

Explainable Text Classification: Improving lexicon-based classification using n-grams

Jiulin Zong

A report submitted for the course
COMP4560

Supervised by: Professor Kerry Taylor
Dr. Priscilla Kan John
The Australian National University

June 2021

© Jiulin Zong 2021

Except where otherwise indicated, this report is my own original work.

Jiulin Zong
6 June 2021

Acknowledgments

First of all, I would like to show my deepest gratitude to my supervisors, Dr. Priscilla Kan John and Prof. Kerry Taylor, for their careful guidance of my project in the past several months, which greatly improved my understanding of academic writing and taught me a lot of specific research skills. Thank you for taking time every week to guide my study and research. Your agreement to guide my project is a great honor to me.

Secondly, I would like to thank Shandong University (Weihai) and Australian National University for giving me a broader view of the world. The staff I met in these two universities are very serious and responsible. They taught me many thing. I am grateful to Shandong University (Weihai) for gaving me a chance to change my major and join the 2 + 2 program. And I also need to thank Australian National University for providing me with the opportunity to communicate with the best students and teachers in the world.

I am also grateful to my best friend Shixuan Liu, who gave advice for both my research carrying out and report writing. He also participated in my dataset sentiment marking. I am also indebted to my friends Peng Zhang and Yuchi Liu, for giving sentiment marks for my dataset. Thank you for taking time out of your busy schedule to help with my project.

Thanks to my family for their support and encouragement. I've been here till today. Thank my mother for being able to endure my shortcomings and encourage me to move forward again and again. Thanks to my father, you are the toughest person I have ever met. When I am anxious, you can always comfort me and lead me out of trouble. In addition, I would also like to thank my grandmother. You have been with me since I can remember and given me meticulous care.

Thanks to all the friends from Shandong University and Australian National University. You accompanied me to spend boring time, discuss and solve problems together, and overcome difficulties. Thanks to Jiahao Zhang and Hang Zhang, you have given me great help in my university study. Without your help, I can not make it this far. Thanks to my former roommates, Yuhang Liu, Boyu Liu, Wenjie Liu, Yuchen He and Chaoyi Yang, who studies at Shandong University (Weihai). I will always remember the happy time with you.

Abstract

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. This classifier has a manually-built lexicon with more than 7000 words. Each word was rated by 10 individuals, and the mean of these scores will be the final score for sentiment analysis. For the manually rated score, -4 means the most negative, 0 means neutral and 4 denotes the most positive. In this project, we want to design an experiment to extend a lexicon-based classifier named VADER to investigate how using n-grams will affect its performance.

VaderSentiment includes python code for rule-based sentiment analysis engine. The grammar and syntax rules described in this paper are implemented, and the influence of each rule on the emotion perception intensity in sentence level text is analyzed by applying the empirical quantitative method. Importantly, these heuristics go beyond what is usually captured in a typical bag of words model. They contain relationships between terms that are order sensitive. We will try to apply n-gram and different types of methods to vaderSentiment, and observe their performance on prepared data set which includes manually marked sentences. We will compare the output with those executed by the original vaderSentiment and analyze the reasons for getting a better or worse result.

keywords: Lexicon-based, Sentiment Analysis, N-gram, VADER, vaderSentiment, $\overline{\text{IDF}}$, TF-IDF, Minkowski Distance

Contents

Acknowledgments	iii
Abstract	v
1 Introduction	1
1.1 Problem Statement	1
1.2 Motivations	2
1.3 Project Scope	2
1.4 Report Outline	3
2 Background and Related Work	5
2.1 Background	5
2.2 Related work	6
2.2.1 Automatic Text Classification	6
2.2.2 Sentiment Analysis	6
2.2.3 Text Classification	7
2.2.4 Lexicon	7
2.2.5 Lexicon-Based Classifier	7
2.2.6 N-gram	8
2.2.7 Language Model	9
2.3 Summary	9
3 Design and Implementation	11
3.1 Design	11
3.1.1 Dataset	11
3.1.2 Coding	11
3.1.3 IDF	12
3.1.4 TF-IDF	12
3.2 Summary	13
4 Experimental Methodology	15
4.1 Software platform	15
4.2 Hardware platform	15
4.3 Data Used	16
4.4 vaderSentiment	17

5	Results	21
5.1	Measure Methods	21
5.1.1	Minkowski Distance	21
5.2	Output	22
5.3	Summary	24
6	Conclusion	27
6.1	Future Work	27
7	Appendix	29
7.1	Negate	29
7.2	Booster	29
7.3	Sentiment Laden Idioms	30
7.4	Special Cases	30
7.5	TF-IDF Implementation	31
7.6	Artifact	32
	Bibliography	33

List of Figures

1.1	Comparison of results of different word order	1
1.2	A simple sample of tokenizer based on n-gram	2
3.1	Formula of IDF	12
3.2	Formula of TF	12
4.1	Example of datasets(with markings)	17
4.2	My figure of structure of vaderSentiment	19
5.1	Formula of minkowski distance	21
5.2	Formula of manhattan distance	22
5.3	Formula of euclidean distance	22
5.4	Formula of chebyshev distance	22

List of Tables

4.1	Software Information	15
4.2	Hardware Information	16
5.1	The performance of different methods on the story dataset	23
5.2	The performance of different methods on the news dataset	23
5.3	The performance of different methods on the sci-fi dataset	24
5.4	The performance of different methods on the total dataset	24
7.1	Vocabulary for negative words	29
7.2	Vocabulary for booster words	30
7.3	List for sentiment laden idioms	30
7.4	List for special case idioms and phrases	30

Introduction

VADER [Hutto and Gilbert, 2014] is a sentiment analysis tool that is specifically attuned to sentiments expressed in social media. It has a detailed lexicon for words, punctuation and emojis. However, the paragraph tested is mainly divided into different words and calculating their weights according to the previous negates and boosters, then summing them up for final scores of sentences. For the score of paragraph, it just sums the scores of different sentences together. In this project we try to change the step of simply adding scores together, and give the sentiment scores of previous words an important role in the final mark.

