Report: Assignment 1 - Corpus Analysis and Sentence Embeddings

CSI 5386 - Natural Language Processing

Fall 2022

School of Electrical Engineering and Computer Science

University of Ottawa

Course teacher: Diana Inkpen

Submission date: 11 October 2022

**Group 1**

Adrien Heymans - 300092260

Mitchell Chatterjee - (uOttawa: 8598617 / Carleton: 101141206)

Bhavika Sewpal - 300089940

# Division of tasks

|  |  |
| --- | --- |
| Student | Task |
| Adrien Heymans | Part 1 - Tokenization and Counting |
| Mitchell Chatterjee | Part 2 - Sentence Embeddings |
| Bhavika Sewpal | Writing the report |

# Part 1 - Corpus processing (legal text): tokenization and word counting

**Merging of documents**

We first started by merging all the text files in the full\_contract\_text directory into a single text file.

**Python packages and modules used**

We used the Natural Language Toolkit package (nltk) to analyze the corpus and the pandas package to store the results in a table.

In the nltk package, we used the following subpackages/submodules:

1. nltk.tokenize - To split words and punctuation in a string in various ways
2. nltk.probability - To make calculations about the frequency of tokens
3. nltk.util - To return bigrams generated from a sequence of tokens

**Methods**

Throughout part 1, we use tokens that consist of lower-case letters.

For the first part, we split the corpus into tokens using the TreeBank tokenizer from the nltk.tokenize package. This tokenizer performs the following steps:

1. split standard contractions, e.g. don't -> do n't and they'll -> they 'll
2. treat most punctuation characters as separate tokens
3. split off commas and single quotes, when followed by whitespace
4. separate periods that appear at the end of line

As a result, the output of the tokenizer contained words, numbers, punctuation symbols and other characters as well (\*, {, (, ), \_ }).

To analyze the corpus without any punctuations or symbols, we used the RegexpTokenizer from the nltk.tokenize package which allows the tokens to be matched to a regular expression. In our case, it is ‘r\w+’. This regular expression matches with letters, numbers, and underscores. Upon further inspection, it was observed that there was a high occurrence of tokens consisting of the page division character ‘\_\_’, which is basically a sequence of underscores. These were consequently removed from the list of tokens.

To analyze the corpus without any stopwords, we first had to compile a list of stopwords:

*I, me, my, myself, we, our, ours, ourselves, you, you're, you've, you'll, you'd, your, yours, yourself, yourselves, he, him, his, himself, she, she's, her, hers, herself, it, it's, its, itself, they, them, their, theirs, themselves, what, which, who, whom, this, that, that'll, these, those, am, is, are, was, were, be, been, being, have, has, had, having, do, does, did, doing, a, an, the, and, but, if, or, because, as, until, while, of, at, by, for, with, about, against, between, into, through, during, before, after, above, below, to, from, up, down, in, out, on, off, over, under, again, further, then, once, here, there, when, where, why, how, all, any, both, each, few, more, most, other, some, such, no, nor, not, only, own, same, so, than, too, very, s, t, can, will, just, don, don't, should, should've, now, d, ll, m, o, re, ve, y, ain, aren, aren't, couldn, couldn't, didn, didn't, doesn, doesn't, hadn, hadn't, hasn, hasn't, haven, haven't, isn, isn't, ma, mightn, mightn't, mustn, mustn't, needn, needn't, shan, shan't, shouldn, shouldn't, wasn, wasn't, weren, weren't, won, won't, wouldn, wouldn't*

Then, we proceeded to remove the tokens that were stopwords themselves. After having filtered all the stopwords in the tokens, we were left with tokens that were significant and more relevant to the corpus.

Finally, to get the most frequently occurring pairs of words (i.e bigrams), we used the ‘ngrams’ function from the nltk.util package to generate all pairs of consecutive tokens. We then calculated the frequency of occurrence of those pairs and determined which pairs occurred more often.

**Questions:**

1. Submit a file output.txt with the tokenizer’s output for the whole corpus. Include in your report the first 20 lines from output.txt.

