Project: spaCy

2015-12-01T10:49:10Z RXminuS 23comments

Feature Request: Vector "File" interface

Web url: https://github.com/explosion/spaCy/issues/197

API url: https://api.github.com/repos/explosion/spaCy/issues/197

Just read in an old spaCy tutorial the following "Future versions of spaCy will allow you to provide a file-like object, instead of a location of a [vector bin] file."

Is this in place yet? Would love to replace standard vector file and in-memory loading with my own Redis (or any other "shared-memory-system") interface to allow a distributed cluster of spacy nodes to share the same "file". Would love to contribute, any pointers on where to start looking?

-------------------------------------------------------------------------

2015-12-03T16:35:02Z honnibal

Yep, this should work: https://github.com/honnibal/spaCy/blob/master/spacy/vocab.pyx#L320

Thanks for mentioning this ‚Äî I'll keep this open until we update the docs.

-------------------------------------------------------------------------

2015-12-04T01:19:38Z RXminuS

That's almost what I'm looking for but exactly like expected so can be closed once updated in docs, thx.

However I was more looking for a way to have spaCy avoid loading in the vectors in it's own memory and allow for an interface so that I can make this centralised in a shared memory solution. So pre-cook a "database" with vector lookups and each spaCy instance just calls class functions like find() and nearest() which can either be implemented as a "hashmap" (like it's currently) or a shared memory source. This makes spaCy much more useable for including in our docker environment where we literally have 100s of these containers running in parallel and memory is wasted for each instance.

I'll hopefully have some spare time soon and will write a little pull request now that I know where to look :-)

-------------------------------------------------------------------------

2015-12-04T12:17:53Z honnibal

This makes sense.

I'd like to change the current set up, because I want to support vectors keyed by different information, e.g. vectors keyed by lemma and part-of-speech. This lets you see different vectors for `take/VERB` and `take/NOUN`. Digital Reasoning wrote a paper showing this got them good results, and early examination of the vectors is looking good to me too.

-------------------------------------------------------------------------

2015-12-04T19:58:47Z RXminuS

That's actually a really good idea! Could you link the papers here and I'll have a look at that as well. (thinking I might actually do this as part of my master thesis :P )

In a way this information is already contained in word vectors because two verbs will be seen in more similar contexts than adjectives but guess that by reducing ambiguity and "false positives" it could make quite a difference.

Do you have some form of basic class design for it yet which I should stick to or shall I just come up with something?

-------------------------------------------------------------------------

2015-12-04T21:10:46Z honnibal

Here's the paper I mentioned, titled "sense2vec": http://arxiv.org/pdf/1511.06388.pdf , by @iamtrask

> In a way this information is already contained in word vectors because two verbs will be seen in more similar contexts than adjectives but guess that by reducing ambiguity and "false positives" it could make quite a difference.

Not so! The most similar words to `take/NOUN` might be things like `opinion|NOUN`, while the most similar words to `take/VERB` might be `give|VERB` or `put|VERB`. In normal Word2Vec these two share a key, so there's no way to look at the two different "senses" separately.

I've been playing with an extension of this idea, where noun chunks and named entities are also merged. I've trained a model on one month of Reddit comments. The results at the moment are quite messy, and many of the phrases need to be pruned from the vocab. But there are also some interesting results in there too. Example:

```

>>> model = gensim.models.Word2Vec.load('np\_ner\_tag\_reddit\_2015\_01-300d.model')

>>> model.most\_similar(['take|NOUN'])

[(u'personal\_stance|NOUN', 0.49355706572532654), (u'best\_guess|NOUN', 0.48488736152648926), (u'personal\_favorite\_song|NOUN', 0.4821690320968628), (u'stance|NOUN', 0.48033151030540466), (u'favorite\_track|NOUN', 0.47577959299087524), (u'personal\_opinion|NOUN', 0.47202107310295105), (u'favorite\_community|NOUN', 0.4700315594673157), (u'personal\_view|NOUN', 0.4675809144973755), (u'overall\_opinion|NOUN', 0.4594742953777313), (u'first\_impression|NOUN', 0.458611398935318)]

