Café, bonne nuit

Have a coffee when you run any code, if it is still running, then have two….

PYTHON

TensorFlow 07: Word Embeddings (2) – Loading Pre-trained Vectors

Posted on January 17, 2017

A brief introduction on Word2vec please check this post. In this post, we try to load pre-trained Word2vec model, which is a huge file contains all the word vectors trained on huge corpora.

Download

Download here .I downloaded the GloVe one, the vocabulary size is 4 million, dimension is 50. It is a smaller one trained on a "global" corpus (from wikipedia). There are models trained on Twitter as well in the page.

Wikipedia+Gigaword 5	50	Wikipedia+Gigaword 5 (6B)	400,000	GloVe	GloVe	AdaGrad

The model is formatted as (word vector) in each line, separated by a space. Below shows a screenshot: not only the words, but also some marks like comma are included in the model.

```
dhcp-892bf8eb:W2Vs ireneli$ head -3 glove.6B.50d.txt
the 0.418 0.24968 -0.41242 0.1217 0.34527 -0.044457 -0.49688 -0.17
0095095 0.011658 0.10204 -0.12792 -0.8443 -0.12181 -0.016801 -0.33
125 -0.19526 4.0071 -0.18594 -0.52287 -0.31681 0.00059213 0.007444
758 -0.045637 -0.44251 0.18785 0.0027849 -0.18411 -0.11514 -0.7858
, 0.013441 0.23682 -0.16899 0.40951 0.63812 0.47709 -0.42852 -0.55
91 0.090201 -0.13324 0.078639 -0.41634 -0.15428 0.10068 0.48891 0.
.28351 3.5416 -0.11956 -0.014533 -0.1499 0.21864 -0.33412 -0.13872
.22786 0.36034 -0.37818 -0.56657 0.044691 0.30392
```

Loading

There is an easy way for you to load the model by reading the vector file. Here I separate the words and vectors, because the words will be fed into vocabulary.

```
1  import numpy as np
2  filename = 'glove.6B.50d.txt'
3  def loadGloVe(filename):
4      vocab = []
5      embd = []
6      file = open(filename,'r')
7      for line in file.readline
8          row = line.strip().sp
9          vocab.append(row[0])
10         embd.append(row[1:])
```


Gallery

```
    print('Loaded GloVe!')
    file.close()
    return vocab,embd
vocab,embd = loadGloVe(filena
vocab_size = len(vocab)
embedding_dim = len(embd[0])
embedding = np.asarray(embd)
```

The vocab is a list of words or marks. The *embedding* is the huge 2-d array with all the word vectors. We initialize the embedding size to be the number of column of the *embedding* array.

Embedding Layer

After loading in the vectors, we need to use them to initialize W of the embedding layer in your network.

```
W = tf.Variable(tf.constant(0.
                trainable=Fals
embedding_placeholder = tf.pla
embedding_init = W.assign(embe
```

Here W is first built as Variables, but initialized by constant zeros. Be careful with the shape: [*vocab_size*, *embedding_dim*], where we can know after loading the model. If *trainable* is set to be False, it would not be updated during training. Change to True for a trainable setup. Then an *embedding_placeholder* is set up to receive the real values (fed from

Posts

January 2017

M T W T F S S

1

2 3 4 5 6 7 8

9 10 11 12 13 14 15

16 17 18 19 20 21 22

23 24 25 26 27 28 29

30 31

« Dec Mar »

the *feed_dict* in *sess.run()*), and at last *W* is assigned.

After creating a session and initialize global variables, run the *embedding_init* operation by feeding in the 2-D array embedding.

```
1 sess.run(embedding_init, feed_
```

Vocabulary

Suppose you have raw documents, the first thing you need to do is to build a vocabulary, which will map each word into an id. TensorFlow process the following code to lookup embeddings:

```
1 tf.nn.embedding_lookup(W, inpu
```

where *W* is the huge embedding matrix, *input_x* is a tensor with ids. In another word, it will lookup embeddings by given Ids.

So we would choose the pre-trained model when we build the vocabulary: word-id maps.

```
1 from tensorflow.contrib import
2 #init vocab processor
3 vocab_processor = learn.prepro
4 #fit the vocab from glove
5 pretrain = vocab_processor.fit
6 #transform inputs
```

```
7 x = np.array(list(vocab_proces
```

First init the *vocab processor* by passing in a *max_document_length*, in default, shorter sentences would be padded by zeros. Then we fit the processor by the *vocab* list to build the word-id maps. Finally, use the processor to transform from real raw documents.

Now you are ready to train your own network with pre-trained word vectors!

Related

NLP 05: From Word2vec to Doc2vec: a simple example with Gensim
In "Algorithm"

Deep Learning 16: Understanding Capsule Nets
In "Deep Learning"

TensorFlow 06: Word Embeddings (1)
In "Theory"

Author: Irene

Keep calm and update blog.

