Sentiment Analysis on Formal and Informal Language

Lakshya
SRM AP University, Amaravati
lakshya_m@srmap.edu.in

Fadzai Muchina
SRM AP University, Amaravati
fadzai_muchina@srmap.edu.in

***Abstract*—In this project an efficient and accurate model is presented which can analyse and quantify the sentiment behind statements on large-scale datasets. Sentiment analysis is an ongoing field of research in text mining. Sentiment analysis is the computational treatment of opinions, beliefs and subjectivity of text. This project involves the study of several deep learning sentiment analysis models, their applications, their challenges, and the construction of a sentiment analysis model of one of the proposed approaches that best suit our needs (Classification of sentiments in Twitter and Metacritic data). In this project, I aim to use the Valence Aware Dictionary for sentiment Reasoner (VADER) to classify the sentiments expressed in Twitter and Metacritic data.**

Keywords—Machine Learning, NLP, Sentiment Analysis

1. Introduction

Natural language processing (NLP) is a field of artificial intelligence in which computers analyze, understand, and derive meaning from human language in a smart and useful way. By utilizing NLP, developers can organize and structure knowledge to perform tasks such as automatic summarization, translation, named entity recognition, relationship extraction, sentiment analysis, speech recognition, and topic segmentation. Sentiment analysis is an approach to natural language processing which identifies the emotional tone behind a body of text. In order to study sentiment analysis, we first have to ask ourselves some basic questions. What is sentiment analysis? Why do we need it? What are its applications? What are the different types of sentiment analysis? and how many ways are there to perform these tasks? We will uncover these topics in our project.

1.1 Objectives

The following are the objectives of this project: To perform Sentiment Analysis on brief text statements acquired from product evaluations and tweets using Natural Language Processing (NLP). Following the generation of overall sentiment analysis for the statements, the goal is to determine the subject in the statement. The project's subsequent study is to determine if the same reasoning can be applied in statements including informal material (i.e., language used on social media). For the formal and informal structures, we shall use Metacritic Game Reviews and Tweets respectively as they capture the essence of the structures we aim to test.

1.2 Background and Literature Survey

Sentiment analysis is a very well-documented field, so in order to understand and study the field I started with the basic question, what is sentiment analysis? “Sentiment analysis is the study of people’s sentiments, opinions, feedback, and attitude towards a certain product or service and their attributes. With the advent of the internet age large amounts of data are collected via our search

histories, tweets, and Instagram posts, the internet generates around 2.5 quintillion bytes of data every day. Which makes sentiment analysis one of the largest studied fields in deep learning and data mining". It can analyse everything from the direction in which people will vote in the next elections to which movie is bound to be the next blockbuster. Nowadays if you want to buy a new phone, instead of asking a family member or a friend for a recommendation you go to google reviews to search for products reviews and ratings, on an individual level one review means nothing to a company but collectively on a global scale these small reviews can impact a company in millions of dollars. But the reviews or tweets on the internet may contain emojis, incorrect grammar, or even sarcasm. This makes the task especially challenging to look through these layers of language and find out the essence of any message on the internet. Now the question arises what are different types of sentiment analysis? To answer this question I referred to a paper by Ronen Feldman named "Techniques and Applications for Sentiment Analysis". In this, he describes that sentiment analysis can be of three types "document-level sentiment analysis, sentence-level sentiment analysis and aspect-level sentiment analysis".

