






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	Contribution	Signature	Date
Chua Peng Shaun U1821442G	20% Question 1		1/11/2020
Ding Si Han U1821300C	20% Question 2, 3		1/11/2020
Gabriel Sze Whye Han U1821069G	20% Question 2, 3		1/11/2020
John Lim Jin U1822862E	20% Question 1		1/11/2020
Tan Chuan Xin U1821755B	20% Question 2, 3		1/11/2020

Important note:

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

CZ4045 Assignment 1 Report

Chua Peng Shaun
PCHUA014@e.ntu.edu.sg

Ding Si Han
DING0111@e.ntu.edu.sg

Gabriel Sze Whye Han
GSZE001@e.ntu.edu.sg

John Lim Jin
JOHN0036@e.ntu.edu.sg

Tan Chuan Xin
CTAN184@e.ntu.edu.sg

ABSTRACT

The field of Natural Language Processing (NLP) has countless real-world applications and is widely incorporated wherever text or speech can be found. In this paper, we describe several fundamental techniques in NLP that lay the groundwork to achieve more complex NLP tasks, their effectiveness on actual datasets and propose potential improvements to these methods.

CCS CONCEPTS

• Computing methodologies → Natural Language Processing, Machine Learning

KEYWORDS

Natural Language Processing; Domain-Specific Text Analysis; Noun-Adjective Pair Ranking; Sentiment Analysis

1. DOMAIN SPECIFIC DATASET ANALYSIS

Three domain specific datasets were formed for analysis. Each dataset consists of 20 documents in one selected topical domain and each domain has its own linguistic characteristics with some specific terms specific to the domain.

1.1 Choice of Domains

1.1.1 Legal Documents

Legal documents aim to set out terms and conditions for a contract or for an entity to pursue legal action, and thus, wording has to be precise and unambiguous. This makes the sentence structure as well as the diction used in legal documents domain specific. Legal documents use some uncommon vocabulary such as 'hereby', 'whereby' and 'henceforth' are used in legal documents, but not commonly found elsewhere.

1.1.2 Instruction Manuals

Instruction manuals aim to guide the user to understand the product they have bought, how to use it and maintain it. Because it is meant for use by the general public, the language used is simple, and sentences short. Since it is in the manufacturer's interest to cover as many error cases as possible extensively, manuals are lengthy and exhaustive. Instruction manuals may include technical terms relating to the product.

1.1.3 Biomedical Journals

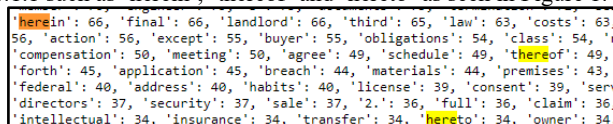
Biomedical journals aim to report and provide an overview of biomedical applications and updates to allow scientists to better communicate with one another. As such, biomedical journals consist of prevalent usage of technical terms like drug names, medical conditions and chemical compounds. This makes the diction used in biomedical journals highly specific.

1.2 Tokenization

Tokenization was performed using the nltk toolset. For preprocessing, each token was first subjected to removal of most punctuation. Stop words such as determiners like 'the' or 'to' were removed as well as standalone numbers as they would not add any value to understanding domain specific tokens.

1.2.1 Legal Documents

The tokenizer recognized the domain specific tokens mentioned in 1.1.1 such as 'herein', 'thereof' and 'hereto' as seen in Figure 1.



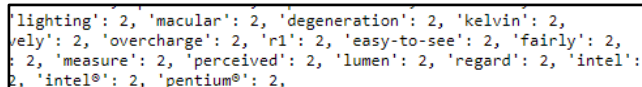
```
herein': 66, 'final': 66, 'landlord': 66, 'third': 65, 'law': 63, 'costs': 63, 'action': 56, 'except': 55, 'buyer': 55, 'obligations': 54, 'class': 54, 'compensation': 50, 'meeting': 50, 'agree': 49, 'schedule': 49, 'thereof': 49, 'forth': 45, 'application': 45, 'breach': 44, 'materials': 44, 'premises': 43, 'federal': 40, 'address': 40, 'habits': 40, 'license': 39, 'consent': 39, 'ser': 37, 'directors': 37, 'security': 37, 'sale': 37, '2.': 36, 'full': 36, 'claim': 36, 'intellectual': 34, 'insurance': 34, 'transfer': 34, 'hereto': 34, 'owner': 34
```

Figure 1. Tokenization of domain-specific tokens

In the case of 'company/developer', the tokenizer does not recognize that the words before and after the '/' are separate, and instead tokenize them as 1 token. A failure to tokenize them as 2 separate entities means that if a tokenizer was used to search for 'developer', several clauses of importance would be missed out on. The tokenizer can be improved by splitting on any character that is non-alphanumeric, so that 'company/developer' can be avoided, and instead be tokenized as 'company' and 'developer', and underscores and headers can be removed.

