lab10a

October 31, 2019

# Training a word2vec model from scratch

– Prof. Dorien Herremans

We will start by training a word2vec model from scratch using the gensim library. You will need to ensure that you have gensim installed, and a file decompressor to load our dataset.

Note: these models may take a while to train. Be sure to switch the runtime of Google Colab to us a TPU or GPU hardware accellerator (in the menu at the top).

Let’s start by installing some libraries that we will use:

In [4]: !pip install gensim

!pip install wget

Requirement already satisfied: gensim in /usr/local/lib/python3.6/dist-packages (3.6.0) Requirement already satisfied: smart-open>=1.2.1 in /usr/local/lib/python3.6/dist-packages (fr Requirement already satisfied: numpy>=1.11.3 in /usr/local/lib/python3.6/dist-packages (from Requirement already satisfied: scipy>=0.18.1 in /usr/local/lib/python3.6/dist-packages (from Requirement already satisfied: six>=1.5.0 in /usr/local/lib/python3.6/dist-packages (from gens Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from smart- Requirement already satisfied: boto>=2.32 in /usr/local/lib/python3.6/dist-packages (from Requirement already satisfied: boto3 in /usr/local/lib/python3.6/dist-packages (from smart-ope Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (f Requirement already satisfied: urllib3<1.25,>=1.21.1 in /usr/local/lib/python3.6/dist-packages Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /usr/local/lib/python3.6/dist-packages Requirement already satisfied: idna<2.9,>=2.5 in /usr/local/lib/python3.6/dist-packages (from Requirement already satisfied: botocore<1.14.0,>=1.13.2 in /usr/local/lib/python3.6/dist-packa Requirement already satisfied: s3transfer<0.3.0,>=0.2.0 in /usr/local/lib/python3.6/dist-packa Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /usr/local/lib/python3.6/dist-package Requirement already satisfied: docutils<0.16,>=0.10 in /usr/local/lib/python3.6/dist-packages Requirement already satisfied: python-dateutil<3.0.0,>=2.1; python\_version >= "2.7" in Requirement already satisfied: wget in /usr/local/lib/python3.6/dist-packages (3.2)

Now we can import these libraries:

In [0]: *# imports needed*

**import gensim import wget**

We will train our model using a very small dataset for demonstrative purposes. Note that for a real data science project you should train on a much larger dataset.

We will use the complete works of Shakespeare. You can find the file at https://dorienherremans.com/drop/CDS/CNNs/shakespeare.txt

In [6]: *# download the dataset*

!wget "https://dorienherremans.com/drop/CDS/CNNs/shakespeare.txt"

--2019-10-31 03:14:29-- https://dorienherremans.com/drop/CDS/CNNs/shakespeare.txt Resolving dorienherremans.com (dorienherremans.com)... 96.127.180.74

Connecting to dorienherremans.com (dorienherremans.com)|96.127.180.74|:443... connected. HTTP request sent, awaiting response... 200 OK

Length: 5447743 (5.2M) [text/plain]

Saving to: shakespeare.txt.3

shakespeare.txt.3 100 [===================>] 5.20M --.-KB/s in 0.1s 2019-10-31 03:14:29 (48.0 MB/s) - shakespeare.txt.3 saved [5447743/5447743]

Let’s read the input file and convert each line into a list of words (tokenizing). Do do this, we create a function read\_input which is called in the penultimate line below:

In [7]: **def** read\_input(input\_file):

print("reading file...")

**with** open (input\_file, 'r') **as** f: lines = f.readlines()

**for** line **in** lines:

*# do some pre-processing and return a (tokenized) list # of words for each review text*

*# you can print the output here to understand # the preprocessing (tokenizing)*

**yield** gensim.utils.simple\_preprocess (line)

*# each review item new becomes a series of words # this is a list of lists*

*# point to the location on your filesystem*

data\_file = 'shakespeare.txt'

documents = list (read\_input (data\_file)) print("Done reading data file")

reading file...

Done reading data file

Now let’s train the word2vec model using our document variable (which is a list of word lists). Note that you can specify a number of hyperparameters below: \* min\_count removes all words that occur less then min\_count \* window: window size in the skip-gram \* workers: how many threads to use \* size: number of dimension of your new word embedding vector (typically 100-200). Smaller datasets require a smaller number

In [8]: model = gensim.models.Word2Vec (documents, size=150, window=5, min\_count=2, workers=4) model.train(documents,total\_examples=len(documents),epochs=10)

Out[8]: (6703813, 8675160)

That’s it! Now you’ve trained the model!