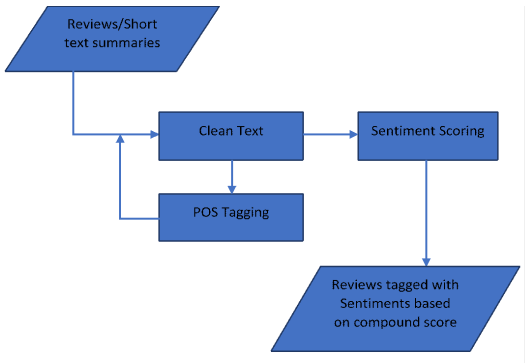
Document-level sentiment analysis is the case when an author expresses one opinion about the subject matter in a document. This task can be done using supervised machine learning models i.e. the number of classes in which the document is supposed to be classified is finite (positive, negative or neutral). We can also use a five-star rating system like amazon. Algorithms like SVM, Naive Bayes or KNN can be used for these tasks. This is one of the easiest analyses that can be done. We can also approach this topic through unsupervised learning in which we evaluate 13 the semantic orientation of specific phrases in the document and if the sum of their values crosses a certain threshold it is classified as positive or negative. Sentence-level sentiment analysis: A single document may contain multiple views about the same entity, for example, the chilly cheese fries were salty but also crisp and delicious. In the example, a small sentence contains two different views so a document will contain multiple views, in order to tackle that sentence-level sentiment analysis is introduced. Aspect-Base sentiment analysis the previous two approaches work when the document or the sentence talks about one attribute. But what happens when in a discussion forum about cars people talk about different cars, their particular engines, the horsepower or the comfort of the car? In such cases when the attributes are multiple aspect-based sentiment analysis works best. Upon further inspection of related literature, I found that most commonly there are three ways to perform the above-mentioned tasks, they are Naive Bayes, Deep Learning LSTM, and VADER Model. In order to find the most efficient way I studied these three models in detail. Naive Bayes approach: In this, the Naive Bayes classifier utilizes the probabilities based on prior knowledge of the conditions that might be related. It does that by constructing of a document term matrix for the model. This approach is easy to understand and fairly reasonable. The use of this classifier requires the preprocessing of data. For capturing the context of words N-grams are used, but Ngrams being blunt instruments can't always capture the essence correctly and they can burden the model with too many features. The data preprocessing includes the standard steps(Converting case, tokenising, lemmatize, stop word removals) with the addition of vectorizing the words(TF-IDF). Deep Learning LSTM approach: LSTM stands for Long-Short Term Memory, which is a type of RNN used to process temporal data (data relating to time-related instances). Deep learning is computationally expensive and does not do well with high dimensional sparse vectors. In this technique each word is converted into a sequence of numbers where each number is mapped to a

word in the vocabulary, Taking it a step further, 14 we need to map words that have similar usage/meaning to similar real number vectors (rather than an index) using Word Embeddings. The preprocessing follows the standard process mentioned in naive Bayes, and while vectorizing the text is converted into sequence and word embedding is used, this step comes under the vectorizing of words. The big advantage of this model as compared to naive Bayes is that we don't have to engineer features, since the model will automatically learn the features during training making the model highly accurate. VADER model: "VADER (Valence Aware Dictionary and sentiment Reasoner) tool. At its core, VADER uses a comprehensive, high-quality lexicon (~7500 features) and sophisticated linguistic rules to produce sentiment scores". Vader being a valance-based lexicon, is capable of detecting both the intensity and polarity of sentiments. This tool combined with modifiers such as negations, contractions, conjunctions etc. is used to calculate scores for sentences. The advantage of Vader over the other two models is the ability of the model to detect slang and sarcasm while being highly efficient and fast. The real challenge for creating tools like this is a large amount of time it takes to create tools like Vader. The preprocessing for Vader follows, Converting case, tokenize, lemmatize, stop word removals followed by POS tagging. Therefore picking the right tool will depend, on if speed is prioritized over accuracy Vader should be used, if accuracy is paramount Deep learning model should be used and if the training data is robust and accurate then the Naive Bayes model should be used. In our case, the data we are using is Twitter and Metacritic review data which isn't robust and contains slang and informal sentence structure. Hence we will be using Vader for our project. In each of the models' data preprocessing is the common step with some small changes, so data preprocessing in the context of sentiment analysis was studied to determine what challenges it might entail. A methodology was determined after the study of the Vader model and will be mentioned later in the report.

