CSI5386: Natural Language Processing

Assignment 1: Corpus analysis and Sentence Embeddings

Group Members (Group 14): -

- Akshat Khare (300342170)
- Siddhant Tiwari (300294512)
- Yash Keswani (300375526)

Task Distribution

The tasks of Assignment 1 were equally distributed among three members of the group in the following manner:

Name	Task
Akshat Khare	1. Concatenating all text files to form the corpus. 2. Implement solution and write report for part 1 A to part 1 C. 3. Implement SBERT and write necessary sections of report for part 2.
Siddhant Tiwari	1. Implement solution and write report for part 1 G. 2. Implement Universal Sentence Encoder, SIMCSE, and Jina as well as write necessary sections of report for part 2. 3. Evaluate Pearson correlation for the models and construct code pipeline for submission along with README files.
Yash Keswani	1. Perform punctuation and stopword removal for part 1. 2. Implement solution and write report for part 1 D to part 1 F. 3. Implement InferSent and write necessary sections for part 2.

PART 1 CORPUS ANALYSIS

a) 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.

Our zip file contains an output.txt made according to the requirements.
The first 20 lines from this file are as follows:

CO-BRANDING
AND
ADVERTISING
AGREEMENT
THIS
CO-BRANDING
AND
ADVERTISING
AGREEMENT
(
the
``
Agreement
''
)
is
made
as
of
June

b) 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.

Answer: Total number of tokens in the corpus: 4746517
Total number of unique tokens: 58850
Type/token ratio: 0.0123985650952056

c) 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.

Top 20 most frequent tokens are listed below:

Tokens	**Frequency**
,	240576
the	239942
of	151491
to	127052
and	125198
.	117504
or	102674
)	78092
(	75436
in	73940
any	58849
--	58712
shall	48424
a	46205
by	41853
be	39157
Agreement	37006
for	35430
this	35215
such	34814

d) How many tokens appeared only once in the corpus?

Answer: The total number of tokens that appeared only once were: 24320

e) 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)?

Answer: Total Number of tokens after removing punctuation: 3921917

Top 20 most frequent words are listed below:

Tokens	Frequency
The	239999
of	151825
and	129000
to	127312
or	106445
in	74273
any	58853
shall	48425
a	46739
by	42050
be	39166
Agreement	37036
for	35481
this	35217
such	34816
with	32575
as	31639
that	27281
other	25149
is	21545

type/token ratio or Lexical Diversity after punctuation removal: 0.009902045351801173

f) 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 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)?

Following are the top 20 most frequent words after removal of stopwords:

Word	**Frequency**
shall	48425
Agreement	37036
Party	20667
may	13160
Section	12409
party	11865
Company	11063
including	9588
Product	8877
use	8070
provided	7875
Parties	7688
time	7658
set	6873
written	6735
applicable	6477
b	6356
information	6342
forth	6238
right	6216

type/token ration after punctuation and stopword removal: **0.017343033712423427**

g) 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.

Top 20 most frequent pairs are listed below:

Bigram	**Frequency**
('set', 'forth')	**6018**
('Agreement', 'shall')	**3487**
('Confidential', 'Information')	**2871**
('third', 'party')	**2427**
('written', 'notice')	**2370**
('Effective', 'Date')	**2263**
('Party', 'shall')	**2228**
('Third', 'Party')	**2048**
('terms', 'conditions')	**1902**
('prior', 'written')	**1807**
('forth', 'Section')	**1689**
('time', 'time')	**1684**
('shall', 'deemed')	**1679**
('without', 'limitation')	**1658**
('Intellectual', 'Property')	**1637**
('shall', 'mean')	**1631**
('including', 'without')	**1463**
('shall', 'provide')	**1430**
('shall', 'right')	**1345**
('written', 'consent')	**1323**

Q) In your report, please fill in the following table, in addition to the answers to the above points:

# of tokens (b)	**4746517**
# of types (b)	**58850**
type/token ratio (b)	**0.0123985650952056**
tokens appeared only once (d)	**24320**
# of words (excluding punctuation) (e)	**3921917**
type/token ratio (excluding punctuation) (e)	**0.009902045351801173**
List the top 3 most frequent words and their frequencies (e)	**the: 239999** **of: 151825** **and: 129000**
type/token ratio (excluding punctuation and stop words) (f)	**0.017343033712423427**
List the top 3 most frequent words and their frequencies (excluding stop words) (f)	**shall: 48425** **agreement: 37036** **party: 20667**
List the top 3 most frequent bigrams and their frequencies (g)	**('set', 'forth'): 6018** **('Agreement', 'shall'): 3487** **('Confidential', 'Information'): 2871**

PART 2 SENTENCE EMBEDDINGS

Sentence embeddings are a vital part of NLP, and they work by transforming variable length sentences into fixed-size vectors and thus enable efficient comparison and similarity measurements. They help capture semantic information so that models can better understand and generalize patterns in Natural Language.