1.2.2 Instruction Manuals

From Figure 2., the tokenizer picks up headers such as 'c1', 'c2', dot separators such as '.....' and because of formatting, some words are cut off such as 'run' and '-ning'. Lastly, there are incorrectly spelt words (dataset, not tokenizer issue). We can consider words which have a dash in between, if the dash is removed and the result is a word found in the dictionary, we remove the dash and tokenize it, otherwise, it is left as-is. We can also cut out headers, dot separators, and incorrectly spelt words. Overall, however, the tokenizer has done well in tokenizing domain specific tokens, such as 'lcd' and other abbreviations.

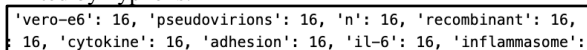


```
lighting': 2, 'macular': 2, 'degeneration': 2, 'kelvin': 2, 'vely': 2, 'overcharge': 2, 'r1': 2, 'easy-to-see': 2, 'fairly': 2, 'measure': 2, 'perceived': 2, 'lumen': 2, 'regard': 2, 'intel': 2, 'intel': 2, 'pentium': 2,
```

Figure 2. Tokenized domain specific terms

1.2.3 Biomedical Journals

It is observed in Figure 3. that the tokenizer was able to correctly identify medical terms and technical jargon as well as terms delimited by hyphens.



```
'vero-e6': 16, 'pseudovirions': 16, 'n': 16, 'recombinant': 16, 'cytokine': 16, 'adhesion': 16, 'il-6': 16, 'inflammasome':
```

Figure 3. Tokenization of domain specific terms

However, while abbreviated technical jargon was able to be successfully recognized, there may be difficulty in distinguishing

these abbreviations with shortened words when all the tokens are transformed to lower case.

Overall, the tokenizer performs well in identifying the domain-specific terms. However, a way to improve tokenization for the biomedical journal dataset might be to include uppercase words. While this may increase the number of tokens received and defeat the purpose of pre-processing, the trade-off may bring about an improvement as technical jargon such as abbreviated names often come in uppercase.

1.3 Stemming

For stemming of tokens, snowball stemmer from nltk was used.

1.3.1 Legal Documents

```
=====Legal Documents dataset=====
Number of distinct tokens for Legal Documents dataset: 4311
Number of distinct tokens after stemming for Legal Documents dataset: 2783
```

Figure 4. Token counts before and after stemming

In Figure 4., the number of distinct tokens dropped severely from 4311 to 2783, and this is likely due to repeated phrases common to legal language, such as ‘agreements’, ‘agreed’ and ‘agree’ as well as ‘indemnity’, ‘indemnatee’ and ‘indemnification’.

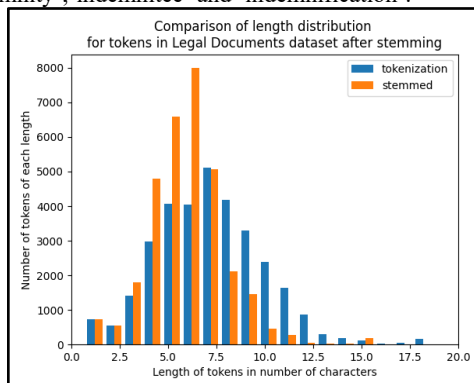


Figure 5. Length distribution before and after stemming

In Figure 5., we compare the length distribution for tokens before and after stemming. The mode of tokens before stemming is about 7 characters, and the mode after stemming is about 6 characters. The main effect of the stemmer is to shorten longer words, and we can see for tokens of length 8-15 characters, the stemmed tokens become much shorter, whereas tokens shorter than that are not affected. Looking at the numbers on the y-scale, the stemmer does reduce the length of many tokens, and this is likely because legal documents use longer words to avoid ambiguity.

1.3.2 Instruction Manuals

```
=====Instruction Manuals dataset=====
Number of distinct tokens for Instruction Manuals dataset: 6931
Number of distinct tokens after stemming for Instruction Manuals dataset: 5082
```

Figure 6. Token counts before and after stemming

We can see from Figure 6. that the number of distinct tokens falls slightly after stemming. A reason for this is that as said in tokenization above, there are many irrelevant tokens such as headers, model number, cut off words and incorrectly spelled words, which inflated the original token count. If we compare this to legal document stemming, the absolute reduction in token count is the same, but percentage wise is less than legal documents due to headers and model numbers.

