Attn: Dr. Sun Aixin

CE/CZ4045 Natural Language Processing

We hereby declare that the attached group assignment has been researched, undertaken, completed and submitted as a collective effort by the group members listed below. We have honored the principles of academic integrity and have upheld Student Code of Academic Conduct in the completion of this work. We understand that if plagiarism is found in the assignment, then lower marks or no marks will be awarded for the assessed work.

Name	Signature / Date
Chen Sihao U1720908B	CSH 2nd November 2020
Ng Kai Chin U1721647D	NKC 2nd November 2020
Thong Hoi Wei U1721328H	THW 2nd November 2020
Yao Cheng Hui U1721932J	YCH 2nd November 2020
Zachary Chua Chee Kian U1721622J	ZCCK 2nd November 2020

Important note:

Name must **EXACTLY MATCH** the one printed on your Matriculation Card. Any mismatch leads to **THREE (3)** marks deduction.

CZ4045 Natural Language Processing – Assignment (Part 1)

Chen Sihao
U1720908B
SCHEN027@e.ntu.edu.sg

Ng Kai Chin
U1721647D
KNG054@e.ntu.edu.sg

Thong Hoi Wei
U1721328H
TH0001EI@e.ntu.edu.sg

Yao Cheng Hui
U1721932J
YAOC0009@e.ntu.edu.sg

Zachary Chua Chee Kian
U1721622J
ZCHUA025@e.ntu.edu.sg

ABSTRACT

This report explores domain specific text data analysis and processing and other Natural Language Processing tasks on various datasets. It explains methodologies used and discusses results obtained.

CCS CONCEPTS

• Artificial Intelligence → Natural Language Processing → Information Extraction • Applied Computing → Document Management and Text Processing

1 Domain Specific Dataset Analysis

Three datasets were chosen for this analysis. They are mainly:

(1) COVID-19 related research paper abstracts (CORD-19)[1]
(2) Wikipedia Math Equations[2]
(3) Finance Tweets[3]

They are chosen mainly because they have linguistic characteristics that are unique to their particular domain. These characteristics will be discussed further in this report. In this experiment, the "nltk" library is used with the library's "Punkt Sentence Tokenizer" used as the tokenizer.

1.1 Tokenization and Stemming

1.1.1 CORD-19

The tokenizer is generally able to recognize domain specific linguistic terms that are related to pneumonia and the medical and biological terms. Below are some examples of correctly identified terms:

Domain Specific Terms[4]	Remarks
Endothelin-1	Tokenizer correctly kept the number from the term
Nidovirus | subgenomic | mRNAs	Tokenizer split the phrase by the spacing
5′ | end | of | the | genome	Tokenizer kept the apostrophe with the number "5"

However, there are also wrongly identified terms:

Domain Specific Terms	Remarks
Reverse | Transcription-polymerase | chain | reaction	"Transcription" should be separated from PCR
100 | g/l	"g/l" refers to grams per litre, so it should be separated
S.pombe	S.pombe is the yeast Schizosaccharomyces pombe. The term should be separated into its genus and species.

1.1.2 ACSII Math Equations

The tokenizer makes several mistakes and is unable to recognize several domain specific terms. This is expected as ACSII math equations have a unique syntax and the lack of spaces in most

[1] Source: COVID-19 Open Research Dataset Challenge (CORD-19).
CORD-19 is a resource of over 200,000 scholarly articles, including over 100,000 with full text, about COVID-19, SARS-CoV-2, and related coronaviruses. In this assignment, we will use the abstracts of these papers.

[2] Source: AsciiMath Equations Collected From Wikipedia. The dataset contains collection of simple equations written in AsciiMath.

[3] Source: Finance Tweets Dataset publicly available on Kaggle. This dataset has all the publicly traded companies (tickers and company names) that were used as input to fill the tweets.csv.

[4] "|" represents where the character sequence is separated

scripts makes it difficult for the tokenizer to recognise where to separate the tokens.

Domain Specific Terms[5]	Remarks
$e^2 \frac{b_0 - \sin v}{R}$ e^ { 2 } \frac { b_0- \sin v } { R }	Right: "e^" and "2" are rightly separated. "\frac" is correctly kept together Wrong: The term "b_0-\sin" should be separated.
$ty \leq \left\{ \sum_{n=1}^{\infty} f_n(x, \phi,$ \textstyle { \sum_ { n=1 } ^\infty f_n (x , \phi , \psi) }	Right: Parameters in the f_n function is rightly separated Wrong: "\textstyle" should be separated into the following tokens: "\texts", "ty", "le". "n=1" should be separated
$x \in E, m \geq N, n \geq N$ x\in E , m\geq N , n\geq N\implies |f_m (x) - f_n (x) | &lt ; \epsilon	Wrong: Tokens on both sides of the punctuations "\", "|" and "-" are kept together instead of being separated. Especially for "|" and "-", as they are not just punctuations in math equations.