2. Methodology

2.1 Proposed System

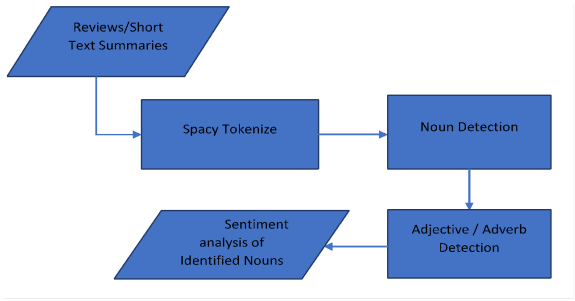
Using NLTK, we performed sentiment analysis on the Metacritic Video Game Reviews. Using NLTK, we tokenized the statements and down the words to their simpler form (also referred to as cleaning of the text). This helps in figuring out the Parts-of-speech (POS) tags for the words in each statement, which is used by the model to perform sentiment analysis. The model used is derived from the NLTK package in the Vader Module. The model is trained by the developers and any user must only clean and normalize the text before providing it as input to the model. We then performed sentiment analysis and classify them into three possible labels: 1. Positive 2. Neutral 3. Negative The following diagram (figure 1) shows the algorithms used for sentiment analysis.



(Figure 1: Workflow for Sentiment Analysis)

For identifying the aspects, i.e., the parts that the sentiments are expressed towards. We look at the grammatical structure of the statement using spaCy, and we can see that the aspects in all of the statements are identified as Nouns and the describing the nouns are Adjectives. spaCy can parse and tag any statement after tokenization using its trained pipeline and its statistical models to forecast which tag or label is most likely to

apply in this situation. Upon the identification of the aspects and their descriptive word(s), we perform sentiment analysis to see the sentiments associated with the descriptive words. The diagram (figure 2) shows the proposed workflow for aspect-based sentiment analysis



(Figure 2: Workflow for aspect-based sentiment analysis)

2.2 Working Methodology

This project has two parts: Sentiment analysis, in which we calculate the polarity of sentences. Aspect-based sentiment analysis, in which we are identifying the aspects, i.e., the parts that the sentiments are expressed towards. Each of these tasks will be carried out on formal sentence structure(Metacritic Game Reviews), and on informal sentence structures(Tweets).

2.3 System Details

This section describes the software details of the system

2.3.1 Software Details

First analysis of both datasets will be performed. This will provide us with a general overview of the dataset. We perform this using the pandas library in python. We start by importing the Twitter dataset and displaying the first four rows of the dataset

Now we run the describe function and isnull function in pandas to check for any null values or inconsistencies in the dataset. This will allow us to clean the data better during data preprocessing.
Now we follow the same steps for the Metacritic.csv file.

Now that we have a general idea of what is in our dataset we shall start with data preprocessing for sentiment analysis and aspect-based sentiment analysis. We start by importing libraries:

1. Pandas is used for dataset operations
2. String is used for identifying parts of grammar; in the program's case it is punctuation
3. pos_tag, Identifies the Parts-of-Speach tags of the word, like, "Canada" is a noun
4. stopwords is a list of words that don't add a significant meaning to the statement during processing
5. WordNetLemmatizer, Lemmatization is a process which converts words in multiple forms to an easy-to-process word, like, "runs" and "running" turns to "run"
6. SentimentIntensityAnalyzer: The Sentiment Analysis model which is trained and is readily available by the module developers
7. Bar: Adding a progress bar to keep track of the progress of the functions
8. spacy: A superior form of the NLTK package which offers better analytical tools for examining the relationships between the words

Now we define a function to clean the text that preprocesses the data by performing the following steps

1. converts the text into lowercase
2. tokenize the text and remove all punctuation
3. Remove words that contain a number
4. Remove stop words
5. Remove empty tokens
6. POS tag the text
7. Lemmatize the text
8. Remove words with only one letter

Now that we have the functions defined we can start with the sentiment and aspect-based sentiment analysis.
Sentiment Analysis:

1. First import the dataset
2. Then select the review column
3. Initialize the progress bar to keep track of the progress.

4. Clean the text using the above function for each review
5. All the clean text is passed through the sentiment intensity analyzer to generate polarity scores.
6. Use the generated scores to generate labels for each review.

Aspect-based sentiment analysis:

1. Define a function that finds sentiments, similar to the code segment above, but is only applied for words or small sets of words to understand its sentiments
2. Each review is analyzed to identify the aspects, the words that provide its sentiments and the sentiments towards the aspects
3. Use spacy to tokenize each sentence
4. For each of the tokens look at the words in the statement and if they are nouns that have links (i.e 'nsubj') then they are the aspects, and if the words 24 are adjectives and are not linked to any adverbs then they are the descriptive terms
5. Add this information in a dictionary format containing the aspects, description and sentiments
6. Find sentiment for each of these aspects and add it to a new column in the CSV file.
7. Save the CSV file under a new name with its sentiment analysis and aspect-based sentiment analysis.