1.1 Problem Statement

```
I like you very much----- {'neg': 0.0, 'neu': 0.615, 'pos': 0.385, 'compound': 0.3612}
I like you just so so----- {'neg': 0.0, 'neu': 0.667, 'pos': 0.333, 'compound': 0.3612}
I like you a little----- {'neg': 0.0, 'neu': 0.615, 'pos': 0.385, 'compound': 0.3612}
I like you----- {'neg': 0.0, 'neu': 0.444, 'pos': 0.556, 'compound': 0.3612}
I hate you just so so----- {'neg': 0.425, 'neu': 0.575, 'pos': 0.0, 'compound': -0.5719}
I hate you----- {'neg': 0.649, 'neu': 0.351, 'pos': 0.0, 'compound': -0.5719}
I hate you very much----- {'neg': 0.481, 'neu': 0.519, 'pos': 0.0, 'compound': -0.5719}
I hate you a little----- {'neg': 0.481, 'neu': 0.519, 'pos': 0.0, 'compound': -0.5719}
AVERAGE SENTIMENT FOR PARAGRAPH: 0.0
```

(a) Output of sentences in original order

```
I very much like you----- {'neg': 0.0, 'neu': 0.59, 'pos': 0.41, 'compound': 0.4173}
I you like just so so----- {'neg': 0.0, 'neu': 0.667, 'pos': 0.333, 'compound': 0.3612}
I you like a little----- {'neg': 0.0, 'neu': 0.615, 'pos': 0.385, 'compound': 0.3612}
You like I----- {'neg': 0.0, 'neu': 0.444, 'pos': 0.556, 'compound': 0.3612}
I hate just so so you----- {'neg': 0.425, 'neu': 0.575, 'pos': 0.0, 'compound': -0.5719}
I you hate----- {'neg': 0.649, 'neu': 0.351, 'pos': 0.0, 'compound': -0.5719}
Very much hate I you----- {'neg': 0.499, 'neu': 0.501, 'pos': 0.0, 'compound': -0.6096}
I you a little hate----- {'neg': 0.46, 'neu': 0.54, 'pos': 0.0, 'compound': -0.5279}
AVERAGE SENTIMENT FOR PARAGRAPH: 0.0
```

(b) Output of sentences with words in random order

Figure 1.1: Comparison of results of different word order

In the original VADER, sentiment scores are calculated separately and then summing them up in for a final scores. This works well for single word(tag), short phrases and paragraphs with sentences that are not related, like the collection of comments. Here we changed the order of words in sentences and found that the results changed

slightly. Which means that the order of words does not influence much in the sentence score.

1.2 Motivations

One of the most obvious improvement of n-gram compared with unigram is that it makes previous words in count. Just like the figures shown below:

$S = \text{'I love deep learning'}$

$Y1 = \{\text{'I', 'love deep', 'learning'}\}$

$Y2 = \{\text{'I love', 'deep', 'learning'}\}$

$Y3 = \{\text{'I', 'love', 'deep learning'}\}$

$p(Y1) = p(I)p(\text{love deep} \mid I)p(\text{learning} \mid \text{love deep})$

$p(Y2) = p(I \text{ love})p(\text{deep} \mid I \text{ love})p(\text{learning} \mid \text{deep})$

$P(Y3) = p(I)p(\text{love} \mid I)p(\text{deep learning} \mid \text{love})$

Figure 1.2: A simple sample of tokenizer based on n-gram

In the example, the probability of one word depends on the previous ones, and if we changes the order of words the result will change with it. Thus, maybe we can apply a similar method to VADER and change probability into sentiment score. In this way we may make the sentiment mark of sentences related to the order of words, but not simply a sum of scores.

1.3 Project Scope

The problem we are planning to address is to make the result changing with the order of words. According to the structure of vaderSentiment, the score calculating part is in SentimentIntensityAnalyzer. We shall mostly change the functions in this analyzer. However we will not change all functions in it. What we want to change are only the functions that doing the calculation of final sentiment score. We will try to change the score of each word by taking the previous words in count, and then calculating the result of sentences. For those used to identify special words or punctuation and calculate weights, we would keep them unchanged and use their outcome for the final calculation.

Apart from the coding part. We also need to build a new dataset with manual marks on sentences. As the demo datasets we used for testing VADER were gatherings of unrelated comments, we would like to chose some paragraphs that have emotional or logical relation between sentences. Also, as VADER lexicon only has score of words and punctuations, we need to manually mark on each sentence, and compare them with the processed result to evaluate the performance.

1.4 Report Outline

There are six chapters in this report:

- **Chapter 1:** Introduce the problem that is going to be solved in this thesis and the related motivation.
- **Chapter 2:** Describe the background of Automatic text classification and the related work on lexicon, lexicon-based classifier and n-gram.
- **Chapter 3:** Present the design and implementation of both new dataset and project structure.
- **Chapter 4:** Introduce of datasets used in this project, experimental equipment and platform.
- **Chapter 5:** Present the results and introduce the methods used for comparing them. Also, analyze the possible reason for such performances.
- **Chapter 6:** Conclude the project and give suggestions for the future work.

Background and Related Work

Inspired by the motivation that making the final sentiment score of latter word related to those of previous words, the literature research part started.

NLP research has a long history due to the large processing requirement of written text. We surveyed the work related to the proposal here to improve VADER, namely lexicon research, lexicon-based classification and n-gram text processing.

This chapter shall includes three sections. In the first section, a brief introduction of the background of this project will be delivered. Then following section shall introduce some useful related works gathered in the literature review process. The final section shall summarize this chapter briefly.

2.1 Background

Learning based approaches for automated text classification tend to be based on complex black-box algorithms that deliver high accuracy predictions but at the cost of not being able to explain the rationale behind their decisions.

Automatic text classification is a guided learning process, which divides a large number of unstructured text information (text documents, web pages, etc.) into specified categories according to the given classification system and text information content [Dalal and Zaveri, 2011]. Apart from lexicon classifier, there are machine learning based classifiers.

Machine learning is a tool that allows the system to automatically learn programs from data. In the past ten years, the development speed of machine learning has accelerated significantly and has expanded to more fields. The most widely used and mature field involving machine learning is classification. Machine learning-based classifiers generally extract features from multiple input samples and characterize them to generate models to predict the categories of new inputs [Chen, 2018].

Lexicon based classifier can be represented as a dictionary like table in which the corresponding weights of each word are stored. Weights are used to scale, for example, how words affect sentences, or in other words, how terms in a sequence affect the trend.

N-gram is an algorithm based on statistical language model. Its basic idea is to operate the content of the text in sliding window with the size of N according to

the byte, forming a byte fragment sequence with the length of N . Each byte segment is called gram. The frequency of all grams is counted and filtered according to the preset threshold to form a key gram list, which is the vector feature space of the text. Each gram in the list is a feature vector dimension. The model is based on the assumption that the occurrence of the N^{th} word is only related to the first $N-1$ word, but not to any other word. The probability of the whole sentence is the product of the occurrence probability of each word. These probabilities can be obtained by counting the number of simultaneous occurrences of N words directly from the corpus [Cavnar et al., 1994].

