Google's trained Word2Vec model in Python

12 Apr 2016

In this post I'm going to describe how to get Google's *pre-trained* Word2Vec model up and running in Python to play with.

As an interface to word2vec, I decided to go with a Python package called gensim. gensim appears to be a popular NLP package, and has some nice documentation and tutorials, including for word2vec.

You can download Google's pre-trained model here. It's 1.5GB! It includes word vectors for a vocabulary of 3 million words and phrases that they trained on roughly 100 billion words from a Google News dataset. The vector length is 300 features.

Loading this model using gensim is a piece of cake; you just need to pass in the path to the model file (update the path in the code below to wherever you've placed the file).

```
import gensim

# Load Google's pre-trained Word2Vec model.
model = gensim.models.Word2Vec.load_word2vec_format('./model/
```

However, if you're running 32-bit Python (like I was) you're going to get a memory error!

This is because gensim allocates a big matrix to hold all of the word vectors, and if you do the math…

```
3 million words * 300 features * 4bytes/feature = ~3.35GB
```

…that's a big matrix!

Assuming you've got a 64-bit machine and a decent amount of RAM (I've got 16GB; maybe you could get away with 8GB?), your best bet is to switch to 64-bit Python. I had a little trouble with this–see my notes down at the end of the post.

Inspecting the Model

I have a small Python project on GitHub called inspect_word2vec that loads Google's model, and inspects a few different properties of it.

If you'd like to browse the 3M word list in Google's pre-trained model, you can just look at the text files in the vocabulary folder of that project. I split the word list across 50 files, and each text file contains 100,000 entries from the model. I split it up like this so your editor wouldn't completely choke (hopefully) when you try to open them. The words are stored in their original order–I haven't sorted the list alphabetically. I don't know what determined the original order.

Here are some the questions I had about the vocabulary, which I answered in this project:

- Does it include stop words?
  - Answer: Some stop words like "a", "and", "of" are *excluded*, but others like "the", "also", "should" are *included*.
- Does it include misspellings of words?
  - Answer: Yes. For instance, it includes both "mispelled" and "misspelled"–the latter is the correct one.
- Does it include commonly paired words?
  - Answer: Yes. For instance, it includes "Soviet_Union" and "New_York".

- Does it include numbers?
  - Answer: Not directly; e.g., you won't find "100". But it does include entries like "###MHz_DDR2_SDRAM" where I'm assuming the '#' are intended to match any digit.

Here's a selection of 30 "terms" from the vocabulary. Pretty weird stuff in there!

```
Al_Qods
Surendra_Pal
Leaflet
guitar_harmonica
Yeoval
Suhardi
VoATM
Streaming_Coverage
Vawda
Lisa_Vanderpump
Nevern
Saleema
Saleemi
rbracken@centredaily.com
yellow_wagtails
P_&C;
CHICOPEE_Mass._WWLP
Gardiners_Rd
Nevers
Stocks_Advance_Paced
IIT_alumnus
```

```
Popery
Kapumpa
fashionably_rumpled
WDTV_Live
ARTICLES_##V_##W
Yerga
Weegs
Paris_IPN_Euronext
##bFM_Audio_Simon
```

64-bit Python on Windows

It took me some effort get a 64-bit Python setup with gensim up and running, so I thought I'd share my steps.

I had been using Python(x, y) to get a nice machine learning-oriented Python environment up and running. However, there doesn't appear to be a 64-bit release of Python(x, y) yet…

I found a package called WinPython that does include 64-bit support. It looks to be actively supported, and includes all of the features I cared about from Python(x, y) (it includes the Spyder IDE and scikit-learn with all its dependencies).

Head over to the homepage for WinPython here. I initially tried WinPython for Python 3.5, but ran into some issues, and ended up just going with Python

2.7, which worked fine!

You may already have this, but for Python 2.7 you will need the Microsoft Visual C++ 2008 Redistributable Package (x64).

I'm using the following version: WinPython-64bit-2.7.10.3

You can extract WinPython wherever you want; I put mine right under C:.

WinPython doesn't put itself in the Windows registry or on the system path; however, it does include some batch scripts for doing this. Look under `C:\WinPython-64bit-3.5.1.2\scripts\` and you'll find `env.bat` and `register_python.bat`.

Open a Windows command prompt and run those two batch scripts. Then, in the same command window, you can install gensim easily by executing the following on the command line: `easy_install -U gensim`

That should do it! With any luck, you should now be able to run the Python code at the top of the post to import Google's model.

Related posts

Product Quantizers for k-NN Tutorial Part 2 22 Oct 2017
Product Quantizers for k-NN Tutorial Part 1 13 Oct 2017
k-NN Benchmarks Part I - Wikipedia 08 Sep 2017