```

The vectors for the verb and noun senses are quite different:

```

>>> model.similarity('take|NOUN', 'take|VERB')

0.13804844694430313

```

The nearest neighbour of `take|VERB` turns out to be a misspelling, that the POS tagger seems to often tag correctly:

```

>>> model.similarity('take|VERB', 'tale|VERB')

0.49611725495669901

```

I'll be writing more about these vectors, and of course releasing the code. I'd like to sharpen up one or two things and run it on more data first.

I can give you some code to get you started on the POS tagged vectors, though.

As much as I dislike dumping state to disk, it's the most practical way to do this. It'd be nice to have the multi-threading sorted out for spaCy, but for now multi-processing is okay, especially for the tagger, which is fast and low-memory. Example:

https://github.com/honnibal/spaCy/blob/master/examples/pos\_tag.py

Here's how to train Word2Vec on the output using Gensim. (@piskvorky):

``` python

from \_\_future\_\_ import print\_function, unicode\_literals, division

import io

import bz2

import logging

from os import path

import os

import plac

import ujson

from gensim.models import Word2Vec

class Corpus(object):

def \_\_init\_\_(self, directory):

self.directory = directory

def \_\_iter\_\_(self):

for filename in os.listdir(self.directory):

text\_loc = path.join(self.directory, filename)

with io.open(text\_loc, 'r', encoding='utf8') as file\_:

for sent\_str in file\_:

yield sent\_str.split()

@plac.annotations(

in\_dir=("Location of input directory"),

out\_loc=("Location of output file"),

n\_workers=("Number of workers", "option", "n", int),

size=("Dimension of the word vectors", "option", "d", int),

window=("Context window size", "option", "w", int),

min\_count=("Min count", "option", "m", int)

)

def main(in\_dir, out\_loc, n\_workers=4, window=5, size=128, min\_count=10):

logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.INFO)

corpus = Corpus(in\_dir)

model = Word2Vec(

Corpus(in\_dir),

size=size,

window=window,

min\_count=min\_count,

workers=n\_workers

)

model.save(out\_loc)

if \_\_name\_\_ == '\_\_main\_\_':

plac.call(main)

```

-------------------------------------------------------------------------

2015-12-04T21:23:10Z iamtrask

That chunking experiment is fantastic. Love the take -> personal\_stance.

-------------------------------------------------------------------------

2015-12-04T21:27:49Z honnibal

> Do you have some form of basic class design for it yet which I should stick to or shall I just come up with something?

I thought the answer was "no", but then I started writing out some "suggestions", and I guess I have a clearer idea than I thought :). Most of these things are demanded by consistency with the rest of the library. You can see examples of pretty much all of this in the `Vocab` and `StringsTable` classes:

Python API:

- Use the `\_\_getitem\_\_`, `\_\_setitem\_\_` and `\_\_iter\_\_` special methods. You don't necessarily have to subclass dict.

- All vectors in the same `VectorTable` must be the same length.

- The table should allow the user to pass in a key function, which should take a `spacy.tokens.Token` object as an argument, and return a 64-bit unsigned integer (used to key the table)

- The hash will be non-reversible, and it won't be possible to iterate over the keys and get back a useful representation of the integer key. This is okay. Otherwise we'll have to store the key strings, which could occupy a lot of memory.

Implementation details:

- The data should be stored in a PreshMap instance.

- The table should be keyed by a `feat\_t` (64 bit unisnged integer), with values being `weight\_t\*`, i.e. raw C arrays of floats.

- Allocate the memory using `cymem.Pool`

-------------------------------------------------------------------------

2015-12-04T22:28:01Z honnibal

A few bonus queries on the chunked model.