Now let’s explore some properties of our new word space. You can get the words most close (read: most similar) to a given word. Remember, the only texts the model has seen is shakespeare!

In [9]: w1 = "king"

model.wv.most\_similar (positive=w1)

/usr/local/lib/python3.6/dist-packages/gensim/matutils.py:737: FutureWarning: Conversion of th if np.issubdtype(vec.dtype, np.int):

Out[9]: [('prince', 0.6737143993377686),

('fifth', 0.5719449520111084),

('warwick', 0.561543881893158),

('duke', 0.5592024922370911),

('plantagenets', 0.543491780757904),

('sixth', 0.5396798253059387),

('bolingbroke', 0.5250260829925537),

('dauphin', 0.502298891544342),

('emperor', 0.5010416507720947),

('princess', 0.5002898573875427)]

In [10]: *# look up top 6 words similar to 'smile'*

w1 = ["smile"]

model.wv.most\_similar (positive=w1,topn=6)

/usr/local/lib/python3.6/dist-packages/gensim/matutils.py:737: FutureWarning: Conversion of th if np.issubdtype(vec.dtype, np.int):

Out[10]: [('laugh', 0.7408881187438965),

('grieve', 0.6916569471359253),

('wink', 0.6894698739051819),

('shine', 0.67576664686203),

('rail', 0.6736947298049927),

('lodge', 0.6682363152503967)]

In [11]: *# look up top 6 words similar to 'france'*

w1 = ["france"]

model.wv.most\_similar (positive=w1,topn=6)

/usr/local/lib/python3.6/dist-packages/gensim/matutils.py:737: FutureWarning: Conversion of th if np.issubdtype(vec.dtype, np.int):

Out[11]: [('england', 0.6454899907112122),

('princess', 0.5820638537406921),

('egypt', 0.5726439356803894),

('britain', 0.5668758153915405),

('wales', 0.5483297109603882),

('scotland', 0.5341385006904602)]

In [12]: *# look up top 6 words similar to 'sword'*

w1 = ["sword"]

model.wv.most\_similar (positive=w1,topn=6)

/usr/local/lib/python3.6/dist-packages/gensim/matutils.py:737: FutureWarning: Conversion of th if np.issubdtype(vec.dtype, np.int):

Out[12]: [('head', 0.7629855275154114),

('knife', 0.7423747777938843),

('pocket', 0.7103098034858704),

('dagger', 0.7093837857246399),

('weapon', 0.6861767768859863),

('finger', 0.6814857721328735)]

In [14]: *# get everything related to stuff on the royalty and not related to farmer*

w1 = ["king",'queen','prince'] w2 = ['farmer']

model.wv.most\_similar (positive=w1,negative=w2,topn=10)

/usr/local/lib/python3.6/dist-packages/gensim/matutils.py:737: FutureWarning: Conversion of th if np.issubdtype(vec.dtype, np.int):

Out[14]: [('princess', 0.665969729423523),

('duke', 0.58847576379776),

('warwick', 0.5665985345840454),

('emperor', 0.5446041226387024),

('bolingbroke', 0.5294402837753296),

('cousin', 0.5196998119354248),

('moor', 0.5129690766334534),

('duchess', 0.5111926198005676),

('comfort', 0.5026594400405884),

('ghost', 0.5024846196174622)]

Explore the similarity (e.g. distance) between two words. Does it make sense?

In [15]: *# similarity between two similar words*

model.wv.similarity(w1="pretty",w2="beautiful")

/usr/local/lib/python3.6/dist-packages/gensim/matutils.py:737: FutureWarning: Conversion of th if np.issubdtype(vec.dtype, np.int):

Out[15]: 0.4908297

In [21]: *# similarity between two opposing words*

model.wv.similarity(w1="king",w2="farmer")

/usr/local/lib/python3.6/dist-packages/gensim/matutils.py:737: FutureWarning: Conversion of th if np.issubdtype(vec.dtype, np.int):

Out[21]: -0.014322285

Try some other combinations :)

We can even use it to perform more ‘smart’ assigments:

In [22]: *# Which one is the odd one out in this list?*

model.wv.doesnt\_match(["cat","dog","france"])