1.1.3 Finance

The tokenizer is generally able to identify domain specific linguistics as many of the financial terms follow the linguistics pattern of the English language. However, there commonly abbreviated terms that are not identified and also general errors made by the tokenizer.

Correctly identified terms:

Domain Specific Terms	Remarks
| 07/20/2018 |	Tokenizer correctly identified dates and kept the values together in a single token
Range | is | 150.00 | to | 170.00	Values are identified correctly, 150.00 is not split at the '.'.

Wrongly identified terms:

Domain Specific Terms	Remarks
YTD	"YTD" refers to Year to Date, so it should be separated
$ | TAL | $ | SPXS | $ | MXIM | $ | WELL | $ |MMM | $ | SRC | $ | CSX | $ | KBR	Tokenizer split the $ from the ticker which should not be the case, as the $ is the symbol for the ticker.
' | s	Tokenizer incorrectly split ' and s, when it should be tokenized as 's.

1.1.4 Identifying wrong tokens and improvements

The Punkt tokenizer works by counting punctuation and tokens that commonly denote a boundary in a sentence before using the resulting frequencies to decide where the boundaries are. Therefore, the tokenizer might be confused with punctuations that would typically denote a boundary in common language but not in domain specific contexts.

For example "x.y", without a space after the period, may usually be numbers (eg. version 2.2) and hence should be kept together. However, in biology, microorganisms like "S.pombe" are named with their genus and species, and typically their genus is shortened by its acronym and a period sign.

Likewise, the tokenizer is also confused with the "-" punctuation. Generally, words preceding and succeeding "-" should be kept together but words like "transcription-polymerase" should be separate. Consider the case of "reverse transcription-polymerase chain reaction", if one uses "transcription-polymerase" as a token instead, there will not be matches with "polymerase chain reaction", which is a superset of the original phrase. Perhaps the tokenizer can look at words before and after the "x-y" phrase to determine if words "x" and "y" should be separated.

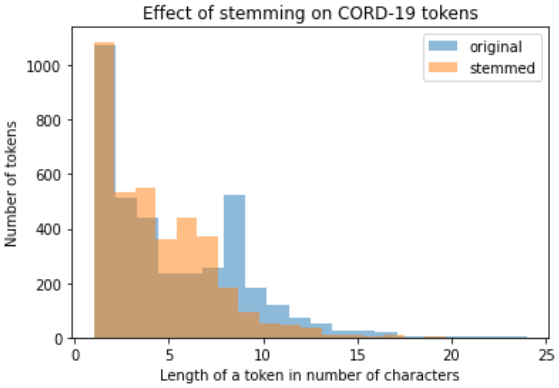
The mistakes made by the tokenizer for the math equations are more obvious. Punctuations like "-", "/", "|" all have semantic meanings and should be separated from the phrases. One possible improvement for the tokenizer is to apply different rules depending on the domain of the text. This is as especially relevant for ACSII math equations as they operate on entirely different grammar rules as normal text.

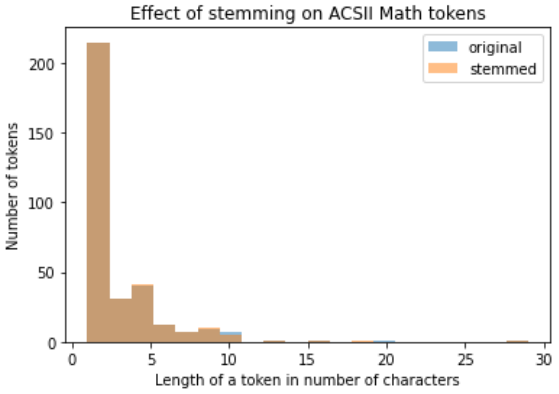
Lastly, for tweets about Financial analysis, there are generally a few common mistakes by the tokenizer. Firstly, there are financial terms that are typically abbreviated and also informal tweets have abbreviations as well. Due to this issue, some phrases are

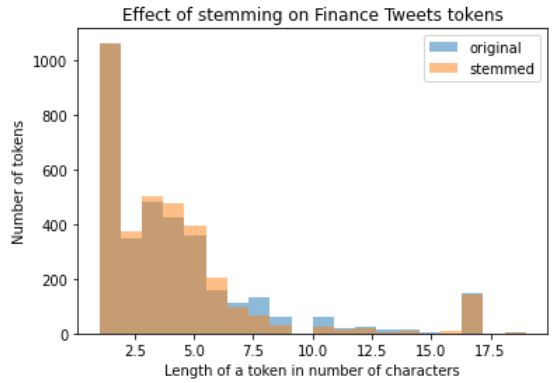
[5] 3 blank spaces denote whether the tokens are separated

incorrectly identified to be a single word, such as "YTD" which actually means "Year to Date". Secondly, there are symbols used in conjunction with tickers (abbreviated names of stocks), which should be part of a single token as they represent different securities, for example "$AAPL" represents Apple stocks, while "/ES" represents S&P500 Index Futures. These are mistaken by the Punkt tokenizer to be symbols representing "dollar value" and have been separated from the ticker symbol incorrectly.