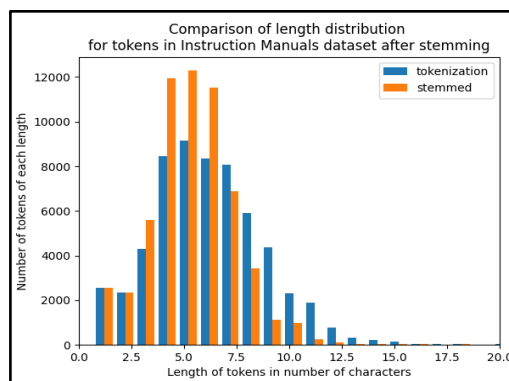


Figure 7. Length distribution before and after stemming

Comparing the effects of stemming, the general curve in Figure 7. is the same, however, the variance in token length is reduced. The mode stays pretty much the same, and we can see that only tokens above 7 characters are affected. This is likely because instruction manuals already use as concise a language as they can, and as it is for the common man, avoid using long and bombastic words. Hence, the mode token length is short at around 5-6 characters.

1.3.3 Biomedical Journals

```
=====Biomedical Journals dataset=====
Number of distinct tokens for Biomedical Journals dataset: 9967
Number of distinct tokens after stemming for Biomedical Journals dataset: 7589
```

Figure 8. Token counts before and after stemming

In Fig 8, the number of distinct tokens dropped from 9967 to 7589. This could be attributed to domain specific jargon being used and spelt in different forms throughout different medical journals as there is no form of standardization of terms. Biomedical journals also utilize many long descriptive words that can be shortened significantly during stemming.

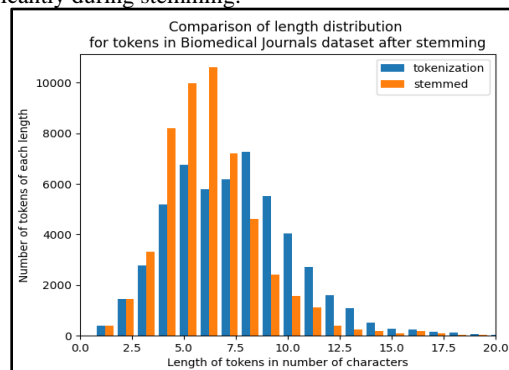


Figure 9. Length distribution before and after stemming

Like the previous two datasets, the effects of stemming can be seen with an increase in the number of tokens with length 7 and below in Figure 9.

1.4 Sentence Segmentation

```
Number of distinct sentences for Legal Documents dataset: 1967
Number of distinct sentences for Instruction Manuals dataset: 5246
Number of distinct sentences for Biomedical Journals dataset: 3465
```

Figure 10. Number of distinct sentences from each dataset

As seen from Figure 10., the Instruction Manuals dataset has the most distinct sentences. Biomedical journals and legal documents aim to be concise and are generally designed for experts, hence being able to assume jargon. Instruction manuals are for appliances which the layman purchases and needs to lay out instructions in the most understandable way. Furthermore, for complex technical equipment, the instruction manual can be extremely lengthy. (Camera manual was 200 pages long)

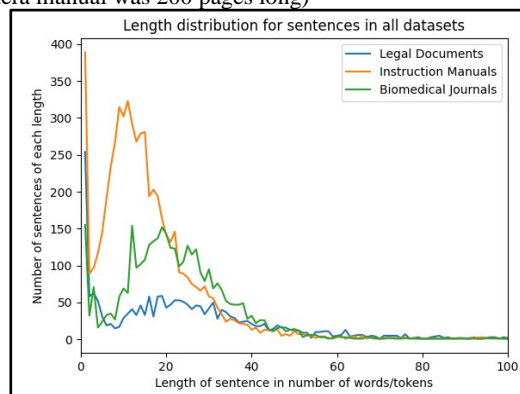


Figure 11. Length distribution of sentences across datasets

From Figure 11, the high number of very short sentences in all 3 datasets is likely due to headers ('3.1', '4.5.2 etc.') as segmentation is done via periods and should be ignored. For instruction manuals, the peak length distribution is the least at around 10, compared to 20 in the other 2 datasets. This is likely because they need to make their sentences understandable and concise, and the tradeoff is that they have a larger number of sentences. The legal documents dataset has a largest distribution of the 3, likely due to needing to make their contracts unambiguous, causing them to use longer sentences.

1.5 POS Tagging

1.5.1 Legal Documents

3 random sentences were picked for POS tagging analysis:

7.3 Independent Activities Members may engage in or possess an interest in other businesses of any nature or description, including, but not limited to, ownership, employment by, or financing, except for businesses that are similar to the business of the Company as determined by the Members.

We find that the POS tagger largely tags each word correctly, with a few exceptions. 'Independent Activities Members' is a proper noun, likely a special group mentioned earlier in the document. However, as it is not a common NNP such as New York, the POS tagger broke it down to its constituent words, instead of staying as one NNP.