/usr/local/lib/python3.6/dist-packages/gensim/models/keyedvectors.py:895: FutureWarning: array vectors = vstack(self.word\_vec(word, use\_norm=True) for word in used\_words).astype(REAL)

/usr/local/lib/python3.6/dist-packages/gensim/matutils.py:737: FutureWarning: Conversion of th if np.issubdtype(vec.dtype, np.int):

Out[22]: 'france'

If you are interested in plotting the words in a multidimensional space, you can actually get the vector coordinates of each word:

In [23]: model.wv['france']

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Out[23]: array([ 0.12898712, | 0.4671235 | , | -1.206576 , | -0.36595318, | | 0.1506167 , |
| 1.1793954 , | -1.1084852 | , | -0.8075331 , | 1.0825065 , | | 0.70289874, |
| -0.28459004, | 1.6774065 | , | -0.9588128 , | -0.22732303, | | -0.98132896, |
| 0.35517594, | 0.728734 | , | 0.13602592, | -0.8034905 , | | 0.70249957, |
| -0.04749535, | 0.08824098, | | -0.32765052, | -0.35695317, | | -0.46274334, |
| 1.7779108 , | 0.3191873 , | | -0.5627077 , | 0.149659 , | | 0.3987616 , |
| -0.91528064, | 1.0425785 , | | -0.9436222 , | -0.6722221 | , | 0.04131044, |
| 0.22009298, | -0.540169 , | | -0.7225806 , | 0.711222 | , | -0.8351769 , |
| -0.937331 , | 0.51296544, | | 0.18035546, | -0.5959368 | , | 0.40614873, |
| 0.52493775, | -0.11186406, | | -0.17614752, | -0.4624433 | , | -0.03196685, |
| 1.1612102 , | 1.3868464 , | | -0.10332501, | 1.6387349 | , | -1.1230714 , |
| -1.7633582 , | 0.94452757, | | -0.18208385, | 1.3770766 | , | -0.0186035 , |
| 1.5058436 , | 0.57089794, | | -1.3749561 , | 0.90754074, | | 0.41828552, |
| 0.38541046, | 0.65224403, | | 1.4327322 , | -1.013402 , | | -1.3828125 , |
| -0.39803568, | -0.13882606, | | -0.6142673 , | -0.8853668 , | | 0.22329934, |
| -1.1264523 , | -0.18498729, | | -0.34871644, | 0.590621 , | | -0.63116354, |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| -0.6274037 , | 0.50085706, | -0.1797085 , | -1.0104762 , | -0.00731183, |
| 0.89476156, | 0.2195267 , | 1.010621 , | 0.2613935 , | -0.3054644 , |
| 0.83509386, | -1.1588943 , | -0.5723498 , | 0.65537244, | 1.5544599 , |
| -1.1141642 , | -0.89630795, | 0.19832838, | 1.2410437 , | 0.04946405, |
| 0.43637577, | -1.2427082 , | -0.08336011, | 0.1798651 , | 0.46433887, |
| -0.90536165, | 1.0460583 , | 0.278637 , | 0.82398534, | -0.6504324 , |
| -0.18127276, | 0.05864932, | -0.19327986, | -0.6500451 , | -0.65463656, |
| -0.24228968, | -0.54006565, | -0.47732472, | 0.43810418, | -0.45715013, |
| 0.56704015, | 0.67070955, | -0.3916365 , | 0.3965296 , | -0.03325389, |
| 1.6251537 , | -0.7214478 , | -0.25146684, | -0.5156361 , | -0.2521658 , |
| -0.09160373, | 0.12279724, | -0.21194759, | 0.9699577 , | -1.4463458 , |
| 0.24896291, | -0.6814331 , | -0.14928418, | 1.5542376 , | 0.09362388, |
| 0.6996671 , | 0.983275 , | -0.3128055 , | -0.48396447, | -0.47671464, |
| 0.41892102, | 1.2290125 , | 0.3226385 , | 1.3000402 , | 0.18256927], |
| dtype=float32) |  |  |  |  |

# Bonus: visualising our model in t-SNE:

In [25]: **from sklearn.manifold import** TSNE **import matplotlib.pyplot as plt matplotlib** inline

**def** tsne\_plot(model):

"Creates and TSNE model and plots it"

