Assignment 6: Text Analysis

Low, Daniel Mark (S3120155)
Petre, Bogdan (S3480941)
Xu, Teng Andrea (S3548120)

Group 7

October 16, 2017

1 State of the corpus

1.1 Compute the term frequencies for all the documents in the corpus.

The code for this exercise is in **1_1_term_frequency.py**.
For this exercise we used:

- Gutenberg Corpus (also for the following exercises)
- nltk word_tokenizer
- nltk WordNetLemmatizer
- nltk stopwords
- string.punctuation
- python function case_fold()

We did the same exercise in two different ways:

- A: thinking of an optimal solution for exercise 1.5., we removed the stopwords and normalized the term frequency with respect to the corresponding doc_length. The reason is 1.5. requires normalization.
- B: for exercise 1.2., we kept the stopwords and we haven't used any normalization. The reason for the distinction is to obtain a Zipf distribution using also stopwords.

The core algorithm for both preprocessing methods was to do, for each book in the corpus, the following:

1. Clean → for each unique word in the corpus, keep/remove stopwords, remove punctuation, case fold, lemmatize and tokenize.

2. term_frequency → after the cleaning part we can count for each word the normalized/not_normalized term frequency.

Finally we have a dictionary with the title of the book as key and as value another dictionary with key the word and value the frequency. The results are saved as csv in the corresponding folder (term_frequency1, term_frequency2).

1.2 Verify that the word frequencies follow Zipf's law, by plotting the frequency of the words vs. the rank of the words on a log-log scale. Plot the best-fitting Zipf distribution as well in the same figure.

The code for this exercise is in **1_2_plot.py**. In Figure 1, we see the distributions seem to follow Zipfs Law, which is plotted as an ideal distribution in black dotted lines.

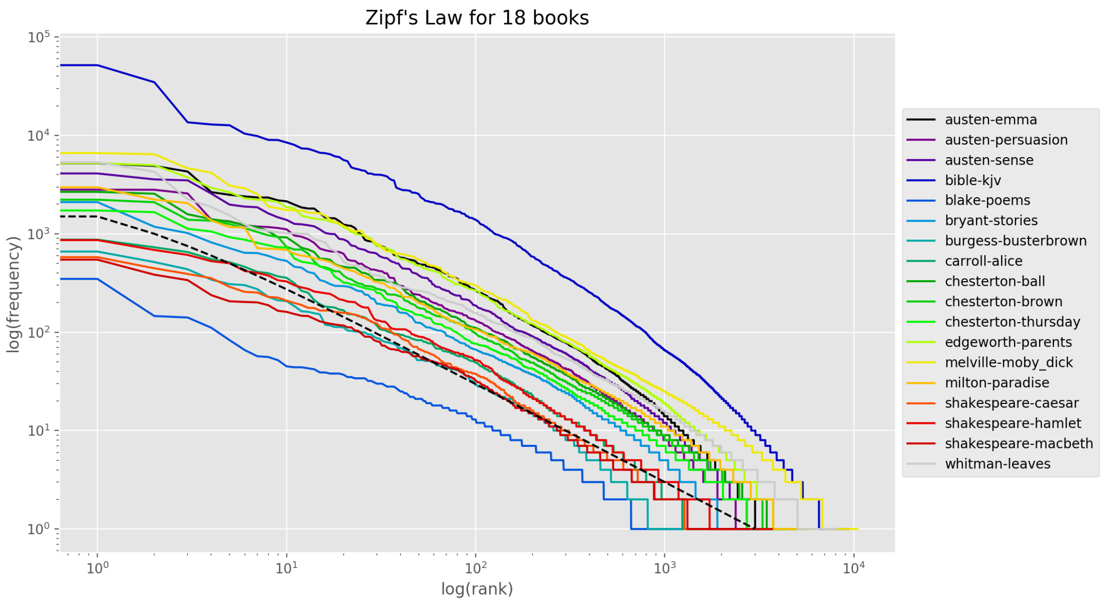

Figure 1: Zipfs Law for 18 books and an ideal power law in black dotted line.

1.3 Clean up the data as much as possible.

The code can be found in **1_3_data_cleaner.py**.
We used the same tools described in exercise 1.1.

1.4 Find the most frequent 10 collocations of three words in the corpus, in which at least two words are not proper names. What happens if you increase the size of the sliding window in which you are looking for collocations?