(ab) Time is of the essence in this Agreement

The tagger incorrectly tags the '(ab)' as an adjective, when it should be used as an equivalent for a cardinal number (used for headers). Subsequently, it considered 'Time' as the second word in the sentence and tagged it as a NNP when it should have been considered an NN. Lastly, it tagged 'Agreement' as an NN instead of an NNP. As an NNP, 'Agreement' holds larger significance as it is referring to the agreement in question.

Free rent will be awarded in the form of a cash rebate following the next on-time rent received

For the last sentence, the POS tagger tagged all words correctly. We can see that for this dataset, the trend seems to be that the POS tagger struggles with differentiating between NN and NNP, especially that of NNP that are specific to the document in question, but performs well with all other instances so far. Thus, we find that the POS tagger does well with domain-specific terms.

1.5.2 Instruction Manuals

3 random sentences were picked for POS tagging analysis:

When traveling by air, carry the computer as hand luggage; do not check it in with the rest of your luggage.

In the first sentence, the only mis-tagged word is 'traveling'. It was tagged as NN instead of VBG.

For software not provided with this computer, you must either download the software from the manufacturer's website or reinstall the software from the media provided by the manufacturer.

In the second sentence, all words are tagged correctly.

What's in the Box Device OxygenOS Safety Information 46 Care and Maintenance Your phone is a product of superior design and craftsmanship and should be treated with care.

In the third sentence, all the capitalized words are tagged as NNP instead of NN, likely due to the tagger not being able to identify the beginning of a sentence. The tagger did not identify '\n' (in the raw data) as indicating a new line, thus it added what came after as a continuation of the sentence, instead of starting a new one. Hence, we see the unusual syntax of the sentence, which is actually 2 sentences joined by a header. One way to rectify this is to include in the sentence segmentation that any '\n' captured immediately begins a new sentence instead of delimiting using periods only.

Overall, the POS tagger works well with domain specific terms ('care and maintenance', 'reinstall' etc), however, if we can improve the sentence segmentation, we can improve the tagger's ability to classify things as NN and NNP correctly.

1.5.3 Biomedical Journals

3 random sentences were picked for POS tagging analysis:

At present, 24 western medicinal treatments are registered in clinical trials.

All tokens were tagged correctly by the POS tagger.

A recent cryo-EM analysis on the intermediate catalysis products of Cas12a [36] revealed that the enzyme utilizes a further three-checkpoint control to sense the hybridization between the crRNA and the DNA.

In this sentence, the tagger incorrectly tags 'cryo-EM' as adjective (JJ) when it should be a proper noun (NNP). It is worth noting that the medical abbreviation 'Cas12a' was correctly tagged as a proper noun (NNP). However, 'crRNA' and 'DNA' which are proper nouns (NNP) are incorrectly tagged as nouns (NN).

The coronavirus was officially named severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) by the International Committee on Taxonomy of Viruses based on phylogenetic analysis.

In this sentence, the tagger incorrectly tags ‘respiratory’ as a noun (NN) when it is an adjective (JJ) that pertains to respiration. ‘Sars-CoV-2’ was also incorrectly tagged as an adjective (JJ) when it should be a proper noun (NNP).

Overall, it is observed that the POS tagger seems to be able to identify domain specific terms including abbreviated medical terms. However, the POS tagger struggles with distinguishing hyphenated terms, tagging them as adjectives instead of nouns.

2. NOUN-ADJECTIVE PAIR RANKER

E-commerce sites, and online shopping in general, rely heavily on the network effect to improve product sales. Therefore, knowing a products’ strengths and weaknesses through reviews from other shoppers is very important when it comes to making a purchase decision. For example, Amazon summarizes what it believes are keywords for a product based on customer reviews.

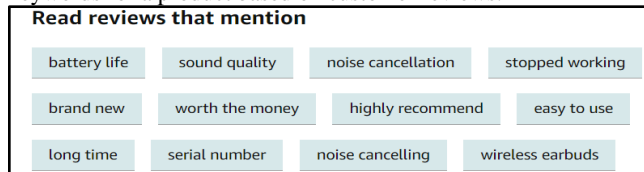


Figure 12. Keywords for a product

In this section, we will attempt to capture representative noun-adjective pairs as keywords for online reviews for a given product, in this case Apple AirPods. We will then rank these pairs according to their meaningfulness.