*'exhibit', '10.6', 'distributor', 'agreement', 'this', 'distributor', 'agreement', '(', 'the', '``', 'agreement', "''", ')', 'is', 'made', 'by', 'and', 'between', 'electric', 'city', 'corp.', ',', 'a', 'delaware', 'corporation', '(', '``', 'company', "''", ')', 'and', 'electric', 'city', 'of', 'illinois', 'llc', '(', '``', 'distributor', "''", ')', 'this', '7th', 'day', 'of', 'september', ',', '1999', '.', 'recitals', 'a.', 'the', 'company', "'s", 'business.', 'the', 'company', 'is', 'presently', 'engaged', 'in', 'the', 'business', 'of', 'selling', 'an', 'energy', 'efficiency', 'device', ',', 'which', 'is', 'referred', 'to', 'as', 'an', '``', 'energy', 'saver', "''", 'which', 'may', 'be', 'improved', 'or', 'otherwise', 'changed', 'from', 'its', 'present', 'composition', '(', 'the', '``', 'products', "''", ')', '.', 'the', 'company', 'may', 'engage', 'in', 'the', 'business', 'of', 'selling', 'other', 'products', 'or', 'other', 'devices', 'other', 'than', 'the', 'products', ',', 'which', 'will', 'be', 'considered', 'products', 'if', 'distributor', 'exercises', 'its', 'options', 'pursuant', 'to', 'section', '7', 'hereof', '.', 'b.', 'representations.', 'as', 'an', 'inducement', 'to', 'the', 'company', 'to', 'enter', 'into', 'this', 'agreement', ',', 'the', 'distributor', 'has', 'represented', 'that', 'it', 'has', 'or', 'will', 'have', 'the', 'facilities', ',', 'personnel', ',', 'and', 'financial', 'capability', 'to', 'promote', 'the', 'sale', 'and', 'use', 'of', 'products.', 'as', 'an', 'inducement', 'to', 'distributor', 'to', 'enter', 'into', 'this', 'agreement', 'the', 'company', 'has', 'represented', 'that', 'it', 'has', 'the', 'facilities', ',', 'personnel', 'and', 'financial', 'capability', 'to', 'have', 'the', 'products', 'produced', 'and', 'supplied', 'as', 'needed', 'pursuant', 'to', 'the', 'terms', 'hereof', '.', 'c.', 'the', 'distributor', "'s", 'objectives.', 'the', 'distributor', 'desires', 'to', 'become', 'a', 'distributor', 'for', 'the', 'company', 'and', 'to', 'develop', 'demand', 'for', 'and', 'sell', 'and', 'distribute', 'products', 'solely', 'for', 'the', 'use', 'within', 'the', 'state', 'of', 'illinois', ',', 'including', 'but', 'not', 'limited', 'to', 'public', 'and', 'private', 'entities', ',', 'institutions', ',', 'corporations', ',', 'public', 'schools', ',', 'park', 'districts', ',', 'corrections', 'facilities', ',', 'airports', ',*

1. How many tokens did you find in the corpus? How many types (unique tokens) did you have? What is the **type/token ratio** for the corpus? The type/token ratio is defined as the number of types divided by the number of tokens.

Type = 52257

Token = 4667146

type/token ratio = 0.011197

1. For each token, print the token and its frequency in a file called tokens.txt (from the most frequent to the least frequent) and include the first 20 lines in your report.

