq	score
0	descending	12	0.93	sarcastic	99	0.81
1	ascending	7	0.91	rude	516	0.77
2	condescending	3	0.9	aggressive	198	0.74
3	condescendingly	3	0.87	uncivil	362	0.73
4	condescended	3	0.86	immature	152	0.7
5	condemning	20	0.86	impolite	25	0.69
6	condoning	13	0.85	dismissive	29	0.69
7	malingering	4	0.84	obtuse	37	0.69
8	condensing	10	0.84	incivil	67	0.68

Appendix - Full word2vec Training Example.ipynb

<https://colab.research.google.com/drive/1E6DuXc0G7R4B6fSGel1ye42zZ-pBeNXkN>



Now that we have a vocabulary built, we're ready to train the model.

```
1 %time
2
3 print('Training the model...')
4
5 model.train(
6     sentences,
7     total_examples=len(sentences),
8     epochs=10, # How many training passes to take.
9     report_delay=10.0 # Report progress every 10 seconds.
10 )
11
12 print(' Done.')
13 print('')
```

worker thread finished; awaiting finish of 6 more threads
worker thread finished; awaiting finish of 5 more threads
worker thread finished; awaiting finish of 4 more threads
worker thread finished; awaiting finish of 3 more threads
worker thread finished; awaiting finish of 2 more threads
worker thread finished; awaiting finish of 1 more threads
worker thread finished; awaiting finish of 0 more threads
EPOCH - 3 : training on 7651029 raw words (5986110 effective words) took 4.
EPOCH 4 - PROGRESS: at 21.62% examples, 1328356 words/s, in_qsize 20, out_q

Appendix - Wiki Attack Comments.ipynb

<https://colab.research.google.com/drive/1x4PSNxno9QrQHILXLbD1k5p7tCQa1lG2>