The following Sentence Embedding Models were taken as per the specified criteria:

1. SBERT (all-mpnet-base-v2) *[Criteria – SBERT implementation]*
2. Universal Sentence Encoder
3. SimCSE
4. InferSent
5. Jina Embeddings (LLM Model used in Dify.AI and JinaAI) *[Criteria – Recent Generative LLM]*

STS 2016 FileName	SE1 – SBERT all-mpnet-base-v2	SE2 – Universal Sentence Encoder	SE3- InferSent	SE4 - SIMCSE	SE5- Jina Embedding	Best Model
Input.answer-answer	0.74844	0.71594	0.52576	0.76293	0.82351	***Jina Embedding v2*** *(0.82351)*
Input.headlines	0.84351	0.76468	0.60739	0.79472	0.84023	***SBERT*** *(0.84351)*
Input.postediting	0.88007	0.85166	0.81739	0.84492	0.88602	***Jina Embedding v2*** *(0.88602)*
Input.plagiarism	0.82394	0.84068	0.76992	0.84310	0.85412	***Jina Embedding v2*** *(0.85412)*
Input.question-question	0.82133	0.75393	0.61960	0.72893	0.83697	***Jina Embedding v2*** *(0.83697)*

For RESULT ANALYSIS go to page 10

SBERT: SBERT is designed to capture semantic similarity between sentences or text passages. As this assignment involved measuring how similar or related two pieces of text are, SBERT can provide effective embeddings for this purpose. For tasks link text classification where understanding the context of the paragraph or entire sentences is crucial, SBERT excels at generating embeddings for them. SBERT introduces the Siamese network concept meaning that each time two sentences are passed independently through the same BERT model. The architecture of SBERT follows, passing of a sentence through BERT, after which a pooling layer is applied to BERT model embeddings to get their lower dimensionality representation: initially 512 (768-dimensional) vectors are transformed to a single (768-dimensional) vector. For the pooling layer, SBERT authors proposed electing a mean-pooling layer as a default one, though they also mention that it is possible to use the max-pooling strategy or simply to

take the output of the [CLS] token instead. SBERT often utilizes transfer learning techniques, allowing you to leverage pre-trained models on large corpora.

Reason for choosing the all-mpnet-base-v2 model:

- It is the best performing model amongst SBERT transformer models as per Pretrained Models — Sentence-Transformers documentation (sbert.net) .
- Also, SBERT provides better performance than even OpenAI GPT-3 Models as per this article: OpenAI GPT-3 Text Embeddings - Really a new state-of-the-art in dense text embeddings? | by Nils Reimers | Medium

Model	Costs encoding 1M docs	Performance
OpenAI GPT-3 Embedding Models		
text-similarity-ada	$800	61.86
text-similarity-babbage	$1,200	62.62
text-similarity-curie	$6,000	62.39
text-similarity-davinci	$60,000	58.11
Google Embedding Models		
st5-base-1	$0.70	67.84
st5-large-1	$2.40	68.74
st5-3b-1	$6.80	69.23
universal-sentence-encoder-large-5	$0.35	64.51
Sentence-Transformers Model		
all-MiniLM-L6-v2	$0.12	68.06
all-MiniLM-L12-v1	$0.25	68.83
all-mpnet-base-v1	$0.70	69.98
all-roberta-large-v1	$2.40	70.23

Encoding costs & average performance on 14 sentence embeddings tasks of OpenAI GPT-3 Embeddings models in comparison to open alternatives. Full results

IMAGE SOURCE – Medium OpenAI GPT-3 Text Embeddings

Universal Sentence Encoder: The Universal Sentence Encoder (USE) encodes text into high dimensional vectors that can be used for text classification, semantic similarity, clustering. USE is designed to be versatile and can be applied to a wide range of NLP tasks, including text similarity, classification, clustering, and more. Its embeddings capture semantic information, making it useful for understanding the meaning of sentences. As we have a large and diverse dataset, this provides embeddings that capture a wide range of linguistic patterns and semantics. The idea behind USE is to summarize any given sentence to a 512-dimensional sentence embedding. Since the same embedding must work on multiple generic tasks, it captures only the most informative features and discards noise. The intuition is that this results in a generic embedding that transfers universally to a wide variety of NLP tasks, such as relatedness, clustering, paraphrase detection, and text classification.

Reasons for choosing:

- Ease of use as it integrates easily into the TensorFlow-based workflow.
- Computationally Efficient Embeddings are generated by USE which also capture semantic information.

