Tokenization and Semantic Text Similarity

Xiao Yang, Mingze Li, Sara Haroon

Carleton University

Ottawa, Ontario

@ February 4, 2024

Contents

Chapter 1

Tokenization on Legal Texts

We implement a word tokenizer that splits texts into tokens and separates punctuation marks and other symbols from the words and a program that counts the number of occurrences of each token in the corpus.

Github Repository:

`https://github.com/Barnett888/Tokenizer-and-Semantic-Similarity-Test/tree/main`

1.1 Data Set

We adopted the Contract Understanding Atticus Dataset (CUAD) data set to test our tokenizer. CUAD [5] is a new dataset for legal contract review, it was created with dozens of legal experts from The Atticus Project and consists of over $13,000$ annotations. The task is to highlight salient portions of a contract that are important for a human to review.

1.2 Description of the Method

There are 510 files in the CUAD data set. We used a for loop to read them and save their text into a single list. The list contains 510 string elements corresponding to the 510 text files in CUAD.

The optimal tokenizer should be able to identify multiple symbols, times, dates, and digits such as 2019/05/18, 12:20 pm, or decimal numbers like 5985.2. They should be recognized as one token. However, most existing tokenizers can not do this. As a result, we implemented our own tokenizer using the RE library. We also programmed and implemented a lemma and contraction fixer using Natural Language Toolkit (NLTK) [1], and we stick to this tokenizer throughout the task.

1.3 Tokenizer Output

We name our ouput file 'output.txt' for the whole corpus, and include the first 20 lines from output.txt file.

The output of the tokenizer is a list of strings of all 510 legal texts.

The first 20 **lines of the tokenizer are:**`https://github.com/Barnett888/Tokenizer-and-Semantic-Similarity-Test/blob/main/Task1/1_a.txt`

1.4 Token Counts and Type Token Ratio

How many tokens did you find in the corpus?

The total number of tokens is: 4730682

How many types (unique tokens) did you have?

The total number of unique tokens is: 38413

What is the type/token ratio for the corpus?

The type/token ratio of the corpus is: 0.008119970862552164

1.5 Output file Tokens.txt

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

The first 20 **lines in the token.txt are:**`https://github.com/Barnett888/Tokenizer-and-Semantic-Similarity-Test/blob/main/Task1/1_c.txt`

1.6 Number of unique tokens

How many tokens appeared only once in the corpus?

14437 **tokens appeared only once in the corpus.**

1.7 Diversity

From the list of tokens, we extract only words, by excluding punctuation and other symbols.

How many words did you find?

The total number of words is: 3920618

The total number of unique words is: 27721

List the top 20 most frequent words, with their frequencies.

https://github.com/Barnett888/Tokenizer-and-Semantic-Similarity-Test/blob/main/Task1/1_f.txt

What is the type/token ratio when you use only words (called lexical diversity)?

The lexical diversity(type/word ratio) of the corpus is: 0.007070568976625623

1.8 Stop words

From the list of words, exclude stop words. List the top 20 most frequent words and their frequencies. Compute the type/token ratio when you use only word tokens without stop words (called lexical density).

The top 20 **frequent words and frequencies are:**

[('agreement', 43655), ('party', 33277), ('parties', 13523), ('section', 13350), ('company', 12638), ('information', 10943), ('product', 10923), ('date', 10181), ('products', 8201), ('rights', 8067), ('services', 7890), ('applicable', 7540), ('business', 7343), ('set', 7058), ('confidential', 6916), ('written', 6818), ('terms', 6714), ('right', 6681), ('term', 6676), ('notice', 6660)]

The lexical diversity(type/word ratio) of the corpus is: 0.0145353172

1.9 Bigrams

Compute all the pairs of two consecutive words (bigrams) (excluding stopwords and punctuation). List the most frequent 20 pairs and their frequencies.

The top 20 **frequent consecutive words and frequencies are:**

`https://github.com/Barnett888/Tokenizer-and-Semantic-Similarity-Test/blob/main/Task1/1_g.txt`

The following table 1.1 summarizes the outputs.

# of types (b)	38413
type/token ratio (b)	0.00812
# of tokens appeared only once (d)	14437
# of words (excluding punctuation) (e)	3920618
type/token ratio (excluding punctuation) (e)	0.0070706
the top 3 most frequent words and their frequencies (e)	'the', 257278 'of', 156457 'and', 132874
type/token ratio (excluding punctuation and stopwords) (f)	0.01453532
The top 3 most frequent words and their frequencies (excluding stopwords) (f)	'agreement', 4365 'party', 3327 'parties', 13523
The top 3 most frequent bigrams and their frequencies (g)	('confidential', 'information'), 3607 ('intellectual', 'property'), 2936) ('effective', 'date'),2840

Table 1.1: Summarizing Tokenizer Output

Chapter 2

Semantic Text Similarity

Word embeddings are dense representations of the meaning of words, build via neural language models. Sentence embeddings are dense representations of sentences, that can be composed by averaging the word vectors or sentence representations can be learned directly. We chose 5 pre-trained sentence embedding models to compare their performance on a bench mark data set.

2.1 Data Set

The data set is test data of gold labels from the Semantic Textual Similarity (STS) competition [2]. STS seeks to measure the degree of semantic equivalence between two snippets of text. It contains two columns of sentence pairs and a separate file of human scores. The data set is given at `http://alt.qcri.org/semeval2016/task1/data/uploads/sts2016-english-with-gs-v1.0.zip`, see readme file for more detail.

