3. Word Vectors and spaCy

Dr. W.J.B. Mattingly
Smithsonian Data Science Lab and United States Holocaust Memorial Museum
August 2021

In this notebook is word vectors, or word embeddings. Because the English small model does not have these saved, we will be working with the next largest model, the English medium model, en_core_web_md. Let's import spaCy and download the medium model.

```
import spacy
!python -m spacy download en_core_web_md
```

Contents

Print to PDF

```
INFO:tensorflow:Enabling eager execution
INFO:tensorflow:Enabling v2 tensorshape
INFO:tensorflow:Enabling resource variables
INFO:tensorflow:Enabling tensor equality
INFO:tensorflow:Enabling control flow v2
Collecting en-core-web-md==3.1.0
  Downloading https://github.com/explosion/spacy-
models/releases/download/en_core_web_md-3.1.0/en_core_web_md-3.1.0-py3-none-any.whl
(45.4 MB)
Requirement already satisfied: spacy<3.2.0,>=3.1.0 in
c:\users\wma22\appdata\local\programs\python\python39\lib\site-packages (from en-
core-web-md==3.1.0) (3.1.1)
Requirement already satisfied: wasabi<1.1.0,>=0.8.1 in
c:\users\wma22\appdata\local\programs\python\python39\lib\site-packages (from
spacy<3.2.0,>=3.1.0->en-core-web-md==3.1.0) (0.8.2)
Requirement already satisfied: typer<0.4.0,>=0.3.0 in
c:\users\wma22\appdata\local\programs\python\python39\lib\site-packages (from
spacy<3.2.0,>=3.1.0->en-core-web-md==3.1.0) (0.3.2)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in
c:\users\wma22\appdata\local\programs\python\python39\lib\site-packages (from
spacy<3.2.0,>=3.1.0->en-core-web-md==3.1.0) (1.0.5)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in
c:\users\wma22\appdata\local\programs\python\python39\lib\site-packages (from
spacy<3.2.0,>=3.1.0->en-core-web-md==3.1.0) (3.0.5)
Requirement already satisfied: jinja2 in
c:\users\wma22\appdata\local\programs\python\python39\lib\site-packages (from
spacy<3.2.0,>=3.1.0->en-core-web-md==3.1.0) (2.11.3)
Requirement already satisfied: packaging>=20.0 in
c:\users\wma22\appdata\local\programs\python\python39\lib\site-packages (from
spacy<3.2.0,>=3.1.0->en-core-web-md==3.1.0) (20.9)
Requirement already satisfied: srsly<3.0.0,>=2.4.1 in
c:\users\wma22\appdata\local\programs\python\python39\lib\site-packages (from
spacy<3.2.0,>=3.1.0->en-core-web-md==3.1.0) (2.4.1)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.7 in
c:\users\wma22\appdata\local\programs\python\python39\lib\site-packages (from
spacy<3.2.0,>=3.1.0->en-core-web-md==3.1.0) (3.0.8)
Requirement already satisfied: pathy>=0.3.5 in
c:\users\wma22\appdata\local\programs\python\python39\lib\site-packages (from
spacy<3.2.0,>=3.1.0->en-core-web-md==3.1.0) (0.4.0)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in
c:\users\wma22\appdata\local\programs\python\python39\lib\site-packages (from
spacy<3.2.0,>=3.1.0->en-core-web-md==3.1.0) (4.59.0)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in
c:\users\wma22\appdata\local\programs\python\python39\lib\site-packages (from
spacy<3.2.0,>=3.1.0->en-core-web-md==3.1.0) (2.25.1)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in
c:\users\wma22\appdata\local\programs\python\python39\lib\site-packages (from
spacy<3.2.0,>=3.1.0->en-core-web-md==3.1.0) (2.0.5)
Requirement already satisfied: blis<0.8.0,>=0.4.0 in
c:\users\wma22\appdata\local\programs\python\python39\lib\site-packages (from
spacy<3.2.0,>=3.1.0->en-core-web-md==3.1.0) (0.7.4)
Requirement already satisfied: catalogue<2.1.0,>=2.0.4 in