2.1 Noun-Adjective Pairs Identification

Our process for generating noun-adjective pairs is as follows:

- Manually identify 30 reviews
- For each review, use spacy senticizer to obtain sentences
- For each sentence, use regex on punctuations to get segment
- For each segment, use dependency parsing + POS tagging to obtain noun-adjective pairs

Through experimentation, we need to account for two primary types of nouns: singular nouns and compound nouns. However, the adjective is predominantly just one word, since adjectives are usually modified through adverbs

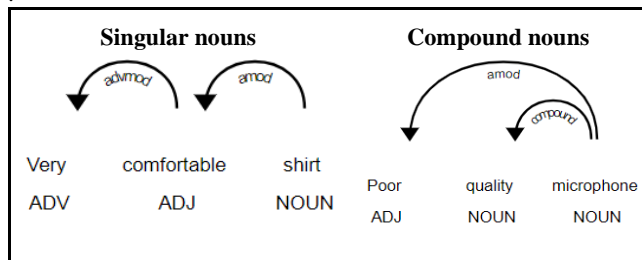


Figure 13. Singular and Compound noun example

Singular nouns are identified first through the dependency tag ‘nsubj’, and then pronouns like ‘he, she, they’ are filtered out by removing nouns with the POS tag ‘pron, det’.

```
nouns = [tok for tok in doc if tok.dep_ in ['nsubj']]
nouns = [tok for tok in nouns if tok.pos_ not in ('PRON', 'DET')]
```

Figure 14. Filtering singular nouns

Compound nouns are identified by locating the first word with the dependency tag of ‘compound’, and joining it with its parent (the ‘nsubj’).

```
compounds = [tok for tok in doc if tok.dep_ == 'compound']
compounds = [c for c in compounds if c.i == 0 or doc[c.i - 1].dep_ != 'compound']
...
noun = doc[tok.head.i : tok.head.i+1]
```

Figure 15. Filtering compound nouns

Adjectives are identified through words that have a noun as its parent in the dependency tree, and carry the POS tags of ‘amod, acl, acomp’.

```
right_adj_list = [adj for adj in noun.root.head.rights if adj.dep_ in ['amod', 'acl', 'acompany']]
left_adj_list = [adj for adj in noun.root.head.lefts if adj.dep_ in ['amod', 'acl', 'acompany']]
```

Figure 16. Filtering adjectives

2.2 Meaningfulness

There are many ways to determine if a noun-adjective pair is meaningful, and our ranking solution takes a weighted average on the various factors to derive a final score that approximates the meaningfulness of a noun-adjective pair. The following factors below highlight and justifies our definition of meaningfulness.

2.2.1 Term Frequency-Inverse Document Frequency (TFIDF)

TFIDF is a statistical reflection of how important a word is to a document in each corpus. It penalizes words that appear frequently across a corpus and emphasizes words that are unique to a document. This is the basis of our metric, on the assumption that unique words to a specific review carry the most meaning. We treat each review as a document, and all reviews collectively as a corpus and utilize sklearn’s feature extraction package to perform TFIDF analysis.

```
tfidfVectorizer=TfidfVectorizer()
tfidf = tfidfVectorizer.fit_transform(df['Reviews'].tolist())
feature_names = tfidfVectorizer.get_feature_names()

dense = tfidf.todense()
denselist = dense.tolist()
tf_df = pd.DataFrame(denselist, columns=feature_names)
```

Figure 17. TFIDF calculations

$TFIDF_{score} \in [0.0 \text{ (very frequent)}, > 0.0 \text{ (largest = rarest)}]$

Figure 18. TFIDF score range

We will apply multipliers to the TFIDF score based on the other contributing factors towards meaningfulness.

2.2.2 Lemmatization

Rare words are very important for their unique take on a product, but frequently occurring words provide important consensus.

Lemmatization is the process whereby a word is reduced to its base form through vocabulary and morphological analysis. This provides a more linguistically accurate representation of the base form of a word versus stemming, e.g. *caresses* - *caress* (lemma) versus *caresses* - *care* (stem). Words in noun-adjective pairs will

be lemmatized to obtain their base forms, which can then be grouped together to obtain a frequency of the most common pairs.

2.2.3 Word Embedding

Word embeddings will improve on lemmatization. Product reviews also often contain many forms of the adjectives ‘good, bad’ for the same noun, such as ‘great/nice/poor design’, etc. These words all have the same inherent meaning, just expressed differently. We will find the word embeddings of the adjectives to reduce words similar to ‘good, bad’ to just ‘good, bad’, thereafter performing a grouping operation and assigning more weight to the most frequently occurring reduced noun-adjective pairs.

$$Mult_{embedding\ frequency} \in [1.0\ (one\ instance), 1.4\ (most\ instances)]$$

Figure 19. Word embedding multiplier range

2.2.4 Rating (Stars)

The rating, or number of stars a reviewer leaves for a product is an indicator of their opinion of the product. The more polarizing their opinion, the more meaning it carries as they are more likely to have strong opinions about the product versus a neutral review. Therefore 1, 5 star reviews will be more meaningful, while 3 star reviews less so. However, ratings are often given without much thought, hence a smaller multiplier should be given to ratings.