During cleaning, we used the pos_tag function to determine if the word is a Proper Noun (NNP). In that case, the word is left capitalized. We look for collocations but exclude any that have more that one word capitalized. And, for instance, collocations such as ('District', 'Columbia', 'Florida') and ('scene', 'Mary', 'Susan') were excluded. Below, we can see the top 10 collocations for the 18 books.

	austen-emma.txt	austen-persuasion.txt	austen-sense.txt
0	(avowed, adoption, assume)	(anyone, intending, inhabit)	(apricot, marmalade, successfully)
1	(beaver, York, tan)	(au, fait, newest)	(ash, acacia, thick)
2	(cameo, coral, shell)	(bawling, newspaperman, muffin)	(boiled, fowl, veal)
3	(cellery, beet, root)	(beautifying, nosegay, ate)	(cod, boiled, fowl)
4	(column, smoke, ascending)	(bustling, housekeeper, flirting)	(fir, mountain, ash)
5	(comtesse, ostalis, madame)	(carefully, steep, flight)	(fowl, veal, cutlet)
6	(dexterously, throwing, ditch)	(compact, tight, parsonage)	(grey, moss, brush)
7	(drawer, medal, cameo)	(covet, loving, longest)	(hen, forsaking, nest)
8	(engraving, drawer, medal)	(cutting, silk, gold)	(jane, austen, 1811)
9	(hymen, saffron, robe)	(dray, bawling, newspaperman)	(lodge, st, James)

	bible-kjv.txt	blake-poems.txt	bryant-stories.txt
0	(Asshurim, letushim, leummim)	(Bard, present, past)	(alang, craping, shtill)
1	(Kinah, dimonah, adadah)	(Dick, joe, ned)	(amused, somewhat, distressed)
2	(abagtha, zethar, carcas)	(Grey, headed, beadle)	(aster, hollyhock, stock)
3	(adalia, aridatha, parmashta)	(London, town, seated)	(beau, ty, flashed)
4	(adithaim, gederah, gederothaim)	(appals, hapless, soldier)	(budge, coaxing, scolding)
5	(alammelech, amad, misheal)	(artful, teazing, folly)	(bunch, sausage, spicy)
6	(apharsathchites, tarpelites, apharsites)	(author, printer, willm)	(cone, bryant, 1918)
7	(aridatha, parmashta, arisai)	(blight, plague, marriage)	(dust, skylark, fluttered)
8	(arisai, aridai, vajezatha)	(blushed, rosy, red)	(endive, mustard, cress)
9	(ashteroth, karnaim, zuzims)	(butterfly, scarce, perch)	(gem, feared, address)

	burgess-busterbrown.txt	carroll-alice.txt	chesterton-ball.txt
0	(blackbird, drummer, woodpecker)	(Lewis, carroll, 1865)	(00, 99, 98)
1	(chickadee, jenny, wren)	(accustomed, usurpation, conquest)	(01, 00, 99)
2	(cut, knife, bitten)	(adjourn, immediate, adoption)	(1000, 1997, August)
3	(drummer, woodpecker, welcome)	(adoption, energetic, remedy)	(113, 1739, University)
4	(gently, tipped, spilled)	(ancient, modern, seaography)	(1739, University, ave)
5	(grave, solemn, please)	(arithmetic, ambition, distraction)	(1971, July, 10)
6	(hawk, scrapper, Kingbird)	(brother, latin, grammar)	(1997, August, 1500)
7	(jenny, wren, redeye)	(canvas, bag, tied)	(95, 94, 93)
8	(pushed, gently, tipped)	(circle, exact, shape)	(96, 95, 94)
9	(redtail, hawk, scrapper)	(crocodile, improve, shining)	(97, 96, 95)

	chesterton-brown.txt	chesterton-thursday.txt	edgeworth-parents.txt
0	(Willie, pragmatist, alternated)	(1908, edmund, clerihew)	(58, dissolution, alien)
1	(abstraction, lesser, gem)	(abandoned, boardroom, closed)	(Ira, furor, brevis)
2	(bibber, sincerely, beg)	(argent, chevron, gules)	(ab, origine, null)
3	(campaign, swift, precision)	(barnum, freak, physically)	(affect, imitate, indiscriminate
4	(causing, stagger, swing)	(bellegarde, Baron, zumpt)	(alien, priory, 1414)
5	(chaplain, co, religionist)	(bite, slice, bread)	(appreciated, numismatic, coll
6	(characterizes, corpulent, charwoman)	(bowing, repeatedly, kissing)	(arc, centre, complement)
7	(congregation, rivet, ecstatic)	(brainless, godforsaken, doddering)	(assistant, maria, edgeworth)
8	(customary, joy, ride)	(chesterton, 1908, edmund)	(autre, mais, pa)
9	(cuttlefish, writhing, polypus)	(chevron, gules, charged)	(bohn, recent, edition)