c:\users\wma22\appdata\local\programs\python\python39\lib\site-packages (from
spacy<3.2.0,>=3.1.0->en-core-web-md==3.1.0) (2.0.4)
Requirement already satisfied: thinc<8.1.0,>=8.0.8 in
c:\users\wma22\appdata\local\programs\python\python39\lib\site-packages (from
spacy<3.2.0,>=3.1.0->en-core-web-md==3.1.0) (8.0.8)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<1.9.0,>=1.7.4 in
c:\users\wma22\appdata\local\programs\python\python39\lib\site-packages (from
spacy<3.2.0,>=3.1.0->en-core-web-md==3.1.0) (1.7.4)
Requirement already satisfied: setuptools in
c:\users\wma22\appdata\local\programs\python\python39\lib\site-packages (from
spacy<3.2.0,>=3.1.0->en-core-web-md==3.1.0) (54.1.1)
Requirement already satisfied: numpy>=1.15.0 in
c:\users\wma22\appdata\local\programs\python\python39\lib\site-packages (from
spacy<3.2.0,>=3.1.0->en-core-web-md==3.1.0) (1.20.2)
Requirement already satisfied: pyparsing>=2.0.2 in
c:\users\wma22\appdata\local\programs\python\python39\lib\site-packages (from
packaging>=20.0->spacy<3.2.0,>=3.1.0->en-core-web-md==3.1.0) (2.4.7)
Requirement already satisfied: smart-open<4.0.0,>=2.2.0 in
c:\users\wma22\appdata\local\programs\python\python39\lib\site-packages (from
pathy>=0.3.5->spacy<3.2.0,>=3.1.0->en-core-web-md==3.1.0) (3.0.0)
Requirement already satisfied: idna<3,>=2.5 in
c:\users\wma22\appdata\local\programs\python\python39\lib\site-packages (from
requests<3.0.0,>=2.13.0->spacy<3.2.0,>=3.1.0->en-core-web-md==3.1.0) (2.10)
Requirement already satisfied: chardet<5,>=3.0.2 in
c:\users\wma22\appdata\local\programs\python\python39\lib\site-packages (from
requests<3.0.0,>=2.13.0->spacy<3.2.0,>=3.1.0->en-core-web-md==3.1.0) (4.0.0)
Requirement already satisfied: certifi>=2017.4.17 in
c:\users\wma22\appdata\local\programs\python\python39\lib\site-packages (from
requests<3.0.0,>=2.13.0->spacy<3.2.0,>=3.1.0->en-core-web-md==3.1.0) (2020.12.5)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
c:\users\wma22\appdata\local\programs\python\python39\lib\site-packages (from
requests<3.0.0,>=2.13.0->spacy<3.2.0,>=3.1.0->en-core-web-md==3.1.0) (1.26.3)
Requirement already satisfied: click<7.2.0,>=7.1.1 in
c:\users\wma22\appdata\local\programs\python\python39\lib\site-packages (from
typer<0.4.0,>=0.3.0->spacy<3.2.0,>=3.1.0->en-core-web-md==3.1.0) (7.1.2)
Requirement already satisfied: MarkupSafe>=0.23 in
c:\users\wma22\appdata\local\programs\python\python39\lib\site-packages (from jinja2-
```

```
>spacy<3.2.0,>=3.1.0->en-core-web-md==3.1.0) (1.1.1)
[+] Download and installation successful
You can now load the package via spacy.load('en_core_web_md')
```

```
2021-09-07 15:23:33.303294: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened
dynamic library cudart64_110.dll
WARNING: You are using pip version 21.1.2; however, version 21.2.4 is available.
You should consider upgrading via the
'C:\Users\wma22\AppData\Local\Programs\Python\Python39\python.exe -m pip install --
upgrade pip' command.
```

```
nlp = spacy.load("en_core_web_md")
with open ("data/wiki_us.txt", "r") as f:
    text = f.read()
doc = nlp(text)
sentence1 = list(doc.sents)[0]
```

3.1. Word Vectors

Word vectors, or word embeddings, are numerical representations of words in multidimensional space through matrices. The purpose of the word vector is to get a computer system to understand a word. Computers cannot understand text efficiently. They can, however, process numbers quickly and well. For this reason, it is important to convert a word into a number.