1.1.5 Effect of Stemming







NLTK's Snowball Stemmer is used to investigate the effect of stemming on the character length of tokens. For normal text, it is expected that the character length of tokens will decrease after performing stemming. This is because a stemmer removes the prefix or suffix of a word to the roots of the word.

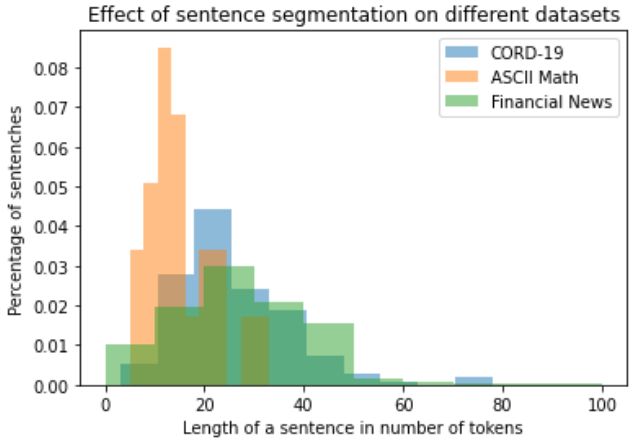
For the CORD-19 dataset, the character length per token clearly decreases after stemming. The distribution of character lengths shifts to the left and the average character length per token also decreased from 5.2 to 4.45.

As for the ASCII Math Equations dataset, there is almost no change in character length per token. The average character length per token decreased from 2.63 to 2.61, which is a marginal change compared to the CORD-19 dataset. This is because variables used in mathematics equations are usually simple words or symbols and would not change after stemming.

Lastly, for the dataset on Finance Tweets, there is a small decrease in character length on average from 4.15 to 3.89. This marginal decrease is likely due to the fact that, most of the words are abbreviations, ticker symbols, or numerical values, hence there are not many instances of characters decreasing due to stemming.

1.2 Sentence Segmentation

1.2.1 Comparing across all datasets



The above graph shows the distribution of the sentence length across the three domains. As each dataset has different numbers of sentences per entry, we chose the percentage of sentences instead of the absolute number of sentences for the y-axis. As expected, "sentences" or equations in the ASCII Math dataset are shorter than the rest of the dataset. It is unusual for equations to be too long as it would be unreadable.

For CORD-19, they are research papers so they tend to have longer sentences than most documents. However, they are also usually not as long to improve readability. These papers are usually peer-reviewed and long sentences might be revised to make smaller and readable sentences instead. As the dataset only looked at abstracts, it is necessary to use shorter sentences to fit the word limit.

As for financial tweets, the distribution of the sentences are more varied, encompassing both ends of the spectrum. A likely reason could be that these articles are less formal than research papers

and the sentence length distribution is likely due to different writing styles of the authors.

1.3 POS Tagging

NLTK's Averaged Perceptron Tagger is used for this exercise.

1.3.1 POS tagging for CORD-19

POS Tagging results:

('Nidovirus', 'NNP'), ('subgenomic', 'JJ'), ('mRNAs', 'NN'), ('contain', 'NN'), ('a', 'DT'), ('leader', 'NN'), ('sequence', 'NN'), ('derived', 'VBD'), ('from', 'IN'), ('the', 'DT'), ('5′', 'CD'), ('end', 'NN'), ('of', 'IN'), ('the', 'DT'), ('genome', 'NN'), ('fused', 'VBD'), ('to', 'TO'), ('different', 'JJ'), ('sequences', 'NNS'), ('(', '('), ('‘', 'JJ'), ('bodies', 'NNS'), ('’', 'VBP'), (')', ')'), ('derived', 'VBN'), ('from', 'IN'), ('the', 'DT'), ('3′', 'CD'), ('end', 'NN'), ('.', '.')	('METHODS', 'NNS'), (':', ':'), ('Patients', 'NNS'), ('with', 'IN'), ('positive', 'JJ'), ('M.', 'NNP'), ('pneumoniae', 'NN'), ('cultures', 'NNS'), ('from', 'IN'), ('respiratory', 'NN'), ('specimens', 'NNS'), ('from', 'IN'), ('January', 'NNP'), ('1997', 'CD'), ('through', 'IN'), ('December', 'NNP'), ('1998', 'CD'), ('were', 'VBD'), ('identified', 'VBN'), ('through', 'IN'), ('the', 'DT'), ('Microbiology', 'NNP'), ('records', 'NNS'), ('.', '.')	('Heme', 'NNP'), ('oxygenase-1', 'JJ'), ('(', '('), ('HO-1', 'NNP'), (')', ')'), (',', ','), ('an', 'DT'), ('inducible', 'JJ'), ('stress', 'NN'), ('protein', 'NN'), (',', ','), ('confers', 'NNS'), ('cytoprotection', 'VBP'), ('against', 'IN'), ('oxidative', 'JJ'), ('stress', 'NN'), ('in', 'IN'), ('vitro', 'NN'), ('and', 'CC'), ('in', 'IN'), ('vivo', 'NN'), ('.', '.')