Repeat this process for the Metacritic dataset.

3. RESULTS AND DISCUSSIONS

A. Metacritic Game Reviews

The program is able to identify the overall sentiments of the results with an F-1 accuracy score of ~82.921%. From the findings, we can see that NLTK provides a fast and reliable method to identify sentiments. It should also be reiterated that the limits of labelling the polarity scores along with the tagging of “True Values” for the F1 testing are set by the individual and its modification will lead to different accuracy scores. The aspect-based sentiment analysis is able to successfully identify certain aspects of the statement and also its sentiments. However, there are statements where the dependencies were not found but the sentiments were detected. Also, in some cases, the detected nouns didn’t make any sense. Another issue with the method proposed in the aspect-based sentiment analysis [3], is that the program can only detect one aspect, if a complex statement is provided with multiple nouns and is described with various adjectives then the program may not be able to detect the aspects properly. For future implementation, the recommendations are to

1. Modify the hyper-parameters to improve accuracy, precision, recall, etc.
2. Modify the code to detect multiple aspects
3. Adjust for the sarcastic tones

B. Twitter Tweets

The sentiments from the tweets are accurately captured. However, the dataset requires cleaning as it contains emoticons, which the program interprets as special characters and messes with the spaCy interpretation of the statement.

4. CONCLUSION AND FUTURE WORK

For future implementation, the recommendations are to

1. An automated data cleaning method to normalize all characters, especially emoticons
2. Conversion of abbreviations like WTH or TMI to a form that can be interpreted by the program

However, the above recommendations have the disadvantage of requiring a significant amount of time and resources to process the information, but the advantage is that a model will be used that is accurate and reliable. Another possible method is to create a new custom model which operates on a similar logic and produces similar results but is tweaked for the inclusion of informal structures. This also brings the advantage of not preprocessing the information to a great extent, but the disadvantage is reworking the entire model which might affect its accuracy and will also require knowledge and time in terms of development.

5. REFERENCES

1. Algorithmia. 11th August 2016. What is natural language processing? Introduction to NLP. [online] Available at: https://algorithmia.com/blog/introductionnatural-language-processing-nlp

2. GeeksForGeeks. Python | Sentiment Analysis using VADER. [online] Available At: https://www.geeksforgeeks.org/python-sentiment-analysis-using-vader/

3. Pattakos, A. (2021). Aspect-Based Sentiment Analysis Using Spacy & TextBlob. Towards Data Science. [online] Available at: https://towardsdatascience.com/aspectbased-sentiment-analysis-using-spacy-textblob-4c8de3e0d2b9

4. spaCy. Facts-Figures. [online] Available at: https://spacy.io/usage/facts-figures

5. spaCy. Linguistic Features. Available at:: https://spacy.io/usage/linguistic-features

6. Doaa Mohey El-Din Mohamed Hussein, A survey on sentiment analysis challenges, Journal of King Saud University - Engineering Sciences, Volume 30, Issue 4, 2018, Pages 330-338, ISSN 1018-3639, https://doi.org/10.1016/j.jksues.2016.04.002. (https://www.sciencedirect.com/science/article/pii/S1018363916300071)

7. Sentiment Analysis and Opinion Mining, Bing Liu, Synthesis Lectures on Human Language Technologies, May 2012, Vol. 5, No. 1 , Pages 1-167, (https://doi.org/10.2200/S00416ED1V01Y201204HLT016)

8. Emma Haddi, Xiaohui Liu, Yong Shi, The Role of Text Pre-processing in Sentiment Analysis, Procedia Computer Science, Volume 17, 2013, Pages 26-32, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2013.05.005. (https://www.sciencedirect.com/science/article/pii/S1877050913001385)

9. https://towardsdatascience.com/sentiment-analysis-comparing-3-commonapproaches-naive-bayes-lstm-and-vaderab561f834f89#:~:text=There%20are%20numerous%20approaches%20for,Trained%20Rule%2DBased%20VADER%20Models.