Initial methods for creating word vectors in a pipeline take all words in a corpus and convert them into a single, unique number. These words are then stored in a dictionary that would look like this: {“the”: 1, “a”, 2} etc. This is known as a bag of words. This approach to representing words numerically, however, only allow a computer to understand words numerically to identify unique words. It does not, however, allow a computer to understand meaning.

Imagine this scenario:

Tom loves to eat chocolate.

Tom likes to eat chocolate.

These sentences represented as a numerical array (list) would look like this:

1, 2, 3, 4, 5

1, 6, 3, 4, 5

As we can see, as humans both sentences are nearly identical. The only difference is the degree to which Tom appreciates eating chocolate. If we examine the numbers, however, these two sentences seem quite close, but their semantical meaning is impossible to know for certain. How similar is 2 to 6? The number 6 could represent “hates” as much as it could represent “likes”. This is where word vectors come in.

Word vectors take these one dimensional bag of words and gives them multidimensional meaning by representing them in higher dimensional space, noted above. This is achieved through machine learning and can be easily achieved via Python libraries, such as Gensim, which we will explore more closely in the next notebook.

3.1.1. Why use Word Vectors?

The goal of word vectors is to achieve numerical understanding of language so that a computer can perform more complex tasks on that corpus. Let’s consider the example above. How do we get a computer to understand 2 and 6 are synonyms or mean something similar? One option you might be thinking is to simply give the computer a synonym dictionary. It can look up synonyms and then know what words mean. This approach, on the surface, makes perfect sense, but let’s explore that option and see why it cannot possibly work.

For the example below, we will be using the Python library PyDictionary which allows us to look up definitions and synonyms of words.

```
from PyDictionary import PyDictionary

dictionary=PyDictionary()
text = "Tom loves to eat chocolate"

words = text.split()
for word in words:
    syns = dictionary.synonym(word)
    print (f"{word}: {syns[0:5]}\n")
```

```
Tom has no Synonyms in the API
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-17-2fe432a3b47a> in <module>
      7 for word in words:
      8     syns = dictionary.synonym(word)
----> 9     print (f"{word}: {syns[0:5]}\n")

TypeError: 'NoneType' object is not subscriptable
```

Even with the simple sentence, the results are comically bad. Why? The reason is because synonym substitution, a common method of data augmentation, does not take into account syntactical differences of synonyms. I do not believe anyone would think “Felis domesticus”, the Latin name of the common house cat, would be an adequite substitution for the name Tom. Nor is “garbage down” a really proper synonym for eat.

Perhaps, then we could use synonyms to find words that have cross-terms, or terms that appear in both synonym sets.

```
from PyDictionary import PyDictionary

dictionary=PyDictionary()

words  = ["like", "love"]
for word in words:
    syns = dictionary.synonym(word)
    print (f"{word}: {syns[0:5]}\n")
```

```
like has no Synonyms in the API
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-18-fefd3418da3d> in <module>
      6 for word in words:
      7     syns = dictionary.synonym(word)
----> 8     print (f"{word}: {syns[0:5]}\n")

TypeError: 'NoneType' object is not subscriptable
```

This, as we can see, has some potential to work, but again it is not entirely reliable and to work with such a list would be computationally expensive. For both of these reasons, word vectors are prefered. The reason? Because they are formed by the computer on corpora for a specific task. Further, they are numerical in nature (not a dictionary of words), meaning the computer can process them more quickly.

3.1.2. What do Word Vectors Look Like?

Word vectors have a preset number of dimensions. These dimensions are honed via machine learned. Models take into account word frequency alongside words across a corpus and the appearance of other words in similar contexts. This allows for the the computer to determine the syntactical similarity of words numerically. It then needs to represent these relationships numerically. It does this through the vector, or a matrix of matrices. To represent these more concisely, models flatten a matrix to a float (decimal number). The number of dimensions represent the number of floats in the matrix.