$$Mult_{review\ rating} \in [1.0\ (rating\ 3), 1.2\ (rating\ 1\ |\ 5)]$$

Figure 20. Review rating multiplier range

2.2.5 Length

Reviews are often written at no benefit to the reviewer. Therefore, we hypothesize that the length of the review is proportional to the effort and rigor of the review. Therefore noun-adjective pairs in longer reviews carry more meaning as they were well thought-out.

$$Mult_{review\ length} \in [1.0\ (shortest), 1.5\ (longest)]$$

Figure 21. Review length multiplier range

2.2.6 Ranking Algorithm

We will apply a multiplier on the TFIDF score of a given noun-adjective pair for each of the other factors in 2.1.2-2.1.5 to obtain a final meaningfulness score. Pairs with the highest score will be the most ‘meaningful’ for the product.

$$Score = TFIDF_{score} * Mult_{embedding\ frequency} * Mult_{review\ rating} * Mult_{review\ length}$$

Figure 22. Ranking algorithm score calculation

2.3 Top 5 Most Meaningful Pairs

We follow the definition of meaningfulness laid out in 2.2 and arrive at the following conclusions.

2.3.1 Self-Annotation Top 5

Our manual annotation of the 30 reviews identifies the following 5 to be the most meaningful noun-adjective pairs.

- sound quality - great (*and other synonyms*)
- battery life - long (*and other synonyms*)
- microphone - terrible (*and other synonyms*)
- quality control - unacceptable
- connection - faster

2.3.2 Algorithm Top 5

Applying the ranking algorithm in 2.2.6 yields these top 5 pairs (combined_pairs column).

Table 1. Top 5 scoring combined_pairs

review_length	rating	combined_pairs	word_embedding_pairs	tfidf_score	final_score
137	5.0	[(feature), nifty]	[feature, nifty]	0.255384	0.306461
227	5.0	[(quality), great]	[quality, good]	0.173079	0.297266
138	1.0	[(quality, microphone), Poor]	[quality microphone, bad]	0.237326	0.284862
137	5.0	[(sound), awesome]	[sound, good]	0.196022	0.235227
445	5.0	[(people), confused]	[people, confused]	0.174431	0.225315

2.3.3 Comparison

We observe the following correspondence between the two sets.

Table 2. Top 5 scoring combined_pairs

Manual	Algorithm
sound quality - great	feature - nifty
battery life - long	quality - great
microphone - terrible	quality microphone - poor
quality control - unacceptable	sound - awesome
connection - faster	people - confused

The algorithm has interestingly captured ‘feature - nifty’ and ‘people - confused’, which we did not look at during our manual annotation. This is due to the underlying TFIDF score, which increased the importance of these rarely seen words. ‘people - confused’ is an interesting finding, and upon looking deeper into the review, and the product, we found that consumers were indeed confused as to which generation of AirPods this listing was for. Evidently, our algorithm has managed to turn up something valuable that our manual annotation has missed.

The algorithm has also picked up on our 1st and 3rd most meaningful pairs, as highlighted in green. These were propped up by their word embedding multiplier, due to their relatively high occurrence providing an offset for their lower TFIDF scores.

Further analysis on why “battery life - long” was not captured revealed that the TFIDF score for “battery life - adjectives” tend to be lower due to their frequency of occurrence, but did not appear enough to warrant a high word embedding multiplier.

Of surprise was “quality control - unacceptable” not being captured. Its score was severely penalized by the very frequent appearance of the word “quality” in general and could not be picked up among the many instances of “sound quality”.

2.4 Challenges and Improvements

Laboriously identifying noun-adjective pairs to craft the language rules was challenging. Identifying the appropriate syntactic structures that must be evaluated to capture noun-adjective pairs required a deep look correctly and accurately into the dependency parse tree and POS tag sequence of every sentence and segment in our dataset. We eventually discovered that most pairs fall into the two categories outlined in 2.1, but it required significant manual labor to determine this relation. This provides significant impetus to employ machine learning to perform such tasks and will likely yield more accurate and reliable results versus painstaking crafting manual rulesets.

Algorithmically ranking the top 5 pairs proved to be a challenge, as there very few publicly searchable algorithms. While we can identify individual contributing factors like rating and TFIDF score, crafting the ideal blend of weights for the different factors to achieve a good ranking is a far more difficult task. We have only implemented a simple multiplier algorithm based on weights that we have manually fine-tuned to provide the most logical results. However, creating a better ranking algorithm requires machine learning, artificial intelligence, and significant statistical analysis, not unlike Google's PageRank problem.

3. APPLICATION - SENTIMENT ANALYSIS

In this section, an application to predict the Amazon review ratings and analyze sentiment was built using the same set of reviews extracted from Amazon AirPods.

