CORPUS ANALYSIS AND SENTENCE EMBEDDINGS

GROUP 26: TEAM MEMBERS	Email ID
Adorin Kripanand Lucas	akrip010@uottawa.ca
Daksh Chaudhary	dchau012@uottawa.ca
Aishwarya Manapuram	amana022@uottawa.ca

Division of Tasks:

Both part 1 and part 2 were divided amongst the team mates equally. The problem statement and ways to solve the tasks were discussed and understood together over MS Teams. The errors in the code for part 1, issues and changes in scripts for part 2 were resolved by all of us together after referring multiple resources online. For each of the word embeddings out of 5, it was also equally divided and evaluated between all the team members.

Task done by Daksh:
Part 1: b, c, d, e, f, g
Part 2: Doc2vec

Tasks done by Adorin:
Part 1: a, b, c
Part 2: SentenceBERT, all-mpnet-base-v2

Tasks done by Aishwarya:
Part 1: a, b, c
Part 2: InferSent (GloVe), Universal Sentence Encoder

The evaluation score using the similarity values obtained vs the standard values for Pearson correlation was done by all the team mates together. The results and documentation of the whole assignment was discussed and written together.

Part 1: Corpus processing: tokenization and word counting

1. Concatenated all the text files in CUAD_v1 into one text file.
2. Used the nltk word_tokenize() method to tokenize the concatenated text file.
3. Printed the total number of tokens in the corpus. Performed Point of Speech tagging (pos_tag) and lemmatization for linguistic pre-processing so that every word can be properly accounted.
4. Calculated the number of unique tokens using the set data structure and dictionary data structure.
5. Calculated the type/token ratio.
6. Printed the frequency of each token.
7. Counted the number of tokens that appeared only once.

8. Removed URLs, digits, and punctuations using regular expression and listed the top 20 frequent words and printed the type/token ratio using only words.
9. Removed stop words and printed the top 20 frequent words, type/token ratio. (Stopwords are the words in any language which does not add much meaning to a sentence. Example: ‘a’, ’the’, ‘is’, etc.)
10. Computed all the bigrams and listed the frequent 20 pairs and their frequencies.

NOTE: The entire code for the task 1 is attached in Task1.ipynb

# of tokens (b)	4789324
# of types (b)	42674
type/token ratio (b)	42674/4789324 (or) 0.008910234513263249
tokens appeared only once (d)	19328
# of words (excluding punctuation) (e)	3891726
type/token ratio (excluding punctuation) (e)	31031/3891726 (or) 0.007973582929527927
List the top 3 most frequent words and their frequencies (e)	‘the’ - 257136
	‘of’ - 156122
	‘to’ - 129875
type/token ratio (excluding punctuation and stopwords) (f)	30474/1868715 (or) 0.016307462614684423
List the top 3 most frequent words and their frequencies (excluding stopwords) (f)	‘party’ – 46335
	‘agreement’ – 45737
	‘product’ - 18925
List the top 3 most frequent bigrams and their frequencies (g)	‘confidential’, ’information’ – 3601
	‘intellectual’, ‘property’ – 2927
	‘term’, ‘agreement’ - 2911

a) First 20 lines from output.txt:

```
CO-BRANDING
AND
ADVERTISING
AGREEMENT
THIS
CO-BRANDING
AND
ADVERTISING
AGREEMENT
(
the
``
Agreement
''
)
```

is
made
as
of
June

b) Number of tokens in the corpus: 4789324
Number of unique tokens: 42674
Type/token ratio: 42674 / 4789324   or 0.008910234513263249

c) First 20 lines from tokens.txt:

```
Token: the      Frequency:257132
Token: ,        Frequency:240576
Token: of       Frequency:156122
Token: to       Frequency:129875
Token: and      Frequency:129054
Token: .        Frequency:117447
Token: or       Frequency:105155
Token: be       Frequency:80380
Token: in       Frequency:79933
Token: )        Frequency:78092
Token: (        Frequency:75436
Token: *        Frequency:67765
Token: any      Frequency:62236
Token: --       Frequency:58712
Token: a        Frequency:51002
Token: shall    Frequency:48794
Token: party    Frequency:46335
Token: agreement        Frequency:45735
Token: by       Frequency:44310
Token: this     Frequency:39986
```

d) Number of tokens that appeared only once in the corpus: 19328

e) Top 20 word after excluding punctuations and other symbols:

```
('the', 257136),
('of', 156122),
 ('to', 129875),
 ('and', 129054),
 ('or', 105156),
 ('be', 80380),
 ('in', 79934),
 ('any', 62236),
 ('a', 51002),
 ('shall', 48794),
 ('party', 46335),
 ('agreement', 45737),
 ('by', 44310),
 ('this', 39986),
 ('for', 38724),
 ('such', 36172),
 ('with', 33883),
 ('as', 32909),
 ('that', 27654),
 ('other', 26395)
```