Let’s take a look at the first word in our sentence. Specifically, let’s look at its vector.

```
sentence1[0].vector
```

```
array([ 2.7204e-01, -6.2030e-02, -1.8840e-01,  2.3225e-02, -1.8158e-02,
        6.7192e-03, -1.3877e-01,  1.7708e-01,  1.7709e-01,  2.5882e+00,
       -3.5179e-01, -1.7312e-01,  4.3285e-01, -1.0708e-01,  1.5006e-01,
       -1.9982e-01, -1.9093e-01,  1.1871e+00, -1.6207e-01, -2.3538e-01,
        3.6640e-03, -1.9156e-01, -8.5662e-02,  3.9199e-02, -6.6449e-02,
       -4.2090e-02, -1.9122e-01,  1.1679e-02, -3.7138e-01,  2.1886e-01,
        1.1423e-03,  4.3190e-01, -1.4205e-01,  3.8059e-01,  3.0654e-01,
        2.0167e-02, -1.8316e-01, -6.5186e-03, -8.0549e-03, -1.2063e-01,
        2.7507e-02,  2.9839e-01, -2.2896e-01, -2.2882e-01,  1.4671e-01,
       -7.6301e-02, -1.2680e-01, -6.6651e-03, -5.2795e-02,  1.4258e-01,
        1.5610e-01,  5.5510e-02, -1.6149e-01,  9.6290e-02, -7.6533e-02,
       -4.9971e-02, -1.0195e-02, -4.7641e-02, -1.6679e-01, -2.3940e-01,
        5.0141e-03, -4.9175e-02,  1.3338e-02,  4.1923e-01, -1.0104e-01,
        1.5111e-02, -7.7706e-02, -1.3471e-01,  1.1900e-01,  1.0802e-01,
        2.1061e-01, -5.1904e-02,  1.8527e-01,  1.7856e-01,  4.1293e-02,
       -1.4385e-02, -8.2567e-02, -3.5483e-02, -7.6173e-02, -4.5367e-02,
        8.9281e-02,  3.3672e-01, -2.2099e-01, -6.7275e-03,  2.3983e-01,
       -2.3147e-01, -8.8592e-01,  9.1297e-02, -1.2123e-02,  1.3233e-02,
       -2.5799e-01, -2.9720e-02,  1.6754e-02,  1.3690e-02,  3.2377e-01,
        3.9546e-02,  4.2114e-02, -8.8243e-02,  3.0318e-01,  8.7747e-02,
        1.6346e-01, -4.0485e-01, -4.3845e-02, -4.0697e-02,  2.0936e-01,
       -7.7795e-01,  2.9970e-01,  2.3340e-01,  1.4891e-01, -3.9037e-01,
       -5.3086e-02,  6.2922e-02,  6.5663e-02, -1.3906e-01,  9.4193e-02,
        1.0344e-01, -2.7970e-01,  2.8905e-01, -3.2161e-01,  2.0687e-02,
        6.3254e-02, -2.3257e-01, -4.3520e-01, -1.7049e-02, -3.2744e-01,
       -4.7064e-02, -7.5149e-02, -1.8788e-01, -1.5017e-02,  2.9342e-02,
       -3.5270e-01, -4.4278e-02, -1.3507e-01, -1.1644e-01, -1.0430e-01,
        1.3920e-01,  3.9199e-03,  3.7603e-01,  6.7217e-02, -3.7992e-01,
       -1.1241e+00, -5.7357e-02, -1.6826e-01,  3.9410e-02,  2.6040e-01,
       -2.3866e-02,  1.7963e-01,  1.3553e-01,  2.1390e-01,  5.2633e-02,
       -2.5033e-01, -1.1307e-01,  2.2234e-01,  6.6597e-02, -1.1161e-01,
        6.2438e-02, -2.7972e-01,  1.9878e-01, -3.6262e-01, -1.0006e-05,
       -1.7262e-01,  2.9166e-01, -1.5723e-01,  5.4295e-02,  6.1010e-02,
       -3.9165e-01,  2.7660e-01,  5.7816e-02,  3.9709e-01,  2.5229e-02,
        2.4672e-01, -8.9050e-02,  1.5683e-01, -2.0960e-01, -2.2196e-01,
        5.2394e-02, -1.1360e-02,  5.0417e-02, -1.4023e-01, -4.2825e-02,
       -3.1931e-02, -2.1336e-01, -2.0402e-01, -2.3272e-01,  7.4490e-02,
        8.8202e-02, -1.1063e-01, -3.3526e-01, -1.4028e-02, -2.9429e-01,
       -8.6911e-02, -1.3210e-01, -4.3616e-01,  2.0513e-01,  7.9362e-03,
        4.8505e-01,  6.4237e-02,  1.4261e-01, -4.3711e-01,  1.2783e-01,
       -1.3111e-01,  2.4673e-01, -2.7496e-01,  1.5896e-01,  4.3314e-01,
        9.0286e-02,  2.4662e-01,  6.6463e-02, -2.0099e-01,  1.1010e-01,
        3.6440e-02,  1.7359e-01, -1.5689e-01, -8.6328e-02, -1.7316e-01,
        3.6975e-01, -4.0317e-01, -6.4814e-02, -3.4166e-02, -1.3773e-02,
        6.2854e-02, -1.7183e-01, -1.2366e-01, -3.4663e-02, -2.2793e-01,
       -2.3172e-01,  2.3900e-01,  2.7473e-01,  1.5332e-01,  1.0661e-01,
       -6.0982e-02, -2.4805e-02, -1.3478e-01,  1.7932e-01, -3.7374e-01,
       -2.8930e-02, -1.1142e-01, -8.3890e-02, -5.5932e-02,  6.8039e-02,
       -1.0783e-01,  1.4650e-01,  9.4617e-02, -8.4554e-02,  6.7429e-02,
       -3.2910e-01,  3.4082e-02, -1.6747e-01, -2.5997e-01, -2.2917e-01,
        2.0159e-02, -2.7580e-02,  1.6136e-01, -1.8538e-01,  3.7665e-02,
        5.7603e-01,  2.0684e-01,  2.7941e-01,  1.6477e-01, -1.8769e-02,
        1.2062e-01,  6.9648e-02,  5.9022e-02, -2.3154e-01,  2.4095e-01,
       -3.4710e-01,  4.8540e-02, -5.6502e-02,  4.1566e-01, -4.3194e-01,
        4.8230e-01, -5.1759e-02, -2.7285e-01, -2.5893e-01,  1.6555e-01,
       -1.8310e-01, -6.7340e-02,  4.2457e-01,  1.0346e-02,  1.4237e-01,
        2.5939e-01,  1.7123e-01, -1.3821e-01, -6.6846e-02,  1.5981e-02,
       -3.0193e-01,  4.3579e-02, -4.3102e-02,  3.5025e-01, -1.9681e-01,
       -4.2810e-01,  1.6899e-01,  2.2511e-01, -2.8557e-01, -1.0280e-01,
       -1.8168e-02,  1.1407e-01,  1.3015e-01, -1.8317e-01,  1.3230e-01],
      dtype=float32)
```