*('the', 257125)(',', 240576)('of', 156117)('to', 129868)('and', 129053)('or', 105155)('in', 79929)(')', 78092)('(', 75436)('any', 62194)('--', 58711)('a', 50381)('shall', 48791)('by', 44309)('this', 39984)('be', 39658)('agreement', 39297)('for', 38721)('.', 38156)('such', 36162)('with', 33871)('as', 32907)('party', 31674)('that', 27652)('other', 26302)("''", 22246)('``', 22173)('is', 21963)('all', 21927)('not', 21620)(']', 21512)('[', 21497)(':', 21110)('its', 19773)('will', 19189)("'s", 19092)(';', 17490)('on', 17000)('\*\*\*', 14366)('under', 14165)('may', 13547)('section', 13217)('which', 13198)('at', 13198)('from', 12864)('parties', 12537)('company', 11998)('each', 11473)('if', 11210)('have', 11197)('information', 10456)('an', 10383)('product', 10295)('including', 9917)('are', 9687)('date', 9659)('has', 9287)('time', 8840)('use', 8656)('provided', 8165)('b', 8098)('it', 7872)('rights', 7671)('products', 7631)('applicable', 7422)('services', 7363)('no', 7286)('business', 7016)('set', 6979)('upon', 6847)('written', 6752)('right', 6626)('confidential', 6557)('terms', 6551)('without', 6421)('respect', 6402)('notice', 6384)('forth', 6358)('term', 6173)('been', 5803)('-', 5703)('i', 5650)('within', 5630)('...', 5630)('prior', 5548)('subject', 5546)('third', 5500)('termination', 5450)('event', 5443)('obligations', 5349)('pursuant', 5244)('means', 5223)('provide', 5196)('than', 5112)('required', 5097)('otherwise', 5090)('customer', 5001)('reasonable', 4895)('days', 4846)('ii', 4744)('after', 4731)('material', 4726)('during', 4717)('accordance', 4686)('property', 4513)('period', 4478)('except', 4383)('agreement.', 4321)('effective', 4234)('following', 4215)('1', 4167)('service', 4162)('c', 4158)('source', 4077)('connection', 4064)('one', 4006)('you', 4001)('law', 3939)('\*', 3923)('either', 3885)('made', 3857)('distributor', 3852)('license', 3845)('payment', 3803)('but', 3782)('and/or', 3768)('order', 3755)('their', 3726)('agrees', 3687)('licensed', 3684)('exhibit', 3640)('between', 3636)('agent', 3631)('software', 3612)('inc.', 3599)('breach', 3551)('agree', 3478)('securities', 3439)('affiliates', 3429)('development', 3385)('hereunder', 3382)('writing', 3369)('costs', 3313)('laws', 3303)('act', 3270)('extent', 3258)('provisions', 3248)('2', 3215)('materials', 3198)('performance', 3115)('necessary', 3090)('person', 3079)('purchase', 3050)('herein', 3045)('&', 3045)*

1. How many tokens appeared only once in the corpus?

22632 tokens

1. From the list of tokens, extract only words, by excluding punctuation and other symbols, if any. Please pay attention to end of sentence dot (full stops). How many words did you find? List the top 20 most frequent words in your report, with their frequencies. What is **the type/token ratio** when you use only words (called lexical diversity)?

Type = 28132

Tokens = 3920681

type/token ratio = 0.006852513363643339

1. ('the', 257278)
2. ('of', 156455)
3. ('and', 132873)
4. ('to', 130141)
5. ('or', 108944)
6. ('in', 80359)
7. ('any', 62243)
8. ('a', 53540)
9. ('shall', 48794)
10. ('by', 44533)
11. ('agreement', 43651)
12. ('this', 39998)
13. ('be', 39709)
14. ('for', 38790)
15. ('such', 36173)
16. ('with', 33886)
17. ('party', 33277)
18. ('as', 32927)
19. ('that', 27654)
20. ('other', 26409)
21. From the list of words, exclude stopwords. List the top 20 most frequent words and their frequencies in your report. You can use [this list](http://www.site.uottawa.ca/~diana/csi5180/StopWords) of stopwords (or any other that you consider adequate). Also compute **the type/token ratio** when you use only word tokens without stopwords (called lexical density)?