	melville-moby_dick.txt	milton-paradise.txt	shakespeare-caesar.txt
0	(afflicted, jaundice, infirmity)	(Abarim, hesebon, horonaim)	(South, weighing, youthfull)
1	(anacharsis, clootz, deputation)	(Ammonite, worshiped, rabba)	(William, shakespeare, 1599)
2	(andrew, jackson, pebble)	(Interpret, advocate, propitiation)	(adder, craues, warie)
3	(apology, raimond, sebond)	(Lavinia, disespous, neptune)	(attendant, absent, swallow)
4	(approve, omnisciently, exhaustive)	(Perplexed, greek, cytherea)	(aule, meddle, tradesman)
5	(authorized, legislative, enactment)	(Vexed, scylla, bathing)	(billow, swimme, barke)
6	(auto, da, Fe)	(almansor, fez, sus)	(cappes, vttered, deale)
7	(availle, returne, againe)	(ascalon, Accaron, gaza)	(charme, commended, beauty)
8	(avers, Earl, leicester)	(aspramont, montalban, damasco)	(commended, beauty, vowes)
9	(balena, vero, sufficit)	(bethink, sleepy, drench)	(couchings, lowly, courtesy)

	shakespeare-hamlet.txt	shakespeare-macbeth.txt	whitman-leaves.txt
0	(Girle, vnsifted, perillous)	(Began, fresh, assault)	(Amazonia, patagonian, feejeeman)
1	(William, shakespeare, 1599)	(Bonelesse, gummes, dasht)	(Barcelona, oporto, lyon)
2	(acquire, beget, temperance)	(Bridegroome, lapt, proofe)	(Beethoven, handel, haydn)
3	(affraide, goose, quils)	(Destinie, vessel, spels)	(Berlin, constantinople, adelaide)
4	(amble, lispe, nickname)	(Gown, vppon, vnlocke)	(Bristol, edinburgh, limerick)
5	(among, minerall, mettels)	(Maggot, pyes, choughes)	(Burg, ist, unser)
6	(angry, parle, smot)	(Mortalitie, toyes, renowne)	(Gothard, hoosac, tunnel)
7	(attends, boystrous, ruine)	(South, entry, retyre)	(Kaqueta, oronoco, wabash)
8	(aygre, droppings, milke)	(Wassell, conuince, memorie)	(Kaubul, cairo, benighted)
9	(bace, hideous, crash)	(William, shakespeare, 1603)	(Lombard, hun, bohemian)

Furthermore, if the size of the sliding window is increased, then the amount of collocations decreases because there are less grams or windows with more amount of words. For instance: For example, for the sentence "introduction to data science is fun", if we use trigrams, we obtain the following grams or windows: ('introduction', 'to', 'data') ('to', 'data', 'science') ('data', 'science', 'is') ('science', 'is', 'fun')

If we use 4-grams, we obtain: ('introduction', 'to', 'data', 'science') ('to', 'data', 'science', 'is') ('data', 'science', 'is', 'fun')

And if we use 5-grams, we obtain: ('introduction', 'to', 'data', 'science', 'is') ('to', 'data', 'science', 'is', 'fun')

Therefore, the amount of co-occurrences between words within different windows decreases because the amount of windows decreases.

1.5 Use a suitable dimensionality reduction technique to make a scatterplot of the texts based on word-occurrences. Can you visually observe clusters of similar texts? Can you spot outliers?

The code for this exercise is in **1_5.py**. We concatenated the top 300 words of each book in a single matrix (18 books x 1519 words/dimensions). We used Principle Component Analysis to reduce the 1519 dimensions to 2 principal components that explained 46.6% 20.1% of variance, respectively. We plotted the PCA in Figure 1.5.
Several clusters can be observed: Shakespeare's Hamlet, Caesar, and Macbeth are clustered and nearby is Milton's Paradise Lost. They are all from 17th century England. Another very tight cluster is composed of the works by Burgess, Carroll, and the three works by Chesterson. The three authors are from late XIXth to early XXth century. Burgess and Carroll share that they are children's books.Finally, there is a cluster around the three works by Jane Austen.
Outliers seem to be the bible, Blake and Bryant, since they fall on higher values of y and do not fall into any cluster. For instance, is different than other works because it includes many numbers as high frequency words. The Bible constantly is counting years, deaths, populations, and ages. This would make it dimensions be different than other works.
Therefore, we can conclude that these two dimensions seem to be a good semantic representation of the works. Moreover, the x axis seems to be somewhat representing time as the larger numbers tend to be older works and the smaller numbers tend to be more contemporary works.