3.1.3. Why use Word Vectors?

Once a word vector model is trained, we can do similarity matches very quickly and very reliably. Let's explore some vectors from our medium sized model. Let's specifically try and find the words most closely related to the word dog.

```
#https://stackoverflow.com/questions/54717449/mapping-word-vector-to-the-most-similar-closest-word-using-spacy
your_word = "dog"

ms = nlp.vocab.vectors.most_similar(
    np.asarray([nlp.vocab.vectors[nlp.vocab.strings[your_word]]]), n=10)
words = [nlp.vocab.strings[w] for w in ms[0][0]]
distances = ms[2]
print(words)
```

```
['dog', 'KENNEL', 'dogs', 'CANINES', 'GREYHOUND', 'pet', 'Pet-Care', 'FELINE', 'cat',
'BEAGLES']
```

3.2. Doc Similarity

In spaCy we can do this same thing at the document level. Through word vectors we can calculate the similarity between two documents. Let's look at the example from spaCy's documentation.

```
nlp = spacy.load("en_core_web_md")  # make sure to use larger package!
doc1 = nlp("I like salty fries and hamburgers.")
doc2 = nlp("Fast food tastes very good.")

# Similarity of two documents
print(doc1, "<->", doc2, doc1.similarity(doc2))
```

```
I like salty fries and hamburgers. <-> Fast food tastes very good. 0.7799485853415737
```

3.3. Word Similarity

We can also calculate the similarity between two given words.

```
# Similarity of tokens and spans
french_fries = doc1[2:4]
burgers = doc1[5]
print(french_fries, "<->", burgers, french_fries.similarity(burgers))
```

```
salty fries <-> hamburgers 0.7304624
```

3.4. Conclusion

As we have seen in this notebook, spaCy is made up of a series of complex P

By William Mattingly