Type = 27996

Token = 2219648

type/token = 0.012612810679891586

1. ('shall', 48794)
2. ('agreement', 43651)
3. ('party', 33277)
4. ('may', 13597)
5. ('parties', 13523)
6. ('section', 13350)
7. ('company', 12636)
8. ('information', 10941)
9. ('product', 10920)
10. ('date', 10176)
11. ('including', 9924)
12. ('time', 9453)
13. ('b', 9235)
14. ('use', 8912)
15. ('provided', 8229)
16. ('products', 8201)
17. ('rights', 8067)
18. ('services', 7888)
19. ('applicable', 7540)
20. ('business', 7342)
21. Compute all the pairs of two consecutive words (bigrams) (excluding stopwords and punctuation). List the most frequent 20 pairs and their frequencies in your report.
22. ('set', 'forth'), 6151)
23. (('third', 'party'), 4600)
24. (('agreement', 'shall'), 3664)
25. (('confidential', 'information'), 3598)
26. (('party', 'shall'), 3280)
27. (('intellectual', 'property'), 2933)
28. (('effective', 'date'), 2822)
29. (('either', 'party'), 2448)
30. (('written', 'notice'), 2383)
31. (('terms', 'conditions'), 2087)
32. (('prior', 'written'), 1811)
33. (('without', 'limitation'), 1779)
34. (('forth', 'section'), 1743)
35. (('time', 'time'), 1696)
36. (('shall', 'deemed'), 1685)
37. (('term', 'agreement'), 1653)
38. (('shall', 'mean'), 1635)
39. (('including', 'without'), 1553)
40. (('party', 'may'), 1547)
41. (('confidential', 'treatment'), 1537)

|  |  |
| --- | --- |
| # of tokens b) | 4667146 |
| # of types b) | 52257 |
| type/token ratio b) | 0.011196778502322405 |
| Tokens appeared only once d) | 22632 |
| # of words (excluding punctuation) (e) | 28132 |
| type/token ratio (excluding punctuation) (e) | 0.007175284 |
| List the top 3 most frequent words and their frequencies (e) | ('the', 257278) |
| ('of', 156455) |
| ('and', 132873) |
| type/token ratio (excluding punctuation and stopwords) (f) | 0.012612811 |
| List the top 3 most frequent words and their frequencies (excluding stopwords) (f) | ('shall', 48794) |
| ('agreement', 43651) |
| ('party', 33277) |
| List the top 3 most frequent bigrams  and their frequencies (g) | (('set', 'forth'), 6151) |
| (('third', 'party'), 4600) |
| (('agreement', 'shall'), 3664) |

**Discussion of results**

For part a), when we display the output of the tokenizer for the whole corpus, we get a variety of tokens which include words, numbers, punctuation symbols, square and round brackets.

For the lexical density (the type/token ratio):

1. The type/token ratio for the output of the tokenizer (0.012) is higher than the ratio obtained after the removal of punctuation and symbols (0.0078) because the total number of unique tokens (i.e the type) decreased significantly from 52257 to 32149. This shows that punctuation and other symbols made up a significant portion of the initial unique number of tokens. Even though the number of tokens decreased from 4667146 to 3920681 due to the removal of punctuation, numbers and symbols, this decrease was offset by the decrease in the type of tokens, causing the lexical density to decrease significantly.
2. The type/token ratio obtained after the removal of stopwords, punctuation and symbols is significantly higher than the ratio obtained from the removal of punctuation and symbols only. There was only a small decrease in the type of tokens (from 28132 to 27996). This means that only 136 stop words were removed from the corpus. However, stop words occur with high frequencies. Even though only 136 stop words were removed from the corpus, it was enough to cause a reduction in the total number of tokens from 3920681 to 2219648. This decrease in the total number of tokens offset the decrease in the type, which caused the lexical density to increase significantly from 0.0078 to 0.013.

From part e), we can observe that after the removal of punctuation and symbols, the 20 most frequent words consisted mostly of stop words: the, of, and, to, or, in, any, a, by, this, be, for, such, with, as, that, other. Only 3 words that were not stopwords made it to the list: shall, party and agreement.

From part f), we can observe that after the removal of stopwords, punctuations and symbols, the 20 most frequent words contained a significant portion of words that were related to the vocabulary used in contracts. We got the following results: shall, agreement, party, may, section, parties, information, company, product, date, including, time, use and provided).

# Part 2: Evaluation of pre-trained sentence embedding models

**Merging of documents**

We first start by merging the different input text files into a single file.