To train a robust model, the Amazon Review Dataset (Cell Phone and Accessories subset) was retrieved for three reasons. Firstly, there were up to 1,128,437 reviews for the model to be trained on. The size of the dataset is extremely important to build a robust model especially for NLP applications, to reduce the effect of sparsity problems. Secondly, the number of the multiclass sentiment labels for the Amazon Review Dataset was 5, which matched the number of categories in the Amazon Airpod Reviews (Ranging from 1 to 5 Stars). Lastly, AirPods belong to this product domain, hence we have obtained a domain-specific dataset, which will likely improve our accuracy in training. For speed of performance, we truncate the dataset to 150,000 rows.

To build a reliable application, three different separate models were tested. Firstly, the Multinomial Naive Bayes (MNB) which operates on the underlying principle of Bayes Theorem. Secondly, a Stochastic Gradient Descent Classifier (SGDC), which utilizes a Support Vector Machine and harnesses the gradient descent algorithm for training. Thirdly, a Linear Support Vector Classifier (LSVC), which is another model that is based on Support Vector Machines.

3.1 Amazon Review Rating Prediction

The three models were first trained on the Amazon Review dataset and subsequently evaluated on the 30 Amazon AirPods Reviews. To analyze the effect of n-gram selection, the following combination of unigram, bigram and trigrams were tested and yielded the following results.

Table 3. Amazon Review Rating Accuracy

N-gram	LSVC	MNB	SGDC
(1, 1)	0.57	0.53	0.67
(2, 2)	0.6	0.47	0.67
(3, 3)	0.67	0.23	0.67
(1, 2)	0.63	0.57	0.70
(1, 3)	0.67	0.57	0.63
(2, 3)	0.53	0.37	0.60

From the results above, it is evident that the SGDC model yielded the best accuracy when unigrams and bigrams were used as inputs to the model. It is interesting to note that amongst the best scores for each model, one common factor was that most of these models

included unigrams. This is aligned with the fact that bigrams and trigrams would have a more significant sparsity problem in comparison to the unigrams, and thus the inclusion of unigrams would reduce the magnitude of the sparsity problem.

The best SGDC model trained on unigrams and bigrams, was further analyzed using a confusion matrix to examine which classes the model was accurate in predicting.

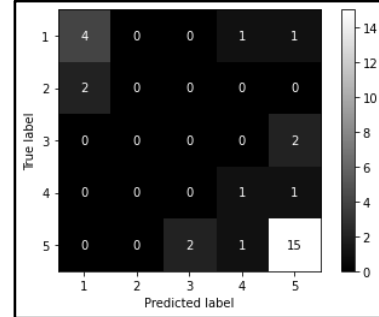


Figure 23. Amazon Review Review Rating Confusion Matrix

It is evident from the confusion matrix that the majority of the negative 1 star ratings and positive 5 star ratings were accurately predicted. However, it can be observed that the SGDC model still has difficulties evaluating ratings 2 and 4 as evidenced from the five confused values around the extreme true positive predictions.

3.2 Sentiment Analysis

To analyze further, a separate sentiment model was built as part of this application to classify negative, neutral and positive sentiment. These sentiments were categorized according to the ratings in Section 3.1 which ranged from [1, 5]. This categorization is depicted in the table below.

Table 4. Sentiment Classification based on Ratings

Sentiment Category	Sentiment	Ratings
0	Negative	1-2
1	Neutral	3
2	Positive	4-5

The same training and evaluation were implemented on the new sentiment labels, whereby 0, 1, 2 corresponded to negative, neutral and positive sentiments. The following was the results for the various models using the same n-gram permutations.

Table 5. Amazon Reviews Sentiment Analysis Accuracy

N-gram	LSVC	MNB	SGDC
(1, 1)	0.73	0.60	0.87
(2, 2)	0.80	0.70	0.80
(3, 3)	0.80	0.20	0.73
(1, 2)	0.83	0.83	0.83
(1, 3)	0.80	0.73	0.83
(2, 3)	0.73	0.53	0.80

From the above table, it can be observed that the Stochastic Gradient Descent Classifier with unigrams yielded the best accuracy of 0.87. This is significantly higher than the 0.70 accuracy

in the Amazon Review Rating Prediction in Section 3.1. This is likely to be due to two reasons. Firstly, the number of classes to predict on has decreased from 5 to 3. Secondly, the misclassification of good reviews, for 4 Stars and 5 Stars, is eliminated as the two categories are combined into one. This can be corroborated by examining the confusion matrix shown below.