VIEW ALL POSTS

Previous Post

Deep Learning 14 : Optimization, an Overview

Next Post

ML Recap Slides sharing

14 thoughts on “TensorFlow 07: Word Embeddings (2) – Loading Pre-trained Vectors”

djargon says:

January 28, 2017 at 12:24 pm

can you provide the full source code of this tutorial please?

★ Like

Reply

Irene says:

January 29, 2017 at 12:25 am

Oh, sorry they are related to my current research. As I am still trying to improve, so I hope would upload to Github soon 😀

★ Like

Reply

Shilei Cao says:

February 4, 2017 at 8:05 am

I think u don't see the source code of VocabularyProcessor seriously. The id of the vocabulary is initialized with an 'UNK', whose id is 0. u can see it from https://github.com/tensorflow/tensorflow/blob/master/tensorflow/contrib/learn/python/learn/preprocessing/categorical_vocabulary.py . so the id ur code return will be 1 bigger than the correct one.

★ Like

Reply

Irene says:

February 16, 2017 at 1:56 pm

Hi Shilei, thanks for the reply. But I did check my model, by using "list(model.vocab.keys())", so it

gives you a list of words that are in your vocabulary. Then for each word, you could apply "model[word]" to get access the word embeddings.

Thanks.

★ Like

Reply

Irene says:

February 16, 2017 at 1:58 pm

It seems that the gensim lib excluding the "UNK" thing (in my previous reply). I will check the TF one, thanks for your reminder 😀

★ Like

Reply

Anonymous says:

June 28, 2017 at 2:33 pm

I agree you , I think he can add vocab[0] = [0]*emb_size

★ Liked by 1 person

Reply

Simpson Family says:

August 23, 2017 at 2:20 pm

After applying ” pretrain = vocab_processor.fit(vocab) ”
The number of vocabulary in vocab_processor compare with vocab
len(vocab_processor.vocabulary_) == 2762098
len(vocab) == 3000000

Is there any similar situation happened? Thank you

★ Liked by 1 person

Reply

Irene says:
January 3, 2018 at 7:00 pm
There is a reply:
“Simpson Family
I think that the problem is the vocab_processor doesn’t collect the quotation mark and I am still thinking about how to deal with this issue, because this issue may cause the mistake when using the method embedding_lookup…
”

I did not have the same issue, did you have it solved?

Thanks.

★ Like

Reply

ahmad says:

August 26, 2017 at 11:36 am

just a small detail: you haven't included the following line in your code :

> import numpy as np

and i think it might be confusing to people who haven't directly worked with np before. although this i unlikely to cause problems as ppl who are working in the field of deep learning have probably worked with numpy too, but just wanted to mention it. maybe it will save the day for someone 😀

★ Like

Reply

Irene says:

January 3, 2018 at 7:02 pm

Hi ahmad, thanks for the suggestion. I just updated 😀

Thanks.

★ Like

Reply

Anonymous says:

September 13, 2017 at 8:47 am

Simpson Family

I think that the problem is the vocab_processor doesn't collect the quotation mark and I am still thinking about how to deal with this issue, because this issue may cause the mistake when using the method embedding_lookup…

★ Liked by 1 person

Reply

Irene says:

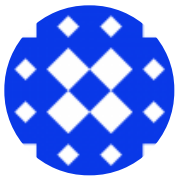
January 3, 2018 at 6:59 pm

I am not sure if this is the reason, did you find out how to solve it?

Thanks.

★ Like

Reply



foo5302 says:

October 14, 2017 at 1:15 pm

Hi Irene,

Thanks for writing this post. I found it really helpful.

Could you please elaborate on the form of raw input in the last line ie:

x = np.array(list(vocab_processor.transform(your_raw_input)))

I tried:

x = np.array(list(vocab_processor.transform(['hello world'])))

And got this result:

array([[12846, 78, 0, 0, 0, 0, 0, 0, 0, 0]], dtype=int64)

But vocab[12846] gives 'carriage' and vocab[78] gives 'government'.

I thought vocab[12846] should give 'hello' and vocab[78] should give 'world'…

Note: I set max_document_length to be 10

★ Like

Reply

Irene says:

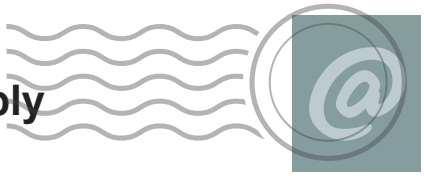
January 3, 2018 at 7:06 pm

Hi there! Sorry I did not work in this area for a very long time. I hope you have solved it?

Thanks.

★ Like

Reply



Leave a Reply

Enter your comment here...