How to load large dataset to gensim word2vec model

Asked 1 year, 4 months ago Active 1 year, 4 months ago Viewed 1k times

5

So I have multiple text files(around 40). and each file has around 2000 articles (average of 500 words each). And each document is a single line in the text file.

So because of the memory limitations I wanted to use dynamic loading of these text files for training. (Perhaps a iterator class?)

so how do I proceed?

- train each text file -> save the model -> load the model and rerun on new data?
- is there a way with iterator class to do this automatically?
- should I give sentence by sentence, article by article or text file by text file as input to model training?

python iterator gensim word2vec

Share Follow

asked Aug 17 '20 at 22:55

little JJ

61 • 3

1 Answer

Active | Oldest | Votes

2

A corpus of `40 text files * 2000 articles * 500 words each` equals about 40000000 words in total, which is still pretty small for this kind of work. I'd guess that's under 400MB, uncompressed, on disk. Even if 4x that in RAM, a lot of desktop or cloud machines could easily handle that 1-2GB of text as Python objects, as a list-of-lists-of-string-tokens. So, you may still have the freedom, depending on your system, to work all-in-memory.

But if you don't, that's OK, because the gensim `Word2Vec` & related classes can easily take all training data from any *iterable sequence* that provides each item in turn, and such *iterables* can in fact read text line-by-line from one, or many, files – each time the data is needed.

Most gensim intro `Word2Vec` tutorials will demonstrate this, with example code (or the use of library utilities) to read from one file, or many.

For example, gensim's included `LineSentence` class can be instantiated with the path to a single text file, where each line is one text/sentence, and single spaces separate each word. The resulting object is a Python iterable sequence, which can be iterated over to get those lists-of-words as many times as needed. (Behind the scenes, it's opening & stream-reading the file each time, so no more than the current text need ever be in RAM as a time.)

An early gensim `Word2Vec` tutorial – https://rare-technologies.com/word2vec-tutorial/ – shows a short `MySentences` Python class that does the same over all files in a single directory:

```
class MySentences(object):
    def __init__(self, dirname):
        self.dirname = dirname

    def __iter__(self):
        for fname in os.listdir(self.dirname):
            for line in open(os.path.join(self.dirname, fname)):
```

```
                yield line.split()

sentences = MySentences('/some/directory') # a memory-friendly iterable
model = gensim.models.Word2Vec(sentences)
```

For `Word2Vec`, it doesn't really matter if you provide the text sentence-by-sentence, or paragraph-by-paragraph, or article-by-article. It's the smaller windows of nearby words that drive the results, not the 'chunks' you choose to pass to the algorithm. So, do whatever is easiest. (But, avoid chunks of more than 10000 words at a time in gensim versions through the current gensim-3.8.3 release, as an internal limit will discard words past the 10000 mark for each text.)

However, **don't** do all training on one batch yourself, then do all training on another batch, etc. Combining all the data into one iterable is best. Then, all examples are consulted for initial vocabulary-discovery, and all examples are trained together, over the automatic multiple training passes – which is best for model convergence. (You **don't** want all of the early training to be among one set of examples, then all the late training to a different set of examples, as that would imbalance the examples' relative influences, and prevent the model from considering the full variety of training data in each optimization pass.)

Share Follow

answered Aug 17 '20 at 23:15

gojomo
45.1k ● 12 ● 79 ● 102

If I understand you correctly, the code example you provided above would feed the model line by line, thus selecting the line as the window of nearby words that is used to train the vectors for the words in the line, correct? My main concern is that, if the window of nearby words is defined otherwise, the words from the last line of one file may be considered nearby to the words in the first line of the following file that is read, which of course wouldn't make sense. – GDwag Mar 22 '21 at 11:45

1 The `Word2Vec` code/algorithm only considers neighboring words in the same text example as being in each others' contexts. Here, there's one example per line, which as one item in the sequence is a list-of-tokens. The last word in any line won't affect the next line in the same file - much less the first line in the next file. – gojomo Mar 22 '21 at 17:35

1 Further: **even if there was such overlap**, if adjacent texts aren't related, such bleed-over would just add a bit of unnecessary noise to training, & be unlikely to affect final results much, especially if texts are dozens or hundreds of words long. That is, the true signal or real relationships would drown out that small bit of extra noise. You could test this, if you have doubts: go ahead & explicitly add a few junk words from prior/next texts to each item that is `yield`ed, & evaluate results. You'll likely see slightly longer training (more words in total) but hardly any diff in quality. – gojomo Mar 22 '21 at 17:41