SimCSE (Similarity-based Contrastive Learning of Sentence Embeddings): SimCSE is an excellent choice for an embedding model due to its emphasis on contrastive learning. Its simplicity, combined with effectiveness, makes it accessible for learners new to NLP and contrastive learning concepts. The model utilizes data augmentation techniques to create positive pairs, enhancing generalization and accommodating diverse inputs. Pre-trained SimCSE models are good for facilitating transfer learning for various downstream tasks with limited labeled data. Its state-of-the-art performance on benchmark datasets showcases its capability in sentence similarity and natural language understanding tasks. Overall, SimCSE stands out as an effective model for understanding modern techniques in sentence embeddings and contrastive learning within the NLP domain.

Reasons for choosing:

- Competitive performance demonstrated on benchmark datasets.
- Contrastive learning to capture semantic similarity and relationships.

InferSent: InferSent is designed to capture diverse semantic information within sentences. It uses supervised training on natural language inference (NLI) data, making it effective for tasks requiring a nuanced understanding of sentence relationships. The architecture is based on the attention mechanism, which allows interpretability of the learned sentence embeddings. Furthermore, it also produces high-quality sentence embeddings that perform well on a variety of NLP tasks, including semantic similarity, text classification, and sentiment analysis.

Reasons for choosing:

- Specifically trained for Sentence Level Tasks so they are suited to our task.

Jina-Embeddings-v2:

Jina Embeddings version 2 falls in the criteria of Sentence Embeddings that are used in generative LLMs. Jina Embedding is implementable via Hugging Face and it is easily accessible as it just needs a Hugging Face access Token. The model supports only English and is based on a BERT architecture. It is available in two sizes – small with 33 million parameters; and base with 137 million parameters. As mentioned in the Hugging Face leaderboards, it did indeed turn out to be the best embedding model for STS 2016 dataset outperforming the others.

Reasons for choosing:

- Freely Accessibly LLM embedding model which can be accessed via Hugging Face tokens.
- On the Hugging Face Leaderboard jina-embeddings-v2 outperforms text-embedding-ada-002.

Result Analysis:

STS 2016 FileName	SE1 – SBERT allnetbasev2	SE2 – Universal Sentence Encoder	SE3- InferSent	SE4 - SIMCSE	SE5- Jina Embedding	Best Model
Input.answer-answer	0.74844	0.71594	0.52576	0.76293	0.82351	***Jina Embeddings v2*** *(0.82351)*
Input.headlines	0.84351	0.76468	0.60739	0.79472	0.84023	***SBERT*** *(0.84351)*
Input.postediting	0.88007	0.85166	0.81739	0.84492	0.88602	***Jina Embeddings v2*** *(0.88602)*
Input.plagiarism	0.82394	0.84068	0.76992	0.84310	0.85412	***Jina Embeddings v2*** *(0.85412)*
Input.question-question	0.82133	0.75393	0.61960	0.72893	0.83697	***Jina Embeddings v2*** *(0.83697)*
Average Score	0.82346	0.78538	0.66801	0.79492	0.84817	***Jina Embeddings v2 (***0.84817***)***

Our evaluation of the 5 models for every file when matched with the provided Perl script for correlation results gave us the results mentioned above.

The Jina Embedding Model was found to outperform the rest in 4 input files:

- Answer
- Post Editing
- Questions
- Plagiarism

The SBERT model just marginally came on top of the Jina model in just one input file:

- Headlines

We have also taken an average score over the 5 files, and we find that the Jina Embeddings Version 2 succeeded with the max score of 0.848. If we rank the models based on this average score criterion:

1. Jina Embeddings V2
2. SBERT all-mpnet-base-v2
3. SimCSE
4. Universal Sentence Encoder
5. InferSent

REFERENCES:

1. Pretrained Models — Sentence-Transformers documentation (sbert.net)
2. Unraveling Textual Semantic Similarity: Jina Embeddings vs. Llama Models implementations | by Anoop Johny | Medium
3. OpenAI GPT-3 Text Embeddings - Really a new state-of-the-art in dense text embeddings? | by Nils Reimers | Medium
4. Pretrained Models — Sentence-Transformers documentation (sbert.net)
5. jinaai/jina-embeddings-v2-base-en · Hugging Face
6. GitHub - facebookresearch/InferSent: InferSent sentence embeddings
7. GitHub - princeton-nlp/SimCSE: [EMNLP 2021] SimCSE: Simple Contrastive Learning of Sentence Embeddings https://arxiv.org/abs/2104.08821
8. Universal Sentence Encoder | TensorFlow Hub
9. Guide To Universal Sentence Encoder With TensorFlow- Analytics India Magazine