1) The vector space seems like it'll give a good way to show compositionality:

"fair game" is not a type of game:

```

>>> model.similarity('fair\_game|NOUN', 'game|NOUN')

0.034977455677555599

>>> model.similarity('multiplayer\_game|NOUN', 'game|NOUN')

0.54464530644393849

```

A "class action" is only very weakly a type of action:

```

>>> model.similarity('class\_action|NOUN', 'action|NOUN')

0.14957825452335169

```

But a class action \_lawsuit\_ is definitely a type of lawsuit:

```

>>> model.similarity('class\_action\_lawsuit|NOUN', 'lawsuit|NOUN')

0.69595765453644187

```

2) Similarity between entities can be kind of fun.

Here's what Reddit thinks of Donald Trump:

```

>>> model.most\_similar(['Donald\_Trump|PERSON'])

[(u'Sarah\_Palin|PERSON', 0.5510910749435425), (u'Rick\_Perry|PERSON', 0.5508972406387329), (u'Stephen\_Colbert|PERSON', 0.5499709844589233), (u'Alex\_Jones|PERSON', 0.5492554306983948), (u'Michael\_Moore|PERSON', 0.5363447666168213), (u'Charles\_Manson|PERSON', 0.5363028645515442), (u'Dick\_Cheney|PERSON', 0.5348431468009949), (u'Mark\_Zuckerberg|PERSON', 0.5258212089538574), (u'Mark\_Wahlberg|PERSON', 0.5251839756965637), (u'Michael\_Jackson|PERSON', 0.5229078531265259)]

```

Discussion of Bill Cosby makes some obvious (and some less obvious) comparisons:

```

>>> model.most\_similar(['Bill\_Cosby|PERSON'])

[(u'Cosby|ORG', 0.6004706621170044), (u'Cosby|PERSON', 0.5874950885772705), (u'Roman\_Polanski|PERSON', 0.5478169918060303), (u'George\_Zimmerman|PERSON', 0.5398542881011963), (u'Charles\_Manson|PERSON', 0.5387344360351562), (u'OJ\_Simpson|PERSON', 0.5228893160820007), (u'Trayvon\_Martin|PERSON', 0.514190137386322), (u'Adnan\_Syed|PERSON', 0.49992451071739197), (u'rapist|NOUN', 0.49792540073394775), (u'srhbutts|NOUN', 0.49792492389678955)]

```

Some queries produce more confusing results:

```

>>> model.most\_similar(['Carrot\_Top|PERSON'])

[(u'Kate\_Mara|PERSON', 0.5347248911857605), (u'Andy\_Samberg|PERSON', 0.5336876511573792), (u'Ryan\_Gosling|PERSON', 0.5287898182868958), (u'Emma\_Stone|PERSON', 0.5243821740150452), (u'Charlie\_Sheen|PERSON', 0.5209298133850098), (u'Joseph\_Gordon\_Levitt|PERSON', 0.5196050405502319), (u'Jonah\_Hill|PERSON', 0.5151286125183105), (u'Zooey\_Deschanel|PERSON', 0.514430582523346), (u'Gerard\_Butler|PERSON', 0.5115377902984619), (u'Ellen\_Page|PERSON', 0.5094753503799438)]

```

I can't say the connection between Carrot Top and Kate Mara is obvious to me. I suppose this is true of most things about Carrot Top, so...Fair play.

3) Reddit talks about food a lot, and those regions of the vector space seem very well defined:

```

>>> model.most\_similar(['onion\_rings|NOUN'])

[(u'hashbrowns|NOUN', 0.8040812611579895), (u'hot\_dogs|NOUN', 0.7978234887123108), (u'chicken\_wings|NOUN', 0.793393611907959), (u'sandwiches|NOUN', 0.7903584241867065), (u'fries|NOUN', 0.7885469198226929), (u'tater\_tots|NOUN', 0.7821801900863647), (u'bagels|NOUN', 0.7788236141204834), (u'chicken\_nuggets|NOUN', 0.7787706255912781), (u'coleslaw|NOUN', 0.7771176099777222), (u'nachos|NOUN', 0.7755396366119385)]