In this project, we want to design an experiment to extend a lexicon-based classifier named VADER (Valence Aware Dictionary and sEntiment Reasoner) to investigate how using n -grams will affect its performance. A comparison with original classifier can also be carried out to find out how performance differs on different types of datasets.

2.2 Related work

2.2.1 Automatic Text Classification

Automatic text classification involves the use of learning techniques to automatically assign text documents to a set of predefined classes. Classification is usually based on meaningful words or features extracted from text documents [Dalal and Zaveri, 2011]. Since the class is predefined, it is a supervised learning task. Many official communications and documents kept by commercial and government organizations are electronic files and e-mails in text form. Many personal and other communications between individuals are carried out in the form of e-mails, blogs, etc. Due to this information overload, effective classification and retrieval of related content becomes very important.

2.2.2 Sentiment Analysis

Sentiment analysis(SA) can be considered as a classification process. There are three main classification levels in SA: document level, sentence level and aspect level [Medhat et al., 2014]. Document level SA aims to classify opinion documents into expressing positive or negative opinions or emotions. It treats the entire document as a basic unit of information (talking about a topic). Sentence level SA aims to classify the emotions expressed in each sentence. The first step is to determine whether the sentence is subjective or objective. If the sentence is subjective, sentence level SA will determine whether the sentence expresses a positive or negative opinion. There is no fundamental difference between document and sentence level classification because sentences are only short documents. Classifying text at the document level or sentence level does not provide the necessary detailed opinions on all aspects of the entities needed in many applications to obtain the detailed information, so we need to get to the aspect level.

2.2.3 Text Classification

Text classification is a classic problem in natural language processing. Its purpose is to assign a label of a known category to a text unit, such as a sentence, paragraph, document, etc [Minaee et al., 2021]. On the Internet, text classification can provide more convenience to people's lives and derive subtasks such as gesture detection and sentiment analysis. [Chen, 2018] At present, many machine learning and deep learning technologies have been applied to text classification and have achieved good results [Kim, 2014]. For these models, complex structural design has produced good results, but it also makes the models lack of interpretability [Ribeiro et al., 2016].

2.2.4 Lexicon

Lexicon is part of NLP system that contains semantic and grammatical information about individual words or word strings, and can provide a structure which not only change the dictionaries into machine readable ones, but also can pick out the right meaning according to the text. The traditional dictionary explain words mostly using a genus term and then exclude other meanings, and may also used Longman Dictionary of Contemporary English to realize how to point out the right meaning, like going through the list and compare the meaning with the rest of the text, and pick the one with the highest overlap rate [Guthrie et al., 1996].

The process of text sentiment computing can be divided into three parts: text information collection, emotion feature extraction and emotion information classification. Specifically, we need to find the features that can be extracted by computer, and use the model that can be used for emotion classification.

There are three different techniques commonly used to build a lexicon. First of all, the traditional manual lexicon, which can be constructed by human hand, all the words contained are marked by different people according to specific sentiment standard (For example, the lexicon for VADER requires every word is given its mark in a range of -4 and 4). Second, the ontology based lexicon, which is generated according to the external relations of known seed words and their related synonyms, antonyms and supersems, spreads objects on the graph. Thirdly, corpus based lexicon, mainly involves conditional probability and pointwise interoperability [Bandhakavi et al., 2017]. However, this technique is more likely to overemphasize the relationship between terms and classes. At a simple statistical level, if a word occasionally appears too much in a class, although the word usually has no trend of any class, it may mislead its weight in the vocabulary and make it tend to that class [Chen, 2018].

2.2.5 Lexicon-Based Classifier

Lexicon based classifier is an attempt of interpretability in text classification task. It studies the trend of text to a certain category by scoring each word in a document or sentence and simply summarizing all the scores belonging to the same text. In this way, we can understand the contribution of a certain word in the text information to the category. For the traditional classifier based on lexicon, each word is scored

by professionals [Stone et al., 1966]. The advantage of this method is that it can provide high interpretability due to handmade. The limitation is that a large number of manual annotations make it inefficient.

The construction of the classifier is based on the algorithm that supports the classifier. Different types of classifiers may have their own characteristics, they have their own target regions and data sets. Therefore, it is necessary to test the algorithm to determine whether the selected classifier is suitable for the data to be processed [Chen, 2018].

The lexicon based classifier can be represented as a dictionary like table in which the weight of each word is stored. Weights are used to measure, for example, how a word affects a sentence, or in other words, how a word in a sequence affects the trend of the category to which the entity belongs [Chen, 2018].

Lexicon based dictionary can be described as an open source dictionary, people can view detailed information at any time. In addition, because its foundation is built manually, it can be easily modified and adjusted when needed. On the other hand, it is a difficult work to build a complete vocabulary, because there are more than 15000 English words in use, and some words may have different meanings, which may make this work even harder [Clos et al., 2017]. Although a dictionary has been successfully constructed, it takes time and effort to test and evaluate it on different types of data, and its accuracy cannot be guaranteed at the same time.

Lexicon-based methods only rely on language knowledge, and cross-domain and cross-text methods are more robust. However, high precision is difficult to achieve. Basically, they use an emotional vocabulary consisting of a pair of words and their polarities. The words belonging to the emotional vocabulary are called emotional words. It should be noted that not every word has a polarity value (so it belongs to a dictionary); usually adjectives, adverbs, and some entity words and verbs have polarity values. In addition, some rules for dealing with negation and strength are used to complete the simpler method [Avanço and Nunes, 2014].

2.2.6 N-gram

N-grams are sequences of characters or words extracted from text. N-gram can be divided into two categories: character-based and word-based.

Character n-gram is a set of N consecutive characters extracted from a word. The main motivation behind this approach is that similar words will have a high proportion of common n-grams. The typical value of N is 2 or 3; they correspond to the use of bigram or trigram, respectively. Character based n-gram is usually used to measure the similarity of strings. Spelling, stemming and OCR are some applications that use character based n-grams. The word n-grams is a sequence of N consecutive words extracted from the text. Word level n-gram model has strong robustness for language statistical modeling and information retrieval, and it does not depend on language very much [Majumder et al., 2002].