The POS Tagger is generally accurate for the CORD-19 dataset. For the first sentence, "Nidovirus" is classified as a proper noun (singular / mass), which is correct as it is a name of a virus and it is considered singular as virus is not quantifiable. "5′" and "3′" are considered cardinal which is correct, as it represents the position of the linkage on the nucleotide sugar ring of the mRNA. However, "mRNAs" should be a plural noun, as its singular form is "mRNA". The tagger could be confused with the use of acronyms in this example. The apostrophe signs are also incorrectly tagged as 'JJ' and 'VBP' respectively.

For the second sentence, all the tags are correct except "pneumoniae" which should be a proper noun which is the name of the bacteria. The confusion could be due to the similarity with "pneumonia" that the bacteria causes. As for the third sentence, 'oxygenase-1' should be classified as a noun instead of an adjective. 'Cytoprotection' is also not a verb but a noun.

1.3.2 POS tagging for ACSII Math Equations

POS Tagging results:

('|f_N', 'NN'), ('(', '('), ('x_0', 'NNP'), (')', ')'), ('-f', 'NN'), ('(', '('), ('x_0', 'NNP'), (')', ')'), ('|=\Big|', 'VBP'), ('[', 'NNP'), ('(', '('), ('1/2', 'CD'), (')', ')'), ('^', 'NN'), ('{', '('), ('1/N', 'CD'), ('}', ')'), (']', 'NNP'), ('^N-0\Big|=1/2', 'NNP'), ('&', 'CC'), ('gt', 'NN'), (';', ':'), ('\epsilon', 'CC')	('e^', 'RB'), ('{', '('), ('2', 'CD'), ('}', ')'), ('\frac', 'JJR'), ('{', '('), ('b_0-\sin', 'JJ'), ('v', 'NN'), ('}', ')'), ('{', '('), ('R', 'NNP'), ('}', ')')	('\underset', 'VBN'), ('{', '('), ('n\to\infty', 'JJ'), ('}', ')'), ('{', '('), ('\mathrm', 'JJ'), ('{', '('), ('unif\', 'JJ'), ('lim', 'NN'), ('}', ')'), ('}', ')'), ('f_n', 'JJ'), ('=', 'NNP'), ('f', 'NN')

As expected, POS Tagging on math equations does not work due to the different rules used by the ASCII notation. The equations do not use conventional speech tags and it would not be meaningful to evaluate these POS Tags.

1.3.3 POS tagging for Finance Tweets

POS Tagging results:

('VIDEO', 'NN'), (':', ':'), ('“', 'NN'), ('I', 'PRP'), ('was', 'VBD'), ('in', 'IN'), ('my', 'PRP$'), ('office', 'NN'), ('.', '.') ('I', 'PRP'), ('was', 'VBD'), ('minding', 'VBG'), ('my', 'PRP$'), ('own', 'JJ'), ('business', 'NN'), ('...', ':'), ('”', 'NN'), ('–David', 'JJ'), ('Solomon', 'NNP'), ('tells', 'VBZ'), ('$', '$'), ('GS', 'NNP'), ('interns', 'VBZ'), ('how', 'WRB'), ('he', 'PRP'), ('learned', 'VBD'), ('he', 'PRP'), ('wa…', 'VBZ'), ('https', 'NN'), (':', ':'), ('//t.co/QClAITywXV', 'NN')	('The', 'DT'), ('price', 'NN'), ('of', 'IN'), ('lumber', 'JJ'), ('$', '$'), ('LB_F', 'NNP'), ('is', 'VBZ'), ('down', 'RB'), ('22', 'CD'), ('%', 'NN'), ('since', 'IN'), ('hitting', 'VBG'), ('its', 'PRP$'), ('YTD', 'NNP'), ('highs', 'NNS'), ('.', '.') ('The', 'DT'), ('Macy', 'NNP'), ("'s", 'POS'), ('$', '$'), ('M', 'NNP'), ('turnaround', 'NN'), ('is', 'VBZ'), ('still', 'RB'), ('happening….', 'JJ'), ('https', 'NN'), (':', ':'), ('//t.co/XnKsV4De39', 'NN')	('Barry', 'NNP'), ('Silbert', 'NNP'), ('is', 'VBZ'), ('extremely', 'RB'), ('optimistic', 'JJ'), ('on', 'IN'), ('bitcoin', 'NN'), ('--', ':'), ('but', 'CC'), ('predicts', 'VBZ'), ('that', 'DT'), ('99', 'CD'), ('%', 'NN'), ('of', 'IN'), ('new', 'JJ'), ('crypto', 'NN'), ('entrants', 'NNS'), ('are', 'VBP'), ('“', 'JJ'), ('going', 'VBG'), ('to', 'TO'), ('zero…', 'VB'), ('https', 'NN'), (':', ':'), ('//t.co/mGMVo2cZgY', 'NN')