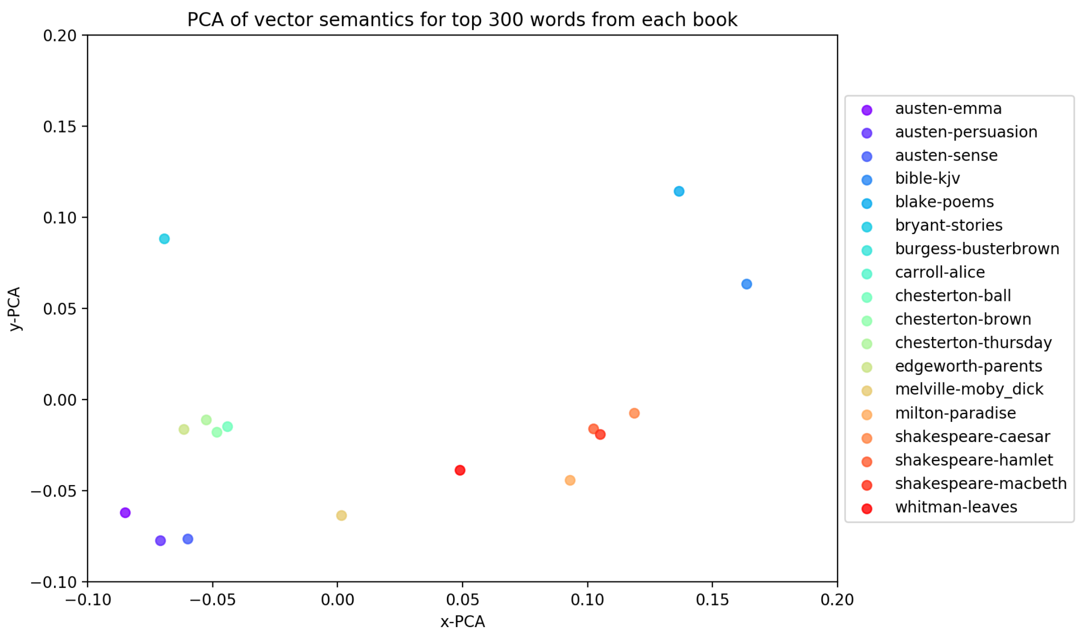

Figure 2: PCA of semantic vectors from each book scaled around to be around 0.

2 Your own search engine

2.1 TF.IDF representation of corpus

The code for this exercise can be found in **2_1_tfidf.py**.

In order to obtain the TF.IDF representation of the Gutenberg corpus, we used the **TfidfVectorizer** class from the **sklearn** package. Using it, we obtained a matrix with 18 rows (the number of documents in the corpus) and 41757 columns (the number of unique words in the corpus).

2.2 Most representatives words in texts

The code for this exercise can be found in **2_2.py**.

For this exercise, we used the TF.IDF representation of the Gutenberg corpus from 2.1. Based on it, we selected the 10 words with the highest TF.IDF score in each of the following 3 texts: **carroll-alice, melville-moby_dick, shakespeare-hamlet**. We obtained the following results accordingly:

```
alice> 0.795421681893
said> 0.36093686006
gryphon> 0.122281398803
hatter> 0.111918628608
duchess> 0.106682822993
dormouse> 0.101602688565
little> 0.0999998227005
turtle> 0.0992248952357
rabbit> 0.0796287418931
know> 0.0687498781066
()
whale> 0.526386131304
ahab> 0.312415734782
stubb> 0.179511643241
```

```
queequeg> 0.17601919882
ship> 0.153515329056
sperm> 0.149176984906
like> 0.138997769525
starbuck> 0.138300799073
whales> 0.134439047837
pequod> 0.120838576968
()
ham> 0.500613812561
haue> 0.33846019827
hor> 0.200953031964
lord> 0.183787971558
king> 0.149817682976
hamlet> 0.148550092748
laer> 0.145000569706
ophe> 0.135333865059
qu> 0.131148294545
selfe> 0.129299401586
()
```

2.3 Where-was-that function

The code for this exercise can be found in **2_3.py**. In order to use the function, you have to open the terminal and run the the script with the 2 arguments (search phrase and corpus) as strings. Example:

```
python 2_3.py 'story with the girl falling into a rabbit hole' 'gutenberg'
```

The search phrase can be any string, while the corpus can only be chosen between *'gutenberg'* and *'state_union'*.