*# fyi: to test specific labels instead of all the words in the vocab: # labels = ['king', 'queen', 'prince', 'farmer', 'blue', 'red']*

*# tokens = []*

*# for label in labels:*

*# tokens.append(model[label])*

labels = [] tokens = []

count = 0

**for** word **in** model.wv.vocab:

*# to speed up the process, let's limit to the first 100 elements*

**if** count < 100: tokens.append(model[word]) labels.append(word)

count = count+1

*# set the t-sne values*

tsne\_model = TSNE(perplexity=40, n\_components=2, init='pca', n\_iter=2500, random\_ new\_values = tsne\_model.fit\_transform(tokens)

x = []

y = []

**for** value **in** new\_values: x.append(value[0]) y.append(value[1])

plt.figure(figsize=(16, 16))

**for** i **in** range(len(x)): plt.scatter(x[i],y[i]) plt.annotate(labels[i],

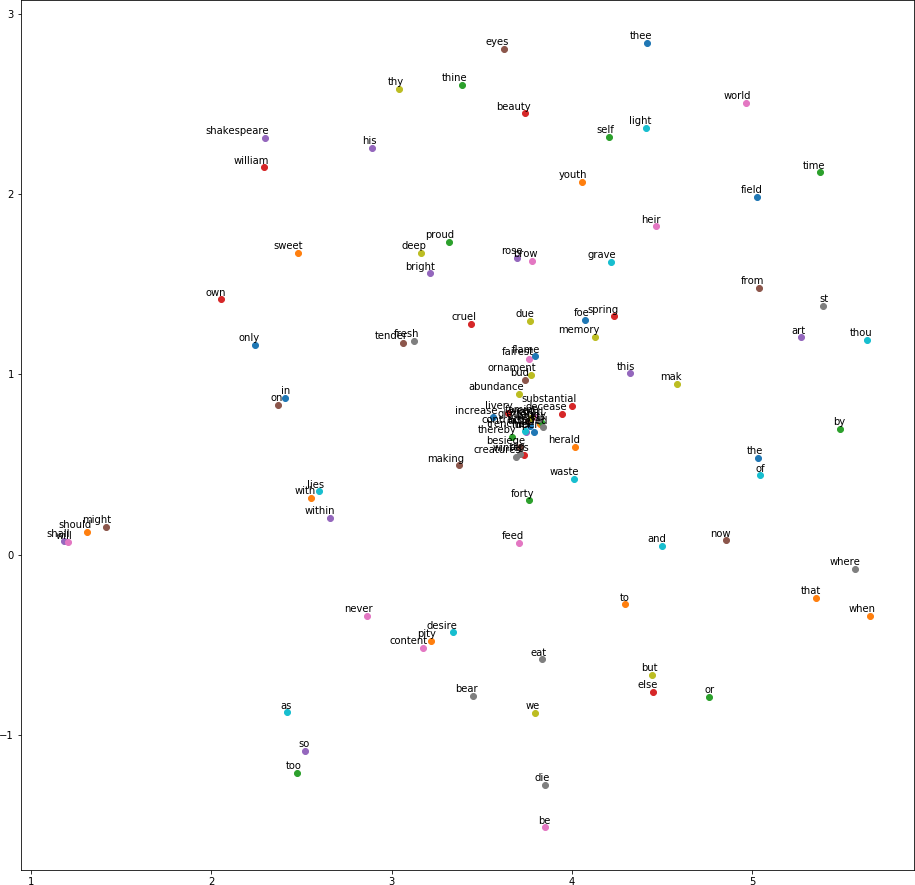
xy=(x[i], y[i]), xytext=(5, 2),

textcoords='offset points', ha='right',

va='bottom')

plt.show() tsne\_plot(model)

/usr/local/lib/python3.6/dist-packages/ipykernel\_launcher.py:21: DeprecationWarning: Call to



# References

* + - https://radimrehurek.com/gensim/models/word2vec.html
    - https://towardsdatascience.com/multi-class-text-classification-model-comparison-and- selection-5eb066197568
    - https://github.com/kavgan/nlp-text-mining-working-examples/tree/master/word2vec
    - [https://medium.com/@mishra.thedeepak/doc2vec-simple-implementation-example-](https://medium.com/%40mishra.thedeepak/doc2vec-simple-implementation-example-) df2afbbfbad5