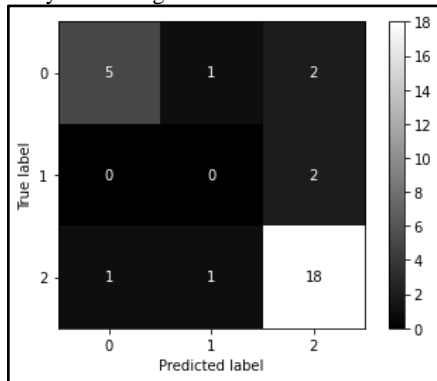


Figure 24. Amazon Reviews Sentiment Analysis Confusion Matrix

One notable observation is that the model did not have any true positive predictions for reviews with label 1, which corresponds to neutral. This was an expected finding, as out of the three classes in the truncated Amazon Reviews Dataset (Negative[1-2], Neutral[3], Positive[4-5]), the most frequent occurring sentiment was positive with 112,764 reviews, followed by neutral and negative with 15,905 and 21,331 respectively. Imbalanced dataset would typically favor the prediction of the most frequent class.

To further understand why this happened, these two neutral examples were analyzed qualitatively. It was observed that one of them included phrases such as “high quality sound” and “impressive battery life”, while the other included “not satisfied” and “it seems expensive”, thus causing them to be classified as a positive and negative review respectively..

3.3 Sentiment Analysis Application

The application can be tested by running the Python File `3_app.py`. The application will allow a user to input a sentence and it will generate the sentiment based on the user’s input. There is also an option to view our predictions for the sentiment of the 30 Airpod reviews we have shortlisted. The sentiment model is based on the model with the highest accuracy for sentiment analysis is Section 3.2: The Stochastic Gradient Descent model trained with unigrams.

3.4 Further Improvements

To further improve this application, there are two key areas that can be further explored.

Firstly, a larger evaluation dataset could be extracted and used for the Amazon Airpod Reviews. Currently, there are 30 Amazon Reviews and a greater number of reviews would increase the reliability of the model in this multiclass prediction.

Secondly, we could increase the amount of training data used from the truncated set of 150,000 rows. However, this will incur significant computing costs due to the training time required.

Lastly, we can handle the imbalanced class distribution in the Amazon Reviews Dataset through oversampling underrepresented classes or undersampling the overrepresented classes.

4. CONCLUSIONS

For the first segment, tokenizers generally work well, but include a large share of noise. To improve this process, we can include uppercase words to show abbreviations clearly, and remove alphanumerics. Moreover, stemmers usually cut tokens longer than 7 characters, and context is crucial in such situations. For example, legal documents are mostly affected, since they use longer morphemes on average.

Analyzing techniques such as sentence segmentation shows trends in datasets. For example, instruction manuals have many short sentences, while legal documents have the most distributed sentences. The POS tagger also works well except when it comes to NN and NNP, due to abbreviations and common nouns becoming NNP varying in each document. A solution to rectify this is to improve sentences segmenter to divide correctly as headers are included which confuses the NNP assignment.

For the second segment, identifying pair rankers proved to be a challenge depending on varying strengths of different libraries and reviews which may affect the outcomes. It is apt to first define what “meaningful” reviews entails and to pragmatically simplify that into a rating algorithm. These may vary across opinions but should be intuitive and fundamental in understanding. However, machine learning methods may be preferable to our rule-based pair identification method, and statistical analysis can be employed for a better ranking system.

The sentiment analysis segment was exploratory and experimental, harnessing trial and error with various models (and testing various datasets). The results largely can be justified with rudimentary understanding of machine learning and can largely differ based on the type of models and datasets selected.

5. REFERENCES AND CITATIONS

- [2] Docracy. (n.d.). Free Legal Documents. Retrieved October 29, 2020, from <https://www.docracy.com/>
- [3] Manuals Online. (n.d.). Free User Manuals and Owners Guides. Retrieved October 29, 2020, from <http://www.manualsonline.com/>
- [4] Biomedical Journal. (n.d.). Retrieved November 01, 2020, from <https://www.journals.elsevier.com/biomedical-journal/most-downloaded-articles>
- [5] Amazon (n.d.). *Apple AirPods with Charging Case*. Product website. Retrieved October 01, 2020 from https://www.amazon.com/Apple-AirPods-Charging-Latest-Model/dp/B07PXGQC1Q/ref=sr_1_4?dchild=1&keywords=airpods&qid=1601361680&sr=8-4#customerReviews
- [6] Spacy (2020). *Annotation Specifications*. Retrieved October 01, 2020 from <https://spacy.io/api/annotation>
- [7] JianMo Ni (2018). *Cell_Phones_and_Accessories_5.json*. Dataset. Retrieved November 01, 2020, from <https://nijianmo.github.io/amazon/index.html#subsets>

6. APPENDIX - PYTHON LIBRARIES

```
[1] nltk
    nltk stopwords
    nltk averaged_perceptron_tagger
    nltk wordnet
[2] string
[3] glob
[4] os
[5] random
[6] numpy
[7] collections
[8] matplotlib
[9] re
[10] pandas
[11] spacy
    spacy en_core_web_lg
[12] sklearn
[13] tqdm
[14] json
[15] google.colab
```