2.2.7 Language Model

The language model can be regarded as the probability distribution of word sequences or n-grams. Specifically, the language model estimates the probability of the next word given the preceding word. The word n-gram language model uses the history of the immediately preceding n-1 words to calculate the occurrence probability p of the current word. The value of n is usually limited to 2 (binary model) or 3 (ternary model). If the vocabulary size is M words, then in order to provide a complete coverage of all possible n word sequences, the language model needs to be composed of $M^n - \text{grams}$ [Niesler and Woodland, 1996]. Generally, n-gram language model only lists the most frequent pairs of words, and uses backoff mechanism to calculate the probability of not finding the required pairs.

2.3 Summary

In conclusion, the lexicon is a part of the natural language process, and is a list of text with corpus. The most common lexicons are manual lexicon, ontology based lexicon and corpus based lexicon.

The classifier based on lexicon can be represented as a dictionary like table, in which the weight of each word is stored. Weights are used to measure, for example, how a word affects a sentence, or in other words, how a word in a sequence affects the trend of the category to which the entity belongs.

After literature review, draft designing and implementation of both dataset and codes shall be carried out and introduced in the next chapter.

Design and Implementation

In this chapter we will introduce the design for both dataset and code. As VADER's lexicon only has sentiment score for words, I am planning to build a new dataset and invite my friends to mark on it. Then compare the output with manually marks to see its performance. I will also modify original VADER code to score words in four different ways(simple bigram, trigram, idf and tf-idf), and conduct experiment to evaluate the performance.

3.1 Design

Here I will introduce the design for dataset and for coding separately.

3.1.1 Dataset

For the dataset I used, I included a classic child story *The Cat in the Hat*, a piece of news from *New York Times* and a paragraph of sci-fi *The Three-Body Problem*. I chose these three articles/paragraphs as there are logical or sentiment relation between sentences. Thus, the former sentence may influence the sentiment of latter one. Unlike the given example dataset of VADER, which includes collections of comments that have no relationship between sentences, dataset like this may better reflect for improvement of my design. Besides, I invited three of my friends, Shixuan Liu, Peng Zhang and Yuchi Liu to manually mark on the dataset, and then calculating the average, which will be used to compare with the output to test performance.

3.1.2 Coding

After getting the input, the original VADER split the text into set of words and pick out punctuation, negative and booster words. They will be calculated as weight for the following word.

Thus, we cannot get the complete text after processing. Because of it, we need to save the complete text in another list for the score calculating, and put the sentiment calculating process ahead. Also, as these special words and punctuation has no score in the given lexicon, I just set them as 1 and observe the performance.

I will keep the additional emotional intensity generating part but mainly changing the simple summing score part. My first design is dividing the sentiment score of the former word by that of the latter score, and replace it as the final sentiment score of the word. This is motivated by the thought of n-gram, and can be regarded as a simple test to see whether the thought of making the previous word in count can work.

Similarly, I may make the processed score related to two previous words. That is, a simple method by using the score of $(n - 1)^{th}$ word dividing that of $(n - 2)^{th}$ word, and using the score of n^{th} word to divide the result and get the final processed score of the n^{th} one

Apart from that, I will also try IDF and TF-IDF:

3.1.3 IDF

IDF means Inverse Document Frequency. The IDF of a specific word can be obtained by dividing the total number of files by the number of files containing the word, and then taking the logarithm of the quotient [Ramos et al., 2003]. If there are fewer documents containing entry T and the IDF is larger, then the entry has a good ability to distinguish categories.

$$idf_i = \log \frac{|D|}{|j : t_i \in d_j|}$$

Figure 3.1: Formula of IDF

Where $|D|$ is the total number of documents in the corpus $|\{j : t_i \in d_j\}|$ denotes the number of files containing the word t_i (that is, the number of files with $n_i, n_j > 0$). If the word is not in the corpus, it will lead to zero denominator, so generally $1 + |\{j : t_i \in d_j\}|$ is used.

3.1.4 TF-IDF

TF means Term Frequency, which indicates how often an entry(keyword) appears in the text. This number is usually normalized(usually the word frequency divided by the total number of words in the article) to prevent it from being biased towards long files [Ramos et al., 2003].

$$tf_{ij} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

Figure 3.2: Formula of TF

Where $n_{i,j}$ are the times that the word appears in the file d_j , and the denominator is the sum of the times that all the words appear in the file d_j .

TF-IDF is actually $TF * IDF$. The high frequency of words in a specific file and the low frequency of words in the whole file set can produce high weight TF-IDF. Therefore, TF-IDF tends to filter out common words and retain important words [Ramos et al., 2003].

Here I will try both idf and tf-idf as this will change the score value according to the frequency and category differentiation ability. The result may not be better than running the original VADER, but there shall be interesting differences due to taking the whole text in count.

3.2 Summary

Here I need to design both the executable codes and dataset. For the dataset part, I combined a short child story, a piece of news and a paragraph from a science fiction together. This is to create logical or sentiment relation between sentences, thus the order between them cannot be changed. After that I will invite my friends to mark on each sentences manually, these scores will be compared with output to analyze improvement. For the coding session I will try simple bi-gram and tri-gram method, as well as IDF and TF-IDF. As original VADER will pick out negative, booster words and punctuation, I need to save them first in another list, and calculate the new score. The function of extracting special words and calculating weights will be kept as the same and their results will be used in the calculation of the final score.

Experimental Methodology

4.1 Software platform

The datasets for testing are in .docx form, and the related manual mark database is in .csv form. The output will be also written in .csv files.

Other information is show as below:

Table 4.1: Software Information

Language	Python3	
Compiler	3.7.7	
System	Windows 10.0.19042.985	
Application	Package	Version
	numpy	1.19.5
	docx	0.2.4
	pandas	1.2.3
	scikit-learn	0.24.2
	matplotlib	3.4.2
Output	Name	Type
	News	.csv
	Story	.csv
	sci-fi	.csv

4.2 Hardware platform

The hardware machine I used is mostly assigned as below:

Table 4.2: Hardware Information

Architecture	CPU	GPU
Model	Intel(R) i7-9700 CPU	NVIDIA GeForce GT 730
Clock	3.0GHz	1.66GHz
Piece(s)	4 cores \times 1	1
Memory	-	2.0 GB

4.3 Data Used

Here I actually have two copies of datasets, one for testing(without manually scoring) and one for evaluating(with manually scoring).

Here the datasets we built includes a classic story *The Cat in the Hat* [Seuss, 2021], a piece of news from *New York Times* [Mueller et al., 2021] and a paragraph from *Death's End* [Liu, 2021], which is the third book of the science fiction series *The Three Body Problem*. We selected these three paragraph because they have stronger inner emotional relation compared with the provided demo dataset.