```

Some of Reddit's ideas about food are kind of...interesting. It seems to think `bacon` and `brocolli` are very similar:

```

>>> model.similarity('bacon|NOUN', 'broccoli|NOUN')

0.83276615202851845

```

Reddit also thinks hot dogs are practically salad:

```

>>> model.similarity('hot\_dogs|NOUN', 'salad|NOUN')

0.76765100035460465

>>> model.similarity('hot\_dogs|NOUN', 'entrails|NOUN')

0.28360725445449464

```

Just keep telling yourself that Reddit.

-------------------------------------------------------------------------

2015-12-04T23:08:17Z iamtrask

Haha that Donald Trump one is quite something.

-------------------------------------------------------------------------

2015-12-04T23:35:45Z RXminuS

Love the link between bacon and broccoli, wonder what adding sentiment into the mix would change about that :P

```

Implementation details:

The data should be stored in a PreshMap instance.

The table should be keyed by a feat\_t (64 bit unisnged integer), with values being weight\_t\*, i.e. raw C arrays of floats.

Allocate the memory using cymem.Pool

```

Isn't this what we want to abstract away behind an interface so we can implement different ways of holding the vectors in memory, i.e. local vs central? \*\*Although to be honest...I'm starting to doubt my own idea in terms of if the speed tradeoff is even worth it.\*\*

```

Not so! The most similar words to take/NOUN might be things like opinion|NOUN, while the most similar words to take/VERB might be give|VERB or put|VERB. In normal Word2Vec these two share a key, so there's no way to look at the two different "senses" separately.

```

agreed, didn't really mean the information was usable or retrievable, rather that the scoring of vectors \*\*not\*\* using POS tagging is influenced by these "use cases" and that making this information explicit seems a natural extension. Bit hard to explain my brainfart...but just meant that your idea made sense :)

I'll read the paper and dig through some more code to get into it. But really love the work you're doing. Is there anything I can help out with straight away or you just want me to wait until you push your initial ideas?

-------------------------------------------------------------------------

2015-12-05T00:47:59Z honnibal

> Isn't this what we want to abstract away behind an interface so we can implement different ways of holding the vectors in memory, i.e. local vs central?

Hmm, maybe you're right. I was immediately thinking of how the C-level API would look.

> Although to be honest...I'm starting to doubt my own idea in terms of if the speed tradeoff is even worth it.

We might end up with use-cases where the vectors data is many gigabytes. Like, think trigram vectors, or vectors for subject/verb/object triples. If this occurs, the architecture you had in mind would make a lot of sense to me. A worker takes a few documents off the task queue, aggregates the vocabulary, and asks the vectors service for all vectors active on the batch.

-------------------------------------------------------------------------

2015-12-05T01:15:20Z RXminuS

Yeah for large vector models it would be a necessity, question is though where supporting that is on your timeline & plans for spaCy. For me it would be brilliant, because I have 100+ [Celery](http://celeryproject.org) workers and the 100M for each instance to load the vector model makes it hard to scale across docker containers. And in the future when we want to load more advanced, and possible context dependant models, and on the fly language switching it would be even more necessary.

When you push your preliminary sense2vec setup I can have a look and how I would change it to acc my use case, so we have something more concrete to design around. Then you can see if there's other places in the spaCy code that would need to change in accordance and we can orchestrate something from there :)

-------------------------------------------------------------------------

2015-12-05T01:55:58Z honnibal

I think there's a design problem here that should be fixed, so we may as well fix it sooner rather than later.

Send me an email, matt@spacy.io .

For now the following work-around could help:

1. Remove the `vec.bin` file from your data directory, to avoid loading the vectors

2. Make your own similarity server, that does the central look-up for you

3. Avoid the `.similarity()` methods on the spaCy objects.

You might want to look into an approximate nearest neighbours library, to avoid the n\*\*2 queries problem . Gensim recommends the `annoy` library. It seems good to me.