POS Tagging is generally accurate for the finance tweets dataset as the language is still proper and follows the typical English lexicon. The accuracy is reduced due to specific symbols used for

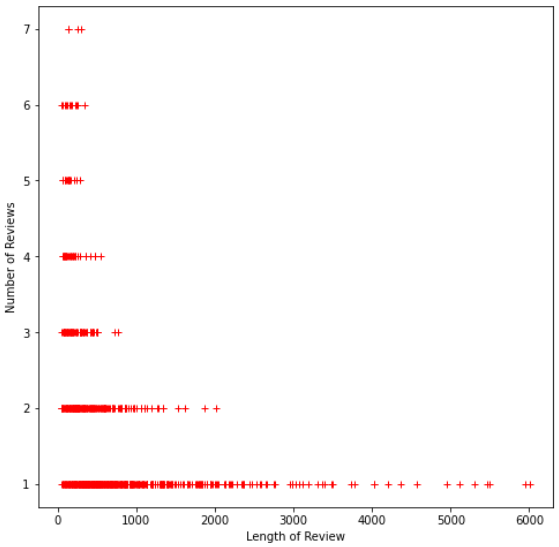
finance that was not accurately identified by the POS tagging. In the first sentence, "$" was identified as "$" which is incorrect as the symbol does not serve the typical use case of dollar value, but is an indication of the ticker symbol; "GS" that follows it. Also, the quotes symbol is incorrectly identified as 'NN' instead of the symbol itself.

2 Developing a <Noun-Adjective> Pair Ranker

2.1 Choice of dataset and data cleaning

The dataset chosen for this task was IMDB user reviews for the film Enola Holmes (2020). The reviews were webscrapped from imdb.com into a .csv file.

The dataset had the following columns: ['user_name', 'review_title', 'review_date', 'review_text', 'rating']. Only 'review_text' and 'rating' were used for this task. There were a total of 1246 reviews.



The graph shows the distribution of reviews by their length (in characters).

Reviews that were <300 characters long were filtered out so that each review would have a sizable number of <noun, adj> pairs. Thereafter, to get a representative sample of 30 reviews, the reduced dataset was sampled via random sampling of 3 reviews per rating value.

2.2 Natural Language Processing

2.2.1 Identifying of <noun-adj> pairs

To identify <noun-adj> pairs, each sentence's dependency tree was visualised. Spacy's English tokenizer was used to identify adjective phrases and noun phrases from each review.

(1) Each word in a review is tokenised.
(2) For each token which is either ['NOUN', 'PROPN'], the corresponding child adjective phrases are identified.
(3) The <noun, adj> pairs are generated for each adjective/adjective phrase determined.

2.2.2 Determining intensity values for each pair

NTLK's VADER (Valence Aware Dictionary and Sentiment Reasoner) sentiment analyser was used to determine the intensity value of each identified pair. VADER is an open source tool which has had success on social media texts and user reviews.

The intensity value uses the 'compound' score of the <noun, adj> pair, which is a 'normalised weighted composite score' of the phrase, which best represents the overall sentiment of the pair.

2.2.3 Definition of meaningful and human identified phrases

Meaning was defined as the degree of the intensity that the <noun, adj> pair has and its impact on the reader in the context of the review. Notably, meaningful phrases could either express positive or negative sentiments.

The top 5 human-identified phrases are as follows:

Rank	<noun, adj> pair	Rating of review
1	'mess', 'gigantic'	4
2	'success', 'great'	10
3	'talent', 'pure'	10
4	'failure', 'complete'	5
5	'propaganda movie', 'stupid'	2

2.2.4 Explanation of scoring method

The *rating* for each review was scaled between [-1, 1]. The overall score for each pair was then calculated as intensity * rating.

The *intensity* value represents the uncontextualized sentiment of each pair. The rating contextualises this sentiment within the review that it occurs in. For example, a pair with a positive intensity will have a higher score when it occurs in a highly rated review (close to 10/10) than when it occurs in a more moderately positive rated review. Intuitively, the positive sentiment of the pair has more impact and relevance in a highly positive review, making it more meaningful.

This scoring system also enables negative intensity pairs in negatively rated (i.e. <5/10) reviews to achieve an overall high score.

A pair that has a low, or even negative score likely has a low intensity sentiment to begin with, or occurred in a moderately rated review (i.e. close to 5).