The second version of datasets includes sentiment marks for each sentence. I invited three friends to give emotional mark on each sentence in the dataset. The mark shall be in the range from -1 to 1 where negative number means a negative overall sentiment and positive represents for overall positive sentiment. The average of four markings will be the final score of this sentence. This dataset shall be used for comparing outputs and evaluate their performance.

The sun did not shine. It was too wet to play. So we sat in the house All that cold, cold, wet day. [-0.1, 0.2, 0.0, 0.2], 0.08]←

I sat there with Sally. We sat there, we two. And I said, "How I wish We had something to do!" Too wet to go out And too cold to play ball. So we sat in the house. We did nothing at all. {[-0.1, 0.2, 0.1, 0.2], 0.1}←

So all we could do was to Sit! Sit! Sit! Sit! And we did not like it. Not one little bit. {[-0.1, 0.0, -0.2, -0.1], -0.1}←

(a) Story dataset

The sun did not shine. It was too wet to play. So we sat in the house All that cold, cold, wet day. [-0.1, 0.2, 0.0, 0.2], 0.08]←

I sat there with Sally. We sat there, we two. And I said, "How I wish We had something to do!" Too wet to go out And too cold to play ball. So we sat in the house. We did nothing at all. {[-0.1, 0.2, 0.1, 0.2], 0.1}←

So all we could do was to Sit! Sit! Sit! Sit! And we did not like it. Not one little bit. {[-0.1, 0.0, -0.2, -0.1], -0.1}←

(b) News dataset

Five days earlier, standing in front of the palace, Helena had demanded to see the emperor. {[0.0, 0.0, -0.1, 0.0], -0.03}←

When guards tried to push her away, she presented a small package that stunned the guards. {[0.0, -0.1, -0.1, 0.0], 0.05}←

They weren't sure what she was showing them, but they knew it was not something she should have possessed. {[0.1, 0.1, 0.1, 0.0], 0.08}←

Instead of being brought to the emperor, she had been held and interrogated about how she had acquired the item. Her confession had been confirmed, and she was then brought to Phrantzes. {[-0.1, -0.1, 0.0, 0.0], -0.05}←

(c) Sci-fi dataset

Figure 4.1: Example of datasets(with markings)

4.4 vaderSentiment

Here the classifier I used and improved is the python file vaderSentiment. The overall code is mainly divided into two classes: SentiText (object) and SentimentintensityAnalyzer (object). And four static methods: negated(), normalize(), allcap_differential() and scalar_inc_dec().

- SentiText(object):
 - _words_plus_punc() stitches the defined punctuation marks before and after the punctuation removed token to form a new token set

- `_ words_and_emoticons()` traverses the token of the text, removes the punctuation before and after the token that conforms to the token set style, and retains the abbreviations and most expressions
- `SentimentIntensityAnalyzer(object)`:
 - Returns a floating-point value of emotional strength based on the input text. Positive values are positive and negative values are negative
 - Use certain rules to calculate the score of words in sentences
- The emotional intensity of the whole sentence is calculated according to the score of each word

Here I made a picture trying to show the structure of VADER. I draw it based on the article *Code analysis of vaderSentiment* [Bao, 2021]:

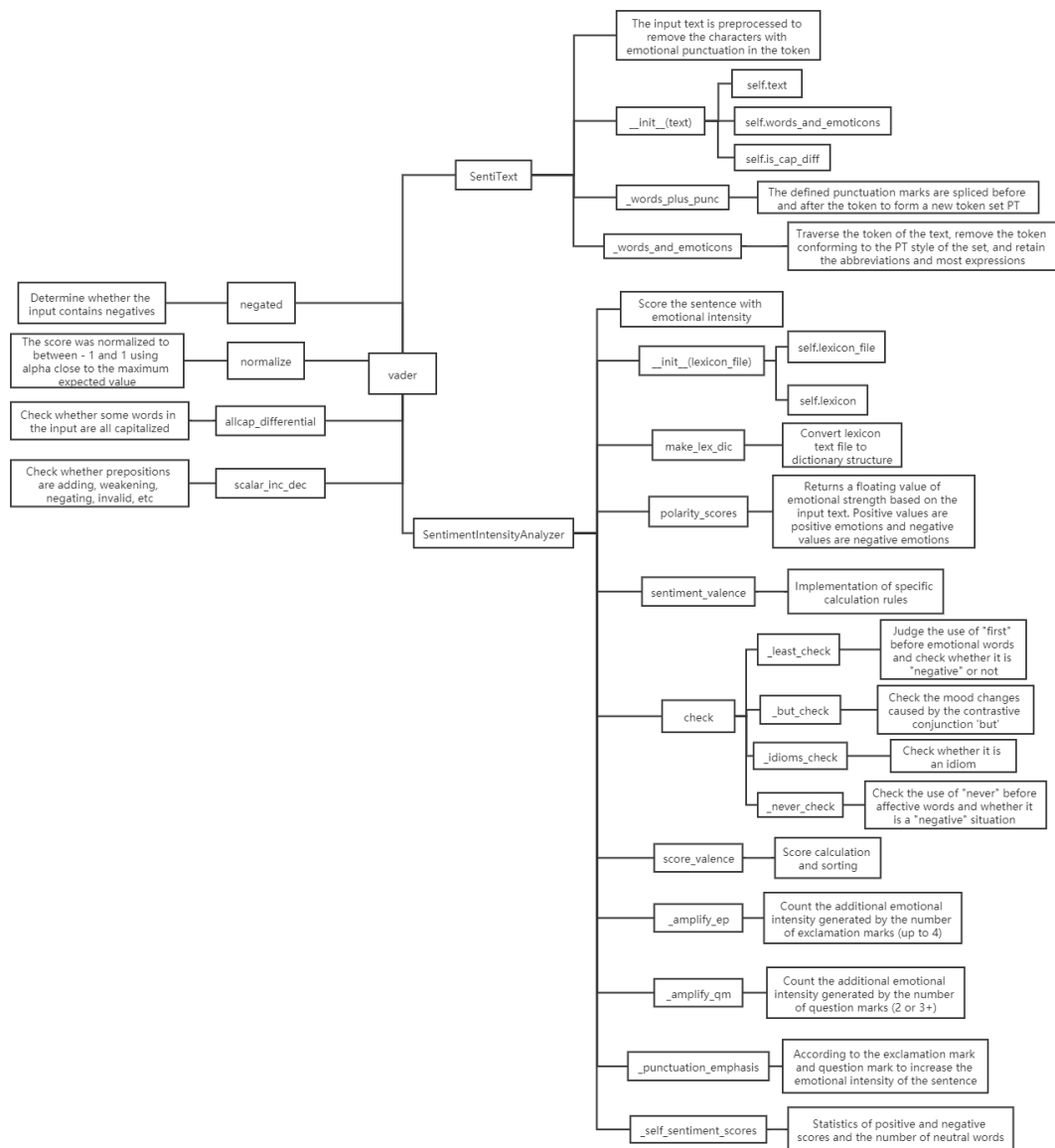


Figure 4.2: My figure of structure of vaderSentiment

Results

5.1 Measure Methods

One of the main problem for the result comparing is that the sentiment result of original VADER is divided into three scores: positive, neutral and negative, we need to get a final score for sentences based on the three outputs. Here I simply used the positive score minus the negative score, and if the result is positive then final sentiment is positive, if it is negative then the sentiment is negative. If the result is in the range from -0.01 to 0.01, then the final emotion is neutral(sentiment score is 0).

In the figure above, the black vectors represent the three sentiment value processed by original VADER, and the blue vector represent the final score that will be used for comparison.

After getting the result of the each sentences, I calculated the absolute value of result scores subtracting the manual marks, then calculated their mean and summary. I also calculated Minkowski Distance between the computed marks and manual scores, trying to figure out which method performs better.

5.1.1 Minkowski Distance

Minkowski distance is a measure in normed vector space. It is not simply a kind of distance, but more like a group of definitions of distance, which is a general expression of multiple distance measurement formulas [Singh et al., 2013].

$$Dist_{XY} = \left(\sum_{k=1}^d |X_{ik} - X_{jk}|^{\frac{1}{p}} \right)^p$$

Figure 5.1: Formula of minkowski distance

X and Y are two objects or sets of a certain class, and the value of the function indicates the degree of "similarity" between them. When $p = 1$, Minkowski distance is Manhattan distance, also known as city block distance. When $p = 2$, Minkowski distance is Euclidean distance. When $p \rightarrow \infty$, the Minkowski distance is Chebyshev distance.

Manhattan Distance is the distance between two points measured at right angles along an axis, it calculates the absolute difference between the coordinates of a pair of objects [Singh et al., 2013].

$$Dist_{XY} = |X_{ik} - X_{jk}|$$

Figure 5.2: Formula of manhattan distance

Euclidean distance between two points in Euclidean space is the length of line segment between two points, it calculate the absolute difference between a pair of object coordinates [Singh et al., 2013].

$$Dist_{XY} = \sqrt{\sum_{k=1}^m (X_{ik} - X_{jk})^2}$$

Figure 5.3: Formula of euclidean distance

Chebyshev distance is a measure defined in vector space, in which the distance between two vectors is the maximum difference in any coordinate dimension. It is also called the maximum distance, which is calculated as the absolute size of the coordinate difference between a pair of objects [Singh et al., 2013].

$$Dist_{XY} = \max_k |X_{ik} - X_{jk}|$$

Figure 5.4: Formula of chebyshev distance

5.2 Output

By the given formulas we can try to test the performance of different methods.

Table 5.1: The performance of different methods on the story dataset

Method	Mean	Manhattan	Euclidean	Chebyshev
Original VADER	0.029625	1.185	0.319128501	0.177
Bi-gram	0.03835	1.534	0.391594178	0.227
Tri-gram	0.0371	1.484	0.386217555	0.227
IDF	0.068225	2.729	0.627863839	0.251
TF-IDF	0.0335	1.34	0.317852167	0.196

Table 5.2: The performance of different methods on the news dataset

Method	Mean	Manhattan	Euclidean	Chebyshev
Original VADER	0.056364	1.24	0.512181608	0.404
Bi-gram	0.054273	1.194	0.445030336	0.312
Tri-gram	0.055773	1.227	0.446252171	0.312
IDF	0.091273	2.008	0.817575685	0.609
TF-IDF	0.045	0.991	0.365079443	0.231

Table 5.3: The performance of different methods on the sci-fi dataset

Method	Mean	Manhattan	Euclidean	Chebyshev
Original VADER	0.071643	1.003	0.333791851	0.148
Bi-gram	0.046143	0.646	0.262701351	0.168
Tri-gram	0.046143	0.646	0.262701351	0.168
IDF	0.155429	2.176	0.716342097	0.347
TF-IDF	0.036214	0.507	0.189765645	0.113

Table 5.4: The performance of different methods on the total dataset

Method	Mean	Manhattan	Euclidean	Chebyshev
Original VADER	0.045105	3.428	0.689630336	0.404
Bi-gram	0.044395	3.374	0.648390315	0.312
Tri-gram	0.044171	3.357	0.646000774	0.312
IDF	0.090961	6.913	1.255304346	0.609
TF-IDF	0.037342	2.838	0.519926918	0.231

From the output we can see that, the original TF-IDF is mostly the best choice as it has the smallest distance, which means that it has the closest marks compared to the manual ones. The outputs for bi-gram and tri-gram are similar and mostly better than that of original VADER. That for IDF is much worse as its distances are almost twice of original VADER's.

The reason for such output may be because, firstly, the bi-gram and tri-gram is too simple and can not exactly show the relation of between such words. Actually, the this is more like a simple test of whether my system can work and what would happen if we make previous words in count.

For the IDF and TF-IDF parts, the the frequency may influence the sentiment, but the influence may not be as important as I thought. Especially Inverse Document Frequency shall make the performance even worse. Thus, the frequency related methods may not be perform as well as expected.

5.3 Summary

In conclusion, I made subtraction operation between positive score and negative score to get the final result and calculated the absolute value of it minus the manually marked score. Then, based on the result, calculated the mean and three different

distances. The smaller the distance is, the closer it is compared with the manual mark, which also means that it has better performance.

According to the output, the TF-IDF is the best one, followed by Tri-gram, Bigram, original VADER and IDF. The reason for this is, firstly the brutal dividing method can not really show the sentiment relation between words. Besides, frequency may not influence the output as well as we expected, and may even be counterproductive.

Conclusion

As VADER is a lexicon and rule-based sentiment analysis tool which performs well on the individual sentences and single words. However, as it roughly calculated the scores by summarizing the lexicon scores of each word, I tried to change the order of words in sentences as well as order of sentences in paragraph.