10. Zhang, Lei, Shuai Wang, and Bing Liu. "Deep learning for sentiment analysis: A survey." Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 8.4 (2018): e1253.

11. Zhang, L., Wang, S. and Liu, B., 2018. Deep learning for sentiment analysis: A survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8(4), p.e1253.

12. Zhang, Lei, Shuai Wang, and Bing Liu. "Deep learning for sentiment analysis: A survey." Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 8, no. 4 (2018): e1253.

13. Taboada, Maite & Brooke, Julian & Tofiloski, Milan & Voll, Kimberly & Stede, Manfred. (2011). Lexicon-Based Methods for Sentiment Analysis. Computational Linguistics. 37. 267-307. 10.1162/COLI_a_00049.

14. Agarwal, Apoorv, Boyi Xie, Ilia Vovsha, Owen Rambow and R. Passonneau. “Sentiment Analysis of Twitter Data.” (2011).

15. Schouten, Kim & Frasincar, Flavius. (2015). Survey on Aspect-Level Sentiment Analysis. IEEE Transactions on Knowledge and Data Engineering. 28. 1-1. 10.1109/TKDE.2015.2485209. 36

16. Hutto, C.J. & Gilbert, Eric. (2015). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Proceedings of the 8th International Conference on Weblogs and Social Media, ICWSM 2014.

17. Elbagir, Shihab & Yang, Jing. (2020). Sentiment Analysis on Twitter with Python’s Natural Language Toolkit and VADER Sentiment Analyzer. 63-80. 10.1142/9789811215094_0005.

18. Zhang, Lei & Wang, Shuai & Liu, Bing. (2018). Deep Learning for Sentiment Analysis : A Survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 8. 10.1002/widm.1253.

19. Kiritchenko, Svetlana & Zhu, Xiaodan & Mohammad, Saif. (2014). Sentiment Analysis of Short Informal Text. The Journal of Artificial Intelligence Research (JAIR). 50. 10.1613/jair.4272.

20. Taboada, Maite. (2016). Sentiment Analysis: An Overview from Linguistics. Annual Review of Linguistics. 2. 10.1146/annurev-linguistics-011415-040518.

21. Hassan Yousef, Ahmed & Medhat, Walaa & Mohamed, Hoda. (2014). Sentiment Analysis Algorithms and Applications: A Survey. Ain

Shams Engineering Journal. 5. 10.1016/j.asej.2014.04.011.
22. Feldman, Ronen. (2013). Techniques and Applications for Sentiment Analysis. Commun. ACM. 56. 82–89. 10.1145/2436256.2436274.
23. Mohey El-Din, Doaa. (2016). A Survey on Sentiment Analysis Challenges. Journal of King Saud University - Engineering Sciences. -. 10.1016/j.jksues.2016.04.002.