2.2.5 Results and Discussion

Rank	Noun	Adj	Intensity	Rating	Score
1	success	great	0.8316	1.0	0.8316
2	love story	cute	0.8020	1.0	0.8020
3	love story	little	0.6003	1.0	0.6003
4	cinemetography	Beautiful	0.5994	1.0	0.5994
5	Saturday film	charming	0.5859	1.0	0.5859

The table above shows the top 5 pairs based on score, where score = intensity * rating. Compared to the human-identified phrases, <success, great> is the only pair that appears in both top 5 lists.

It can be seen that from the ranking system of the pair ranker, it favours pairs that occur in strongly rated reviews. The majority of the human-identified pairs were from moderately rated reviews, and hence were ranked lower.

Rank	Noun	Adj	Intensity	Rating	Score
52	mess	gigantic	-0.3612	-0.4	0.14448
1	success	great	0.8316	1.0	0.8316
17	talent	pure	0.4215	1.0	0.4215
58	failure	complete	-0.5106	-0.2	0.1021
9	propaganda movie	stupid	-0.6597	-0.8	0.5278

2.2.6 Challenges Encountered

During the designing of the pair ranker, there was debate regarding the definition of meaning. It was decided that meaning should not be confined to positive emotions as reviews are polarising by nature and can be expected to contain varied emotions which should be indiscriminately taken into account.

Also, as this task requires the dataset to be reduced to only 30 reviews, there was difficulty coping with the small size of the dataset. We expected each pair to occur multiple times in the dataset, however, the vast majority of pairs occurred only once. Hence, it was unviable to average the "score" of each pair across multiple occurrences to determine an aggregate value. Therefore, by nature of the small dataset, the pair ranker is inherently inaccurate. Expanding the corpus of data would greatly increase the accuracy of the pair ranker and make the results more reliable.

3 NLP Sentiment Analysis Application

This NLP application is based on the reviews collected earlier and classifies the reviews into positive, neutral and negative classes. The following open source sentiment analysis tools were used in this application: NLTK, sklearn, Keras and Pytorch. Experiments were conducted on six models before choosing the best model based on its f1 score. The six models tested were Random forest classifier, linear SVC, Multinomial Naïve Bayes, Logistic Regression, bidirectional LSTM and BERT [1].

3.1 Preprocessing

The following steps were used in the preprocessing of the movie reviews:

(1) Generate the target sentiment labels. The dataset used had class imbalance with ratings largely on the positive side. To generate the target sentiment labels, the ratings were divided into [positive (7-10), neutral (5-6), negative (1-4)] ratings. Although the class balance is not fully restored as shown in fig 3.1, for the application of reviewing movies, other datasets are also skewed to positive ratings, hence the predictions follow the same distribution and the model should maintain its accuracy.
(2) Combine review titles and review text to a single column "review_combined". This maximizes the tokens and data available for the experiment.
(3) Tokenize. The words were tokenized using Python's NLTK library.
(4) Denoise. Removed punctuations, stop words and converted words to lowercase.
(5) Generate word embeddings. The words were vectorized with a TFIDF vectorizer from sklearn which was fit on the review dataset.
(6) Split into train and test set. Test set size was set to 0.2.
(7) Fit models. Used pipeline with 5-fold cross validation and the following models: Random forest classifier, linear SVC, Multinomial Naïve Bayes and Logistic Regression.

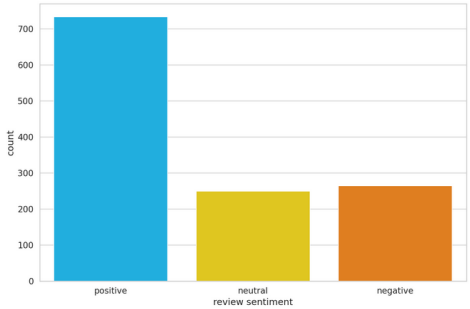


Fig 3.1 Count of sentiment labels

Preprocessing steps (5) and (7) for bidirectional LSTM with Glove embeddings and for BERT model were different:

- Generate word embeddings. The words were vectorized using the keras and BERT tokenizer (bert-based-case) respectively and padded until MAX_LENGTH=500.
- Fit model. The embedding matrix from the glove embeddings was created for the bidirectional LSTM and the pretrained BERT model was used. Both of their training parameters were 10 epochs and a batch size of 32.

3.2 Result Discussion

From the classical machine learning models, LinearSVC attained the highest score of 0.779. However, the model produced did not generalize well to context and to word negations such as "not good" because like the Naïve Bayes model, only the count of positive or negative words affects the prediction.

The single layer bidirectional LSTM attained an accuracy of 0.752. The bidirectional LSTM should be able to take in context of a sentence and perform well on word negation, however, even with glove word embeddings, it did not perform as well as the LinearSVC as the training dataset used is small with only 1200 reviews.