The scheme of the algorithm can be summarized as follows:

1. Compute the TF.IDF representation of the corpus. Get a matrix of the form describes in 2.1.
2. Split the search phrase into words (lowercase).
3. For each row of the matrix(document):
   (a) Get the score for each of the words in search phrase (so get the values of multiple columns on the same line in the matrix)
   (b) The score of the search phrase for a document is the sum of the individual scores of the words composing it.
4. The row where the search phrase got the highest score belongs to the document that most likely talks about the description in the search phrase.

Additionally, we print the 3 closest documents (with scores more than 0). We present some of the results of the search phrases we tried using the Gutenberg corpus in Table 1.

Table 1: Results of the where-was-that function

Search Phrase	Best Match	Contenders
story with the girl falling into a rabbit hole	carroll-alice.txt	chesterton-brown.txt, bryant-stories.txt, edgeworth-parents.txt
story with the sailor and the whale	melville-moby_dick.txt	bryant-stories.txt, chesterton-brown.txt, edgeworth-parents.txt
the one with the chosen people	bible-kjv.txt	bryant-stories.txt, edgeworth-parents.txt, chesterton-brown.txt
the one with the rabbit	carroll-alice.txt	burgess-busterbrown.txt, bryant-stories.txt, chesterton-brown.txt
the story with the old ship captain	melville-moby_dick.txt	austen-persuasion.txt, chesterton-brown.txt, bryant-stories.txt

3 Bonus

The code for this exercise can be found in **bonus.py**.

For this part, we used the idea suggested in the exercise text of ranking the sentences in a document by their *interestingness*. Similar to what we did in 2.3, we defined *interestingness* of a sentence as the sum of the TF.IDF scores of all its words. We then normalize this sum by dividing it to the number of words in the sentence, as not to favor longer sentences.

The algorithm for summarizing a book is the following:

1. Compute TF.IDF representation of the corpus.
2. Split the document into sentences.
3. Get the into a list the sentence text, its *interestingness* score, as well as the number of words and the position inside the document.
4. Sort the list by the *interestingness* score.
5. Keep in the list only the first sentences, for which the sum of their word count is lower than 200.
6. Sort this truncated list by the position in the document, as to have a chronological order in the summary.
7. The summary is formed by concatenating these sentences.

It is important to mention that in step 5, we only keep the sentences with more than 7 words. We do this because shorter sentences, although having a higher score might actually convey less information (e.g., 'said Alice', 'Emma could not doubt', 'Ahab turned'). We selected 7 after comparing the outputs we got using different thresholds.

The summaries of each of the documents can be found in the ./**bonus**/ directory. An example of the summary we obtained for Moby-Dick by Herman Melville:

The whale is doubtless the largest animal in creation.""Spain–a great whale stranded on the shores of Europe."Captain Ahab is the Captain of this ship."well, spose him one whale eye, well, den!"THE

WHALE NO FAMOUS AUTHOR, AND WHALING NO FAMOUS CHRONICLER?THE WHALE NEVER FIGURED IN ANY GRAND IMPOSING WAY?good people all,–the Greenland whale is deposed,–the great sperm whale now reigneth!(RAZOR-BACK).–Of this whale little is known but his name."Do ye know the white whale then, Tash?"I tell you, the sperm whale will stand no non-sense.He cut it; and the whale was free.whale eat him, 'stead of him eat whale.Like those mystic rocks, too, the mystic-marked whale remains undecipherable.from which not the mightiest whale is free.Of the grand order of folio leviathans, the Sperm Whale and the Right Whale are by far the most noteworthy.Again, the Right Whale has two external spout-holes, the Sperm Whale only one.Physiognomically regarded, the Sperm Whale is an anomalous creature.Has the Sperm Whale ever written a book, spoken a speech?CHAPTER 109 Ahab and Starbuck in the Cabin.He declared that a whale must be near.