doc2vec - How to infer vectors of documents faster?

Asked 5 years, 3 months ago Active 1 year, 7 months ago Viewed 2k times

5

1

I have trained paragraph vectors for around 2300 paragraphs(between 2000-12000 words each) each with vector size of 300. Now, I need to infer paragraph vectors of around 100,000 sentences which I have considered as paragraphs(each sentence is around 10-30 words each corresponding to the earlier 2300 paragraphs already trained).

So, am using

`model.infer_vector(sentence)`

But, the problem is it is taking too long, and it does not take any arguments such as "`workers`" .! Is there a way I can speed up the process by threading or some other way? I am using a machine with 8gb ram and when I checked the available cores using

```
cores = multiprocessing.cpu_count()
```

it comes out to be 8.

I need this for answering multiple choice questions. Also, are there any other libraries/models such as `doc2vec` which can help in this task?

Thanks in advance for your time.

python gensim word2vec doc2vec

Share Follow

edited Dec 20 '17 at 11:56

asked Sep 19 '16 at 18:57
Dreams
4,913 ●7 ●39 ●62

2 Answers

Active Oldest Votes

4

You might get a slight speedup from calling `infer_vector()` from multiple threads, on different subsets of the new data on which you need to infer vectors. There would still be quite a bit of thread-contention, preventing full use of all cores, due to the Python Global Interpreter Lock ('GIL').

If your RAM is large enough to do so without swapping, you could save the model to disk, then load it into 8 separate processes, and have each do inference on 1/8th of the new data. That'd be the best way to saturate all CPUs.

Any faster speedups would require making optimizations to the `infer_vector()` implementation in gensim – which is an open issue on the project that would accept contributed improvements.

Share Follow

answered Sep 27 '16 at 4:19
gojomo
45.1k ●12 ●79 ●102

You may use multiprocessing:

2

```python
from multiprocessing import Pool
from gensim.models import Doc2Vec

MODEL = Doc2Vec.load('my_doc2vec_model', mmap='r')
MODEL.delete_temporary_training_data(keep_doctags_vectors=False,
keep_inference=True)

def infer_vector_worker(document):
    vector = MODEL.infer_vector(document)
    return vector

documents = [
    ['now', 'is', 'the', 'time'],       # first document
    ['for', 'all', 'good', 'men'],      # second document
    ['to', 'come', 'to', 'the'],        # third document
    ['aid', 'of', 'their', 'country'],  # fourth document
]
with Pool(processes=4) as pool:
    vectors = pool.map(infer_vector_worker, documents)
```

Since `MODEL` is a global variable, it will be shared copy-on-write between the processes. Therefore, there will be no extra memory consumption beyond what the inference consumes.

Share Follow

answered May 31 '20 at 17:48
Witiko
2,747 ● 3 ● 23 ● 40