The last experiment used the pretrained, current state-of-the-art BERT model, which is a bidirectional transformer model. A dropout layer is used for regularization and a final fully connected layer is added for the classification output as shown in Fig 3.4. The output is the probabilities of each sentiment label.

```
class SentimentClassifier(nn.Module):

  def __init__(self, n_classes):
    super(SentimentClassifier, self).__init__()
    self.bert = BertModel.from_pretrained(PRE_TRAINED_MODEL_NAME)
    self.drop = nn.Dropout(p=0.3)
    self.out = nn.Linear(self.bert.config.hidden_size, n_classes)

  def forward(self, input_ids, attention_mask):
    _, pooled_output = self.bert(
      input_ids=input_ids,
      attention_mask=attention_mask
    )
    output = self.drop(pooled_output)
    return self.out(output)
```

Fig 3.2 BERT classifier with added output layer

The test accuracy obtained was 0.808 which is the highest of all the models tested. The accuracy is largely due to the pretrained weights of the BERT model as well as the transformer architecture that introduces attention to important words in the review. The application mostly misclassifies on the neutral sentiment as shown in fig 4.6. In fig 3.6, the negated bigram "not horrible" is predicted as having a sentiment of neutral whereas for the other classifiers the word "horrible" would have produced a negative output.

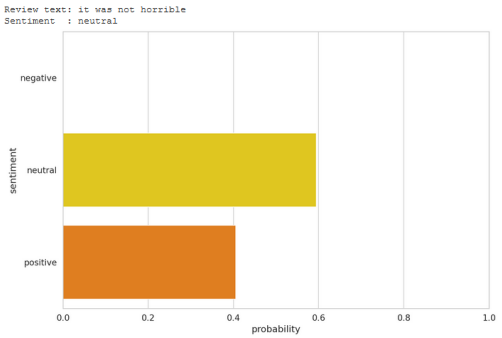


Fig 3.3 Prediction output probabilities

3.3 BERT model analysis

The BERT model is based on the transformer architecture [2] which consists of an Encoder and a Decoder. The Encoder takes an input sequence and maps it into an n-dimensional vector and that vector is fed into the Decoder which turns it into an output sequence which is sentiment class in this case.

The Transformer encoder reads the entire sequence of words at once. Therefore, it is bidirectional. This allows the model to learn the context of a word (attention mechanism). The attention-mechanism decides which inputs (words) are important by attributing different weights to those inputs. The Decoder will then take the encoded sentence and the weights provided by the attention-mechanism as its input. The position of each word in an input sequence is added to the embedded representation of each word since it is not captured naturally like in an RNN. In fig 4.8, the Encoder is on the left and the Decoder is on the right.

The pretrained BERT model was trained using the following method. 15% of the words in each sequence were replaced with a [MASK] token and then used as input to BERT. The model then learns to predict the original value of the masked words, based on the context provided by non-masked words in the sequence.

3.4 Evaluation of application and future work

By applying transfer learning, with a small dataset of 1200 reviews, a reasonable result has been achieved for the sentiment analysis application. In the future to improve the application, more experiments with the BERT model may be conducted to fine tune the hyperparameters or to acquire more data for the model. Additionally, research could be made into BERT models which were pretrained using different techniques.

4 REFERENCES

[1] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[2] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).