2.2 Description of Method

Our method is to generate a cosine similarity score for each sentence pairs for all 5 groups. As the cosine similarity between two sentence vectors ranges from -1 to 1, we normalize it to 0 to 5 using $g(x) = 5(x + 1)$. A score of 5 means the two sentences are completely equivalent, while 0 means they are completely different. For each model, we calculate the similarity score for all five groups of data and average them to obtain a single score as the model's final output score.

2.3 Model Description

A large language model (LLM) is a language model notable for its ability to achieve general-purpose language generation. LLMs are artificial neural networks, which are built with a transformer-based architecture. We selected five representative pretrained models of sentence embedding.

2.3.1 USE

Universal sentence encoder is a language model proposed by Google research in 2018 to encode sentences into embedding vectors of fixed length (512 dimensions) that specifically target transfer learning to other NLP tasks [3].

A transformer architecture and deep averaging network (DAN) approach are included as two variants. The input is English strings and the model produce a fixed dimensional embedding representation of the string as output. Transfer learning of transformer based sentence encoder usually performs better than the DAN encoder. The sentence level embeddings surpass the performance of transfer

learning using word level embeddings [3].

2.3.2 BERT

Bidirectional Encoder Representations from Transformers (BERT) is a language model based on the transformer architecture, introduced in October 2018 by Google AI languages.

BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers [4]. The pretrained BERT model can be fine tuned with one additional output layer to create state-of-the-art models for tasks, such as question answering and language inference, without substantial task specific architecture modifications [4].

Nowadays, BERT is notable for its dramatic improvement over previous state of the art models. BERT has now become a ubiquitous baseline in NLP experiments, and there has been numerous attempts to refine and improve the model based on its initial ideas.

2.3.3 RoBERTa

RoBERTa is a robustly optimized method for pretraining NLP systems that improves on BERT. RoBERTa is implemented in PyTorch, and it modifies key hyperparameters in BERT.

The model is essentially a replication study of BERT, motivated by the fact that BERT was significantly undertrained. RoBERTa [6] carefully evaluate design decisions when pretraining BERT models, and conclude BERT's performance can be substantially improved by training the model for longer time, with bigger

batches over more data; removing the next sentence prediction objective; training on longer sequences; and dynamically changing the masking pattern applied to the training data [6].

RoBERTa achieves state-of-the-art results on GLUE, RACE and SQuAD, illustrating the importance of previously overlooked design decisions and suggest BERT's pretraining objective remains competitive with recently proposed alternatives [6].

2.3.4 DistilBERT

DistilBERT is a distilled version of the BERT language model. DistilBERT uses knowledge distillation, a technique used for getting a smaller model (student) from a larger one (teacher) by training the student to reproduce the results of the teacher [7].

Like BERT, DistilBERT is based on Transformer architecture. Investigations by Sanh et. al showed that reducing the number of layers had a larger impact on computational efficiency than other variations to the original architecture. Thus, DistilBERT was created from BERT by having the token-type embeddings and pooler layers removed, and only keeping half of the remaining layers. This leads to DistilBERT having 40% fewer parameters than BERT [7].

DistilBERT achieves 97% of BERT's score on GLUE, showing that it retains most of BERT's performance despite its smaller size. It also achieves this performance in 60% less time than BERT [7].

2.3.5 SBERT

Sentence-BERT (SBERT) is a language model that is a modification of BERT. BERT comes with a high computational and time cost for problems of semantic textual similarity (STS) and clustering. SBERT is a solution to this problem.

The high computational and time cost of BERT in checking STS comes from its cross-encoder, which checks all pair-wise combinations of sentences given to the model and finds the pair with the highest similarity. With a large input of 10,000 sentences, this can easily lead to 65 hours of computations on a GPU. Even if we derive fixed-sized sentence embeddings or average the output layer (BERT embeddings), we would get bad sentence embeddings.

A solution to this problem lies in SBERT's siamese network which allows us to derive fixed-size vectors for each input sentence. The siamese network runs two sentences individually through separate BERT models, updating weights for both models simulataneously, and then applying a pooling operation to the output of each BERT run. In fact SBERT gives the best overall score of the five models

2.4 Result

The following table 2.1 summarizes the pearson correlation coefficient (scores) for each groups. A value closes to 1 indicates a perfect match with the human score gold standard. From the five models we selected, back in 2018, SBERT gives the best overall performance.

	answer	question	headlines	plagiarism	postediting	average
Model 1: USE	0.352	0.797	0.793	0.139	0.167	0.45
Model 2: BERT	0.331	0.679	0.951	0.195	0.186	0.468
Model 3: RoberTa	0.262	0.594	0.71	0.165	0.164	0.379
Model 4: DistilBERT	0.322	0.743	0.759	0.116	0.156	0.419
Model 5: SBERT	0.349	0.75	0.953	0.139	0.167	0.472

Table 2.1: STS LLM Pearson Coefficients

Datasets	SE1	SE2	SE3	SE4	SE5	Best Score
STS test data:	0.4496	0.4684	0.3789	0.4192	0.4715	0.4715

Table 2.2: STS LLM Pearson Coefficients2

Bibliography

[1] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[2] Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In Steven Bethard, Marine Carpuat, Marianna Apidianaki, Saif M. Mohammad, Daniel Cer, and David Jurgens, editors, *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada, August 2017. Association for Computational Linguistics. URL: `https://aclanthology.org/S17-2001`, `doi:10.18653/v1/S17-2001`.

[3] Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder, 2018. `arXiv:1803.11175`.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert:

Pre-training of deep bidirectional transformers for language understanding, 2019. arXiv:1810.04805.

[5] Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. Cuad: An expert-annotated nlp dataset for legal contract review, 2021. arXiv:2103.06268.

[6] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. arXiv:1907.11692.

[7] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020. arXiv:1910.01108.