So I thought about n-gram, which takes the probability of the previous words in count to avoid the situation that the the results are the same after changing the order of words. Thus, I was considering to apply a similar method to make the final output related to previous words.

Here I built a new dataset to with paragraphs that have sentences related to each other. I also invited my friends to manually to mark on each sentences. These sentiment scores will be used to measure the performance of the changings.

I used the absolute value of output score minus the dataset score, and then calculated manhattan distance, euclidean distance, chebyshev distance based on it. The smaller the distance is, the better the performance is. As for the methods, I used a simple method of using the score dividing the score of the former one, and another try is to use the score of $(n - 1)^{th}$ word to divide that of $(n - 2)^{th}$ word, and using the score of n^{th} word to divide the result and get the final sentiment score of the n^{th} word. IDF and TF-IDF methods were also applied.

According to the result, TF-IDF is the best choice. For the other methods, the ti-gram related one is the best. However, all of their performances are not significantly better than that of original VADER. This may be because brutally dividing the sentiment scores can not really show the sentiment relation between words, and frequency may even be counterproductive.

6.1 Future Work

There are still two main problems in this topic. Firstly, a reasonable method to review the sentiment relation between words are still not be found. Here I would recommend making a phrases lexicon and check the phrases one first when marking. If there are not recorded score for phrases then check the word lexicon(offered by VADER) for each word.

Another problem is that the lexicon does not provide sentiment score for the

special words and symbols like the negative words or exclamation marks. Here I simply set the score as 1, and a reasonable score should be worked out for them.

Appendix

7.1 Negate

The vocabulary of negative words used in this project was made by C.J. Hutto [Hutto, 2021].

Table 7.1: Vocabulary for negative words

Negate Vocabulary					
aint	arent	cannot	cant	couldnt	darent
didnt	doesnt	ain't	aren't	can't	couldn't
daren't	didn't	doesn't	dont"	hadnt"	hasnt
havent	isnt	mightnt	mustnt	neither	don't
hadn't	hasn't	haven't	isn't	mightn't	mustn't
neednt	needn't	never	none	nope	nor
not	nothing	nowhere	oughtnt	shant	shouldnt
uhuh	wasnt	werent	oughtn't	shan't	shouldn't
uh-uh	wasn't	weren't	without	wont	wouldnt
won't	wouldn't	rarely	seldom	despite	*

7.2 Booster

The dictionary of booster words used in this project was made by C.J. Hutto [Hutto, 2021].

Table 7.2: Vocabulary for booster words

Booster Dictionary					
absolutely	amazingly	awfully	completely	considerable	considerably
decidedly	deeply	effing	enormous	enormously	entirely
especially	exceptional	exceptionally	extreme	extremely	fabulously
flipping	flippin	frackin	fracking	fricking	frickin
frigging	friggin	fully	fuckin	fucking	fuggin
fugging	greatly	hella	highly	hugely	incredible
incredibly	intensely	major	majorly	more	most
particularly	purely	quite	really	remarkably	so
thoroughly	total	totally	tremendous	tremendously	uber
unbelievably	unusually	utter	utterly	very	almost
barely	hardly	just enough	kind of	kinda	kindof
kind-of	less	little	marginal	marginally	occasional
occasionally	partly	scarce	scarcely	slight	slightly
somewhat	sort of	sorta	sortof	sort-of	*

7.3 Sentiment Laden Idioms

The list of sentiment laden idioms that do not contain lexicon words was built by C. Hutto and E. Gilbert [Hutto and Gilbert, 2014].

Table 7.3: List for sentiment laden idioms

Sentiment Laden Idioms					
cut the mustard	hand to mouth	back handed	blow smoke	blowing smoke	upper hand
break a leg	cooking with gas	in the black	in the red	on the ball	under the weather

7.4 Special Cases

The list of special case idioms and phrases containing lexicon words was built by C. Hutto and E. Gilbert [Hutto and Gilbert, 2014].

Table 7.4: List for special case idioms and phrases

Special Cases					
the shit	the bomb	bad ass	badass	bus stop	yeah right
kiss of death	to die for	beating heart	broken heart	*	*

7.5 TF-IDF Implementation

The function below shows my implementation of tf-idf:

```

1 def TF_IDF(path):
2     document = docx.Document(path)
3     sentences = []
4     for para in document.paragraphs:
5         sentences.append(para.text)
6
7     doc_frequency = defaultdict(int)
8     list = []
9     for sentence in sentences:
10         list.append(re.split(r'[.,;!?"\n\'] ', sentence))
11
12     words = []
13     for i in list:
14         for j in i:
15             if j != '':
16                 words.append(j.lower())
17
18     for word in words:
19         doc_frequency[word.lower()] += 1
20
21     word_tf = {}
22     for i in doc_frequency:
23         word_tf[i] = doc_frequency[i] / sum(doc_frequency.values())
24
25     doc_num = len(list)
26     word_idf = {}
27     word_doc = defaultdict(int)
28     for i in doc_frequency:
29         for j in list:
30             has = 0
31             for word in j:
32                 if word.lower() == i:
33                     has = 1
34                     break
35             if has:
36                 word_doc[i] += 1
37
38     for i in doc_frequency:
39         word_idf[i] = math.log(doc_num / (word_doc[i] + 1))
40
41     word_tf_idf = {}
42     for i in doc_frequency:
43         word_tf_idf[i] = word_tf[i] * word_idf[i]
44
45     return word_idf, word_tf_idf

```

7.6 Artifact

For further information, please check my Github repository:

<https://github.com/JLZong/Improving-lexicon-based-classification-using-n-grams>

Bibliography

- AVANÇO, L. V. AND NUNES, M. D. G. V., 2014. Lexicon-based sentiment analysis for reviews of products in brazilian portuguese. In *2014 Brazilian Conference on Intelligent Systems*, 277–281. doi:10.1109/BRACIS.2014.57. (cited on page 8)
- BANDHAKAVI, A.; WIRATUNGA, N.; MASSIE, S.; AND PADMANABHAN, D., 2017. Lexicon generation for emotion detection from text. *IEEE Intelligent Systems*, 32, 1 (2017), 102–108. doi:10.1109/MIS.2017.22. (cited on page 7)
- BAO, X., 2021. Code analysis of vadersentiment. <https://www.jianshu.com/p/0fa71dc592b0>. (cited on page 18)
- CAVNAR, W. B.; TRENKLE, J. M.; ET AL., 1994. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval*, vol. 161175. Citeseer. (cited on page 6)
- CHEN, Q., 2018. Text classification: Producing explainable and performant solutions using hybrid lexicon-based approaches. <https://github.com/sidongfeng/csproj18s1>. (cited on pages 5, 7, and 8)
- CLOS, J.; WIRATUNGA, N.; AND MASSIE, S., 2017. Towards explainable text classification by jointly learning lexicon and modifier terms. (cited on page 8)
- DALAL, M. K. AND ZAVERI, M. A., 2011. Automatic text classification: a technical review. *International Journal of Computer Applications*, 28, 2 (2011), 37–40. (cited on pages 5 and 6)
- GUTHRIE, L.; PUSTEJOVSKY, J.; WILKS, Y.; AND SLATOR, B. M., 1996. The role of lexicons in natural language processing. *Commun. ACM*, 39, 1 (Jan. 1996), 63–72. doi: 10.1145/234173.234204. <https://doi.org/10.1145/234173.234204>. (cited on page 7)
- HUTTO, C., 2021. Vocabulary for modifiers and negative words. <https://github.com/cjhutto/vaderSentiment/blob/master/vaderSentiment/vaderSentiment.py>. (cited on page 29)
- HUTTO, C. AND GILBERT, E., 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 8. (cited on pages 1 and 30)
- KIM, Y., 2014. Convolutionalneuralnetworksforsentence classification. <http://arxiv.org/abs/1408.5882>. (cited on page 7)

-
- LIU, C., 2021. Death's end. <https://us.macmillan.com/books/9780765386632>. (cited on page 16)
- MAJUMDER, P.; MITRA, M.; AND CHAUDHURI, B. B., 2002. N-gram: a language independent approach to ir and nlp. (2002). (cited on page 8)
- MEDHAT, W.; HASSAN, A.; AND KORASHY, H., 2014. Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, 5, 4 (2014), 1093–1113. (cited on page 6)
- MINAEE, S.; KALCHBRENNER, N.; CAMBRIA, E.; NIKZAD, N.; CHENAGHLU, M.; AND GAO, J., 2021. Deep learning-based text classification: A comprehensive review. *ACM Computing Surveys (CSUR)*, 54, 3 (2021), 1–40. (cited on page 7)
- MUELLER, B.; PRONCZUK, M.; AND STEVIS-GRIDNEFF, M., 2021. The european union sues astrazeneca over missing vaccine doses. <https://www.nytimes.com/2021/04/26/world/europe/astrazeneca-eu-lawsuit.html?searchResultPosition=18>. (cited on page 16)
- NIESLER, T. R. AND WOODLAND, P. C., 1996. A variable-length category-based n-gram language model. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, vol. 1, 164–167. IEEE. (cited on page 9)
- RAMOS, J. ET AL., 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, vol. 242, 29–48. Citeseer. (cited on pages 12 and 13)
- RIBEIRO, M. T.; SINGH, S.; AND GUESTRIN, C., 2016. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 1135–1144. (cited on page 7)
- SEUSS, T., 2021. Words to the poem the cat in the hat dr seuss. <https://www.oatridge.co.uk/poems/d/dr-seuss-cat-in-the-hat.php>. (cited on page 16)
- SINGH, A.; YADAV, A.; AND RANA, A., 2013. K-means with three different distance metrics. *International Journal of Computer Applications*, 67, 10 (2013). (cited on pages 21 and 22)
- STONE, P. J.; DUNPHY, D. C.; AND SMITH, M. S., 1966. The general inquirer: A computer approach to content analysis. (1966). (cited on page 8)

INDEPENDENT STUDY CONTRACT

Note: Enrolment is subject to approval by the Honours/projects co-ordinator

SECTION A (Students and Supervisors)

UniID: u6921252

FAMILY NAME: Zong PERSONAL NAME(S): Jiulin

PROJECT SUPERVISOR (*may be external/HDR*): Kerry Taylor

FORMAL SUPERVISOR (*a RSCS academic*): Priscilla Kan John

COURSE CODE, TITLE AND UNIT: COMP4560 Advanced Computing Project 12 units

SEMESTER(S) S2 YEAR: 2020 ☒ S1 YEAR: 2021

PROJECT TITLE: Explainable Text Classification: Improving lexicon-based classification using n-grams

LEARNING OBJECTIVES:

In this project, the student will:

- *Conduct a literature review on chosen topic
- *Improve the performance of a lexicon-based classifier using n-grams (project classifier)
- *Set up experiments to compare the performance of project classifier using suitable datasets
- * (stretch goal) Benchmark project classifier with other classifiers of different families
- *Analyse data to draw meaningful conclusions
- *Communicate findings effectively in a report and oral presentation
- *Reflect on learning experience

PROJECT DESCRIPTION:

Learning based approaches for automated text classification tend to be based on complex black-box algorithms that deliver high accuracy predictions but at the cost of not being able to explain the rationale behind their decisions.

In this project, we want to design an experiment to extend a lexicon-based classifier named VADER (Valence Aware Dictionary and sEntiment Reasoner) to investigate how using n-grams will affect its performance. A comparison with learning-based classifiers can also be carried out to find out how performance differs on different types of datasets. By design, VADER has been built to work well on sentiment analysis on social media, it would be of interest to see how VADER (both with unigrams and n-grams) perform in other areas (transfer learning).

Reference

PyPI. 2021. *vaderSentiment*. [online] Available at: <<https://pypi.org/project/vaderSentiment/>> [Accessed 4 February 2021].

ASSESSMENT (as per course's project rules web page, with the differences noted below):

<input type="checkbox"/> Honours (24 credit)	(fixed)	<input checked="" type="checkbox"/> Projects (6-12 credit) / (fixed)
Assessed project components:	% of mark	Assessed project components:	% of mark
Thesis	(85%)	Thesis (reviewer mark)45....%
Presentation	(10%)	Artefact (supervisor project mark)45%
Critical Feedback	(5%)	Presentation	(10%)

MEETING DATES (IF KNOWN):

STUDENT DECLARATION: I agree to fulfil the above defined contract:

Jiulin ZONG
.....
Signature

05/02/2021
.....
Date

SECTION B (Supervisor):

I am willing to supervise and support this project. I have checked the student's academic record and believe this student can complete the project.


.....
Signature

23/02/2021
.....
Date

Reviewer:

Name: Pouya Ghiasnezhad Omran.....

Signature: P.G.


Reviewer 2: (for Honours only)

Name:

Signature:

REQUIRED DEPARTMENT RESOURCES:

SECTION C (Honours / Projects coordinator approval)


.....
Signature

25/02/2021
.....
Date