Number of tokens after removing punctuations and digits: 31031

Lexical diversity: 31031/3891726 or 0.007973582929527927

f) 20 most frequent words after excluding stopwords:

```
('party', 46335),
('agreement', 45737),
('product', 18925),
('right', 14725),
('section', 14216),
('term', 13292),
('company', 12905),
('service', 12293),
('date', 11180),
('information', 10920),
('agree', 8946),
('write', 8614),
('material', 8145),
('law', 8050),
('notice', 7815),
('obligation', 7802),
('applicable', 7533),
('business', 7512),
('set', 7289),
('respect', 7055)
```

Lexical diversity: 30474/1868715 or 0.016307462614684423

g) **Frequent 20 pairs of bigrams:**

```
(('confidential', 'information'), 3601),
(('intellectual', 'property'), 2927),
(('term', 'agreement'), 2911),
(('effective', 'date'), 2846),
(('…', '…'), 2466),
(('write', 'notice'), 2413),
(('agreement', 'party'), 2380),
(('term', 'condition'), 2190),
```

(('applicable', 'law'), 2082),
(('party', 'party'), 1967),
(('party', 'agree'), 1962),
(('set', 'section'), 1914),
(('prior', 'write'), 1814),
(('provision', 'agreement'), 1671),
(('confidential', 'treatment'), 1535),
(('receive', 'party'), 1515),
(('termination', 'agreement'), 1436),
(('security', 'exchange'), 1423),
(('disclose', 'party'), 1422),
(('pursuant', 'section'), 1416)

Issues faced:

1. word_tokenize did not handle the multi-word tokenizers and word-Internal punctuation properly.

2. Tested Stanford Dependency Parser, but did not go with it as we need to install Jars and files for this method. It was giving issues to the corrector.

3. Stanford MultiWord Tokenizer Expansion is not available for English.

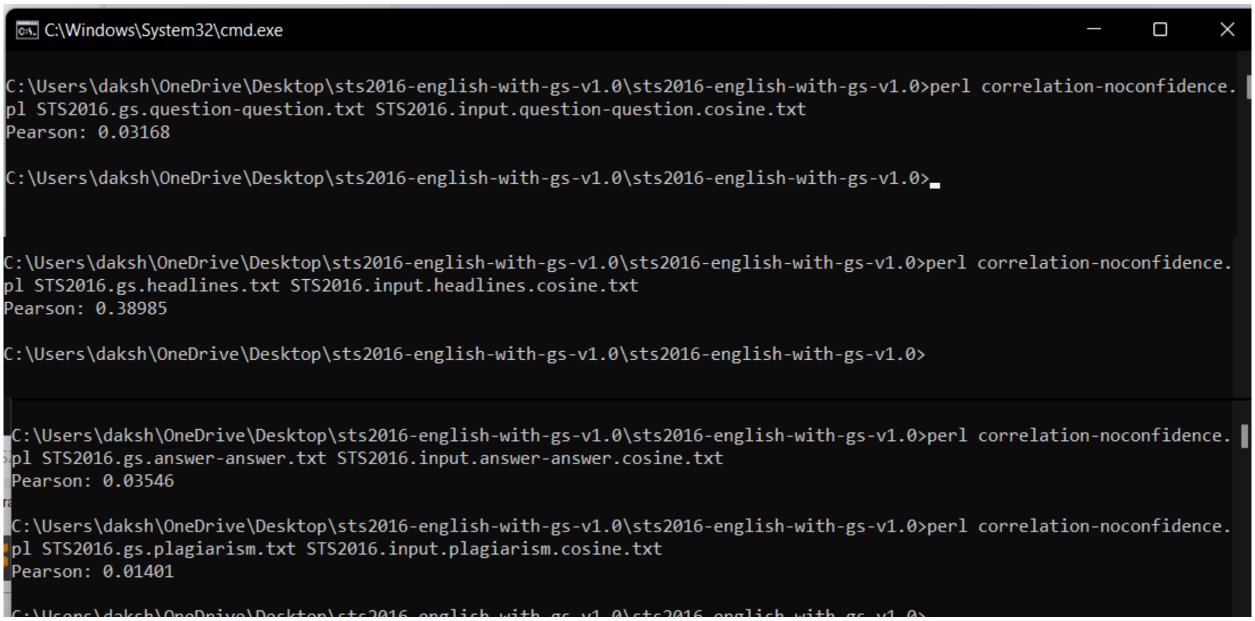
4. That lemmatizer was not giving the correct result without pos tagging. Hence performed Point of Speech tagging (pos_tag) before lemmatization for getting correct output.