-------------------------------------------------------------------------

2015-12-05T04:26:20Z piskvorky

Hi guys, would appreciate your input on issue https://github.com/piskvorky/gensim/issues/527.

We're in the process of abstracting away particular vector stores (in-memory matrix, sharded on-disk store, approximate kNN index...) from gensim, behind a common API. What operations that API should support is an open question; knowing the use cases required by spaCy or other tools would be extremely useful!

How do you use such stores in spaCy, what metrics do you employ, what API signatures?

We'd like to end up with something that is flexible enough to cover all standard use cases (`give\_me\_vector\_for\_this\_key`, `give\_me\_most\_similar\_key\_to\_these\_other\_keys`, `index\_vector\_for\_this\_key` etc) but still concise and clearly scoped. This will be used throughout gensim (doc2vec, word2vec, docsim...).

CC @gojomo @tmylk

-------------------------------------------------------------------------

2015-12-05T09:03:17Z honnibal

Great!

Definitely want to get together on this. Will review.

-------------------------------------------------------------------------

2015-12-05T09:19:07Z jli05

Just to get enlightened: it seems great idea but does that mean that SpaCy and gensim will work together? Do we have a rough plan for the change of APIs?

-------------------------------------------------------------------------

2015-12-05T16:09:39Z honnibal

I want spaCy and Gensim to interoperate sanely. But it's more important that both libraries stay internally consistent, and they have fairly different API norms. I'd also rather spaCy didn't depend directly on Gensim, because that drags in scipy, so in total it's a fairly heavy-weight dependency. I'm guessing Gensim would hesitate to depend on spaCy. Among other things, we support a narrower range of platforms.

So, I'd say it's more of a design thing. We'd like to figure out what sort of work-flows are required.

-------------------------------------------------------------------------

2015-12-06T02:40:49Z piskvorky

Yes. I hope to discuss what kinds of behaviour people expect from such "vector stores", so we can design a sane API.

-------------------------------------------------------------------------

2015-12-08T19:20:13Z lechatpito

We've been using an external server for word2vec for over a year now. It would be great to be able to plug it in spaCy. Currently we are accessing the vector space through the https://github.com/3Top/word2vec-api/ project. A HTTP query will return a base 64 encoding of the vector. If there is any interest, I would be glad to improve the service to enable communication with spaCy.

-------------------------------------------------------------------------

2015-12-22T13:55:46Z jli05

Re: fusion of SpaCy and gensim APIs, I personally find the current gensim API tree not as straightforward/simple as scikit-learn (don't mistake me, gensim is extremely uniquely useful, e.g LDA, wikicorpus, etc). I think it'd be great to introduce an API lineage of scikit-learn flavour, or that simple.

As for SpaCy, I hope there's a portable way to train/retrieve the word embeddings across domains (pharma, legal, finance, etc) and natural languages. Could we make the underlying workings of word embeddings compositional/consistent as well (what if we need to do text analysis over legal+finance texts, or multi-lingual texts)?

Could SpaCy keep an eye on Apache Flink, Apache Spark, and TensorFlow's about-to-be-released distributed processing framework as well?

Please don't give up working on SpaCy. A versatile/portable/production-ready/modern NLP framework is never ever done before!

-------------------------------------------------------------------------

2015-12-23T16:14:12Z syllog1sm

We're definitely not about to give up working on SpaCy! We're barely getting started.

-------------------------------------------------------------------------

2016-06-16T10:46:15Z navesg

Hi, I am new to spacy and NLP and ML. I was going through the documentation of spacy. I am trying to make a QnA system. Was wondering if spacy gives a direct method to find similarity between 2 sentences? I could only find sentence tokenizations and word similarities.

-------------------------------------------------------------------------

2018-05-09T08:12:35Z lock[bot]

This thread has been automatically locked since there has not been any recent activity after it was closed. Please open a new issue for related bugs.

-------------------------------------------------------------------------