5 Appendix

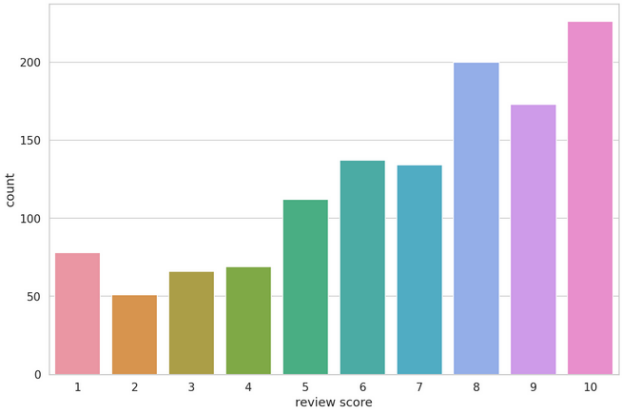


Fig 4.1 Movie review score distribution

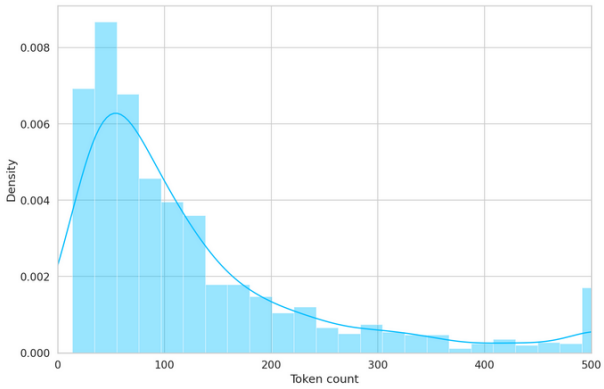


Fig 4.2 Token distribution of movie reviews

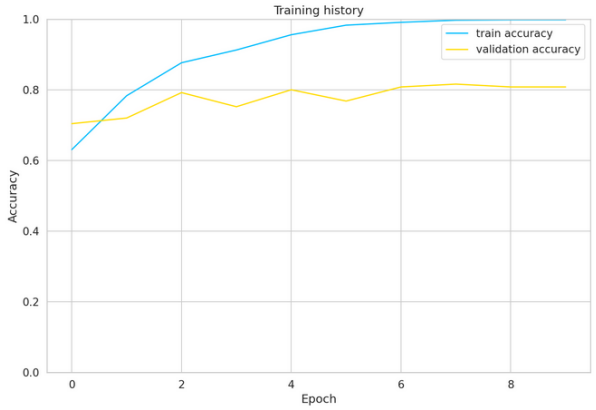


Fig 4.3 Training history of BERT

	precision	recall	f1-score	support
negative	0.83	0.74	0.78	27
neutral	0.52	0.52	0.52	25
positive	0.89	0.93	0.91	73
accuracy			0.81	125
macro avg	0.75	0.73	0.74	125
weighted avg	0.81	0.81	0.81	125

Fig 4.4 Accuracy metrics of BERT

```
model = keras.Sequential()
model.add(Embedding(num_words,
                    embedding_dim,
                    embeddings_initializer=Constant(embedding_matrix),
                    input_length=sequence_length,
                    trainable=True))
model.add(SpatialDropout1D(0.2))
model.add( Bidirectional( LSTM(50, dropout = 0.2, recurrent_dropout = 0.2)))
model.add(Dense(3, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])



epochs = 20
batch_size = 64
```

Fig 4.5 Bidirectional LSTM implementation

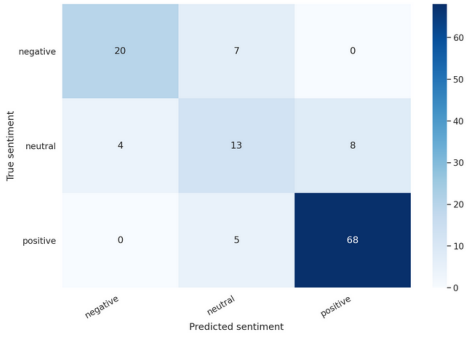


Fig 4.6 Classification heatmap for BERT test prediction

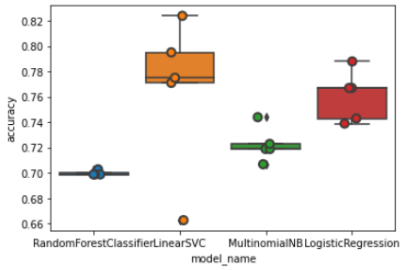


Fig 4.7 CV score of classical ML models

Detailed Analysis of BERT Model

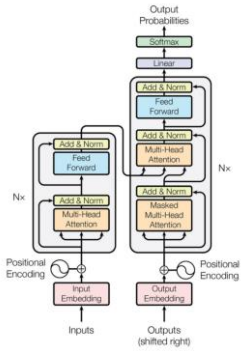


Fig 4.8 Transformer architecture

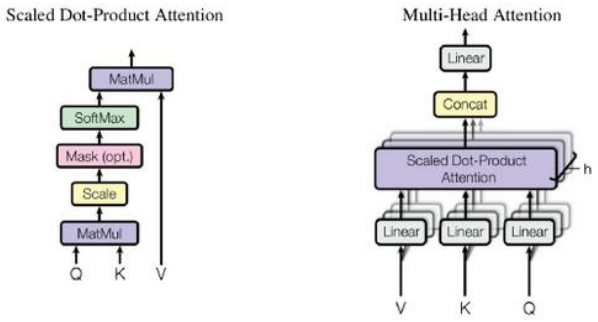


Fig 4.9 Multi head attention block

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Fig 4.10 Scaled dot product attention equation

$$a = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)$$

Fig 4.11 Attention weights equation

In the attention mechanism, the values in V are multiplied and summed with some attention-weights *a*. This means that the weights *a* are defined by how each word of the sequence (Q) is influenced by all the other words in the sequence (K). Finally, the SoftMax function is applied to the weights *a* to distribute the weights between 1 and 0. Those weights are then applied to all the words in the sequence that is in V and the result is passed to the Decoder.

6 CONTRIBUTIONS

Name	**Contribution**
Chen Sihao	Section 1: Analysis for datasets CORD-19 and ASCII math equations. Answered respective questions for part 1.
Ng Kai Chin	Section 2: Design of pair ranker algorithm, compiled analysis and results into report.
Thong Hoi Wei	Section 2: Analysis of dataset and design of pair ranker algorithm.
Yao Cheng Hui	Section 1: Analysis for dataset 'Financial tweets'. Answered respective questions for part 1. Compiled analysis into report.
Zachary Chua Chee Kian	Section 3: Developed and tested the NLP application and wrote the report on the NLP application.