5. while removing numbers and punctuations there were issues because of irregularities in data. So, dates, words starting with numbers and the digits themselves needed to be treated separately (example: 2themart.com). We did not use a regex to remove all tokens starting with a digit.

Part 2: Evaluation of pre-trained sentence embedding models

5 text files (STS2016.input.answer-answer.txt, STS2016.input.headlines.txt, STS2016.input.plagiarism.txt, STS2016.input.postediting.txt, STS2016.input.question-question.txt) are taken into consideration for Sentence Embedding Evaluations. We have selected below 5 models

i. **Doc2vec:** Here we used Gensim Python library for document indexing and similarity retrieval.
resources referred:
https://radimrehurek.com/gensim/auto_examples/tutorials/run_doc2vec_lee.html#sphx-glr-auto-examples-tutorials-run-doc2vec-lee-py,
https://stackoverflow.com/questions/53503049/measure-similarity-between-two-documents-using-doc2vec,
https://www.tutorialspoint.com/gensim/gensim_doc2vec_model.htm

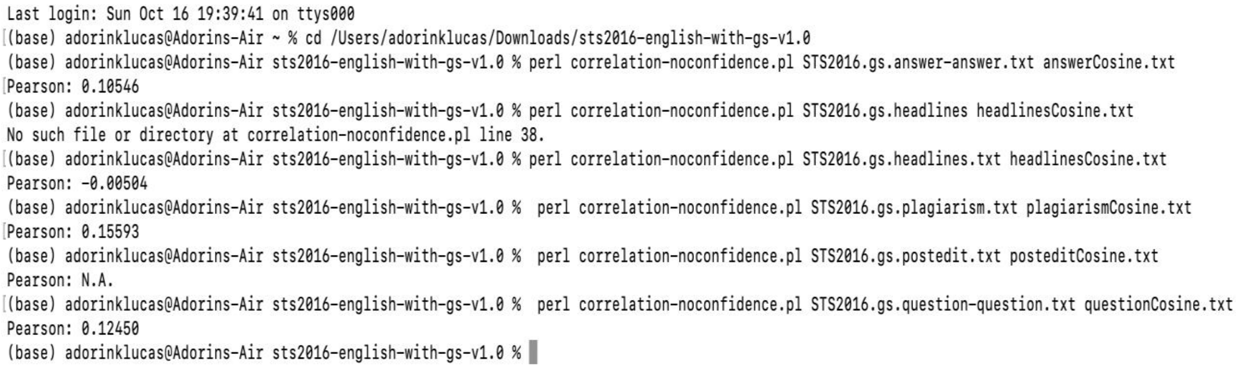


```
C:\Users\daksh\OneDrive\Desktop\sts2016-english-with-gs-v1.0\sts2016-english-with-gs-v1.0>perl correlation-noconfidence.
pl STS2016.gs.question-question.txt STS2016.input.question-question.cosine.txt
Pearson: 0.03168

C:\Users\daksh\OneDrive\Desktop\sts2016-english-with-gs-v1.0\sts2016-english-with-gs-v1.0>
```

```
C:\Users\daksh\OneDrive\Desktop\sts2016-english-with-gs-v1.0\sts2016-english-with-gs-v1.0>perl correlation-noconfidence.
pl STS2016.gs.headlines.txt STS2016.input.headlines.cosine.txt
Pearson: 0.38985

C:\Users\daksh\OneDrive\Desktop\sts2016-english-with-gs-v1.0\sts2016-english-with-gs-v1.0>
```

```
C:\Users\daksh\OneDrive\Desktop\sts2016-english-with-gs-v1.0\sts2016-english-with-gs-v1.0>perl correlation-noconfidence.
pl STS2016.gs.answer-answer.txt STS2016.input.answer-answer.cosine.txt
Pearson: 0.03546

C:\Users\daksh\OneDrive\Desktop\sts2016-english-with-gs-v1.0\sts2016-english-with-gs-v1.0>perl correlation-noconfidence.
pl STS2016.gs.plagiarism.txt STS2016.input.plagiarism.cosine.txt
Pearson: 0.01401

C:\Users\daksh\OneDrive\Desktop\sts2016-english-with-gs-v1.0\sts2016-english-with-gs-v1.0>
```