**Python packages and modules used**

SBERT Models

To get the pretrained embeddings, we used the sentence\_transformer package. This package provides an easy method to compute dense vector representations for sentences. We used the following SBERT models:

1. all-MiniLM-L6-v2
2. All-mpnet-base-v2
3. Paraphrase-mpnet-base-v2
4. Distiluse-base-multilingual-cased-v2

We also use the sentence\_transformer.util.cos\_sim function to compute the cosine similarity between word embeddings.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Training Data | Suitable Score Function | Dimensions of embeddings |
| all-MiniLM-L6-v2 | All-round model tuned for many use-cases. Trained on a large and diverse dataset of over 1 billion training pairs. | Util.cos\_sim | 384 |
| All-mpnet-base-v2 | All-round model tuned for many use-cases. Trained on a large and diverse dataset of over 1 billion training pairs. | Util.cos\_sim | 768 |
| Paraphrase-mpnet-base-v2 | Trained on a large and diverse dataset of over 1 billion training pairs. | Util.cos\_sim | 768 |
| Distiluse-base-multilingual-cased-v2 | Multi-Lingual model of Universal Sentence Encoder for 50 languages. | Util.cos\_sim | 512 |

Doc2Vec Model

Doc2Vec is based on word2Vec and it is used to create numerical representations of documents.

To use the doc2Vec model, we used the gensim.models.doc2Vec.Doc2Vec class in the gensim package. This package contains useful functions to build the vocabulary of the model and train it.

The doc2Vec model was trained using the “Text8” dataset, which contains all the text available on Wikipedia pages.

When building the model, the dimensionality of the feature vectors was set to 40.

The min\_count argument was set to 2, which meant that the model would ignore all words with total frequency less than 2.

The number of epochs was set to 30, which meant that during the training phase, there would be 30 iterations over the corpus.

Furthermore, we also required the word\_tokenize function from the nltk package to tokenize the input sentences for the doc2Vec model.

**Methods**

In the input files, each line contains 2 sentences. The goal of this task is to compute the similarity between these two sentences. Once all the input files have been merged in a single document, these two sentences are extracted for every single line. Then, the sentence embeddings are calculated for each of these 2 sentences. These sentence embeddings are essentially vectors that capture the entire semantic information of sentences. We then compute the cosine similarity of the embeddings to determine the degree of similarity between the two sentences. It is important to note that the util.cos\_sim function only yields a value between -1 and 1. To return a value between 0 and 5, we multiply the absolute value of the output of the util.cos\_sim function by 5 and round it off to the nearest integer.

We repeat the above steps for all the 4 SBERT models.

We follow a similar process for the doc2Vec model except that we tokenize the input instead of using sentence embeddings.

Finally, once we obtain the similarity score (on a scale of 0 to 5) for every model, we compute the Pearson correlation between the score obtained by the model and the expected solution provided in the gold standard files.

However, it turns out that the gold standard files contained the expected similarity score for only certain pairs of sentences (only certain lines). Before computing the Pearson correlation for each model, we had to make sure that we would only be comparing the available similarity scores in the gold standard files with the corresponding similarity scores generated by the different models.

The correlation is computed using the correlation-noconfidence.pl script.

To run the script, use the following command:

./correlation-noconfidence.pl <gold standard> <output from model>

The table below gives the Pearson Correlation for each model:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Model | SE1  all-MiniLM-L6-v2 | SE2  All-mpnet-base-v2 | SE3  Paraphrase-mpnet-base-v2 | SE4  Distiluse-base-multilingual-cased-v2 | Doc2Vec | Best Score |
| STS Test Data | 0.71288 | 0.70574 | 0.73025 | 0.7415 | 0.75144 | 0.75144 |

**Discussion of results**

From the table, we can conclude that Doc2Vec has the highest correlation score. However, all of the models have a correlation greater than 0.71. Therefore, there is not a significant difference in correlation between the SBERT models and the Doc2Vec model. In particular, the Distiluse-base-multilingual-cased-v2 SBERT model has a correlation score that is very close to that of the Doc2Vec model.

**Google Colab**

It is faster to run the notebook for part 2 on google colab using the gpu.

You must make sure to upload the original data text files as a zip file.

Here is the link to the github page: [Assignment 1](https://github.com/adrien-heymans/CSI5386)