Appendix

```
import pandas as pd # Pandas is used for dataset operation
import string # String is used for identifying parts of gram
from nltk import pos_tag # Identifies the Parts-of-Speach ta
from nltk.corpus import stopwords # A list of words that don
from nltk.stem import WordNetLemmatizer # Lemmatization is a
                                        # process word, like
from nltk.corpus import wordnet
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from progress.bar import Bar # Adding a progress bar to keep
import spacy # A superior form of the NLTK package which off
```

```
print("Packages Loaded\n")

# spaCy offers the computations operating on the GPU, which during development showed the
# If the user doesn't have a GPU, please comment the statement below
spacy.prefer_gpu()
nlp = spacy.load("en_core_web_sm") # A pretrained module which can tokenize the statements

# The function below cleans the text, so that the Sentiment Analysis model can process the
def clean_text(text):      Lakshya Kumar, 9 months ago • sentiment analysis python code
    # Lower text
    text = text.lower()
    # Tokenize text and remove puncutation
    text = [word.strip(string.punctuation) for word in text.split(" ")]
    # Remove words that contain numbers
    text = [word for word in text if not any(c.isdigit() for c in word)]
    # Remove stop words
    stop = stopwords.words('english')
    text = [x for x in text if x not in stop]
    # Remove empty tokens
    text = [t for t in text if len(t) > 0]
    # POS tag text
    pos_tags = pos_tag(text)
    # Lemmatize text
    text = [WordNetLemmatizer().lemmatize(t[0], get_wordnet_pos(t[1])) for t in pos_tags]
    # Remove words with only one letter
    text = [t for t in text if len(t) > 1]
    # Join all
    text = " ".join(text)
    return(text)
```

```
# The function below works with the clean_text funct
def get_wordnet_pos(pos_tag):
    if pos_tag.startswith('J'):
        return wordnet.ADJ
    elif pos_tag.startswith('V'):
        return wordnet.VERB
    elif pos_tag.startswith('N'):
        return wordnet.NOUN
    elif pos_tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN
```

```
# This function will convert the compound socre generated by
# More labels can be added based on the user's preference
# The limits set are arbitrary and can be modified based on
def sentiment_sen(x):
    if x >= 0.00 :
        return("Positive")

    elif x <= - 0.10 :
        return("Negative")

    else :
        return("Neutral")

# Load the dataset; dataset must be a CSV form and the user
# The statements that are to be processed must belong in the
reviews = pd.read_csv("metacritic_critic_reviews.csv")
```

```
reviews = pd.read_csv("metacritic_critic_reviews.csv")

print("#################################START#################################")
print("\nFormat Cells")

rev = reviews['review'] # Select the review column only

print("\nNormalise Reviews")
bar = Bar('Processing', fill = "|", max=len(reviews), suffix='%(percent).1f%%') # Set up a progress bar to trac
for i in range(len(reviews)):
    reviews.loc[i,'n_review'] = clean_text(reviews.loc[i,'review']) # Each statement is cleaned and tokenized a
    bar.next()
bar.finish()

print("\nSentiment Analysis")
sid = SentimentIntensityAnalyzer()
# All clean/normalized text are provided as input to the model and the polarity socres are generated
reviews["sentiments"] = reviews["n_review"].apply(lambda x: sid.polarity_scores(x))
reviews = pd.concat([reviews.drop(['sentiments'], axis=1), reviews['sentiments'].apply(pd.Series)], axis=1) # A

reviews["sentiment"] = reviews["compound"].apply(lambda x: sentiment_sen(x)) # The socres generated are convert
```

```
print("#################################ASPECT BASED SENTIMENT ANALYSIS#################################")

# The following function finds sentiments, similar to the code segment above, but are only applied for words or
def find_sentiment(text):
    text = clean_text(text)
    sid = SentimentIntensityAnalyzer()
    sentiments = sid.polarity_scores(text)
    return sentiment_sen(sentiments['compound'])

aspects = [] # Aspects are the parts of the statement that the sentiments are reffering
sentences = reviews['review'].to_list()
print("Aspect Sentiments")
# Following couple lines of code are for the progress bar to keep track of the progress
length = len(sentences)
bar = Bar('Processing', fill = "|", max=length, suffix='%(percent).1f%%')
# Each review is analyzed to identify the aspects, the words that provide its sentiments and the sentiments tow
for i in range(length):
    sentence = sentences[i]
    doc = nlp(sentence)
```

```
    for token in doc:
        descriptive_term = ''
        target = ''
        # All sentiment delivering words are Adjectives and it describes Nouns
        # The following lines of code look at the words in the statement and if they are nouns that have links
        # And if the words are adjectives and are not linked to any adverbs then they are the descriptive terms
        if token.dep_ == 'nsubj' and token.pos_ == 'NOUN':
            target = token.text
        if token.pos_ == 'ADJ':
            prepend = ''
            for child in token.children:
                if child.pos_ != 'ADV':
                    continue
                prepend += child.text + ' '
            descriptive_term = prepend + token.text
        aspects.append({'aspect': target,
            'description': descriptive_term,
        'sentiment': find_sentiment(descriptive_term)}) # Adds the infromation in a dictionary format contining

    bar.next()
bar.finish

reviews['aspect_sentiment'] = pd.Series(aspects) # Converts to a pandas seeries and adds it to the final result

reviews.to_csv("reviews_sentiment.csv") # Saves the resulting dataset as a CSV file
print("\nReviews with overall sentiment analysis and aspect based sentiment analysis saved as a .csv file")
print("##################################FINISH####################################")
```