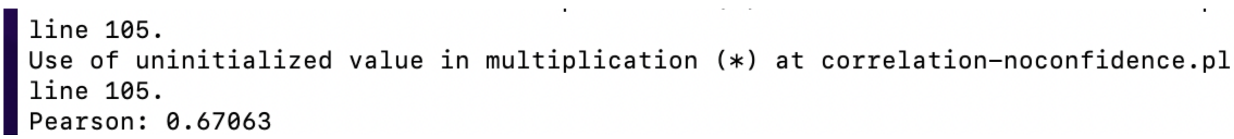
ii. **SentenceBERT-** We use sentence-transformers framework in python to compute dense vector representations for sentences. Using pre-trained BERT model, we'll encode sentences and find the cosine similarity values.
resources referred: https://www.sbert.net, https://www.analyticsvidhya.com/blog/2020/08/top-4-sentence-embedding-techniques-using-python/



```
Last login: Sun Oct 16 19:39:41 on ttys000
(base) adorinklucas@Adorins-Air ~ % cd /Users/adorinklucas/Downloads/sts2016-english-with-gs-v1.0
(base) adorinklucas@Adorins-Air sts2016-english-with-gs-v1.0 % perl correlation-noconfidence.pl STS2016.gs.answer-answer.txt answerCosine.txt
Pearson: 0.10546
(base) adorinklucas@Adorins-Air sts2016-english-with-gs-v1.0 % perl correlation-noconfidence.pl STS2016.gs.headlines headlinesCosine.txt
No such file or directory at correlation-noconfidence.pl line 38.
(base) adorinklucas@Adorins-Air sts2016-english-with-gs-v1.0 % perl correlation-noconfidence.pl STS2016.gs.headlines.txt headlinesCosine.txt
Pearson: -0.00504
(base) adorinklucas@Adorins-Air sts2016-english-with-gs-v1.0 %  perl correlation-noconfidence.pl STS2016.gs.plagiarism.txt plagiarismCosine.txt
Pearson: 0.15593
(base) adorinklucas@Adorins-Air sts2016-english-with-gs-v1.0 %  perl correlation-noconfidence.pl STS2016.gs.postedit.txt posteditCosine.txt
Pearson: N.A.
(base) adorinklucas@Adorins-Air sts2016-english-with-gs-v1.0 %  perl correlation-noconfidence.pl STS2016.gs.question-question.txt questionCosine.txt
Pearson: 0.12450
(base) adorinklucas@Adorins-Air sts2016-english-with-gs-v1.0 %
```

iii. **InferSent:** InferSent is a supervised sentence embedding technique. There are 2 versions of InferSent. Version 1 uses GLovE while version 2 uses fastText vectors. We used InferSent Version 1 (GLovE) model and pre-trained GLoVe word vectors and build the vocabulary for our input file.
resources referred: https://nlp.stanford.edu/projects/glove/, https://github.com/facebookresearch/InferSent

iv. Universal Sentence Encoder
resources referred: https://www.analyticsvidhya.com/blog/2020/08/top-4-sentence-embedding-techniques-using-python/



```
line 105.
Use of uninitialized value in multiplication (*) at correlation-noconfidence.pl
line 105.
Pearson: 0.67063
```

v. **all-mpnet-base-v2**
resources referred: https://huggingface.co/sentence-transformers/all-mpnet-base-v2

Sentence Similarity Results:

Datasets	Doc2vec	SBERT	InferSent	Universal Sentence Enoder	all-mpnet-base-v2	Best Score
STS2016.input.answer-answer.txt	0.03546	0.10546	0.36618	0.52688	0.54591	0.54591
STS2016.input.headlines.txt	0.38985	-0.00504	0.38099	0.67063	0.36420	0.67063
STS2016.input.plagiarism.txt	0.01401	0.15593	0.39360	N.A.	0.05664	0.39360
STS2016.input.postediting.txt	-0.10898	N.A.	0.14023	0.74557	0.37308	0.74557
STS2016.input.question-question.txt	0.03168	0.12450	0.19691	0.63687	0.24203	0.63687

Issues:

We got Pearson correlation values for some Models with some Datasets as N.A.