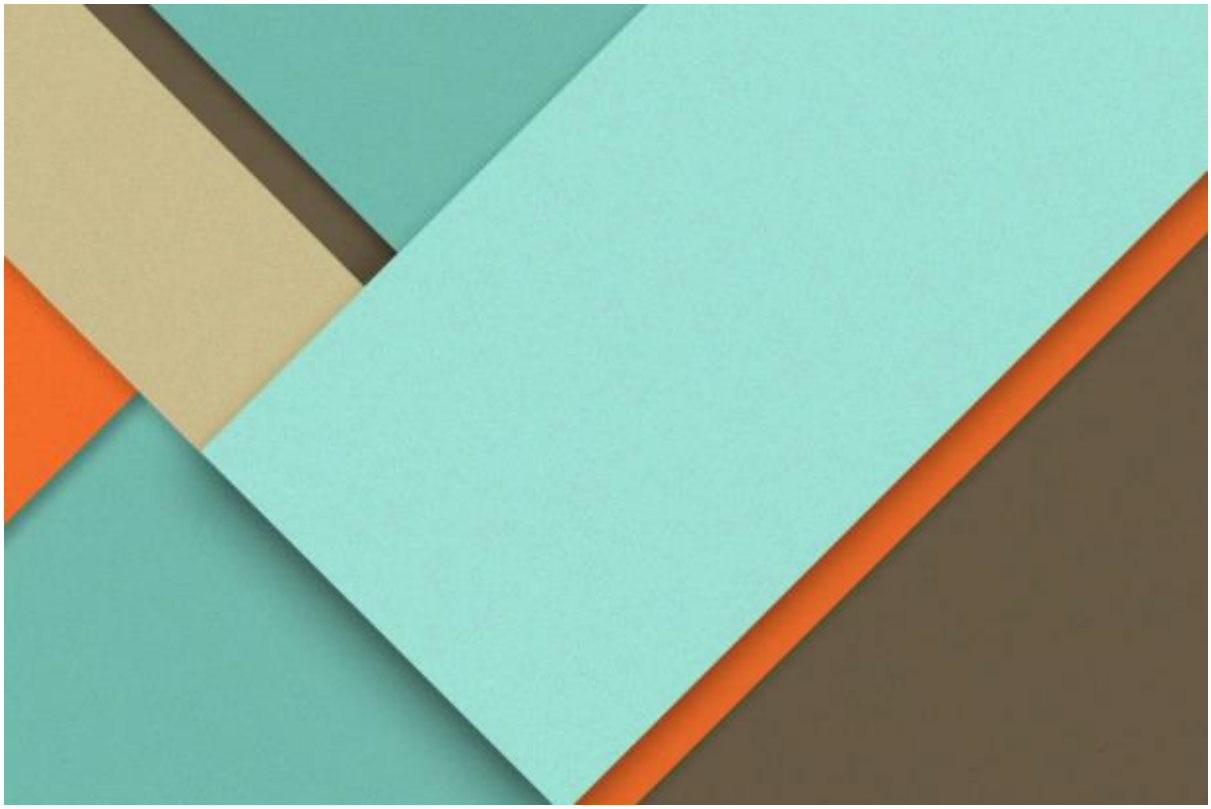
Project Report



Aspect-Based Sentiment Analysis

Team : Sifael Sebastian Ndandala, Piumi Abeyrathne, Protogen Hahirwabayo

Contents

1. Project Overview

NLP, or Natural Language Processing, is a field of AI concerned with enabling computers to comprehend and generate human language effectively. It encompasses the development of algorithms and models for interpreting and producing meaningful text interactions. Aspect-Based Sentiment Analysis (ABSA) is a cutting-edge natural language processing (NLP) technique that aims to analyze and understand opinions expressed in text at a more granular level. Unlike traditional sentiment analysis, which provides an overall sentiment score for a segment of text, ABSA focuses on identifying specific aspects or attributes of a product, service, or experience and analyzing the sentiment expressed towards each aspect individually. This project aims to develop an ABSA system capable of automatically extracting aspects and their associated sentiments from user reviews.

The main objective of this project is to build a robust and accurate ABSA model that can effectively identify and analyze aspects mentioned in user reviews and determine the sentiment polarity (positive, negative, or neutral) associated with each aspect. By understanding the nuanced opinions expressed by users towards different aspects of a product or service, businesses can gain valuable insights into customer preferences, strengths, and weaknesses, ultimately improving customer satisfaction and product quality.

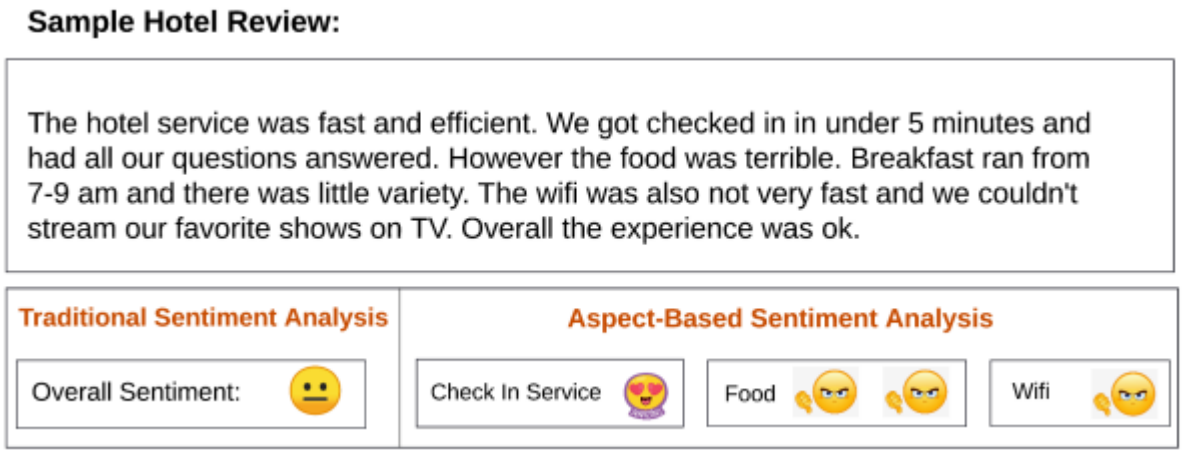


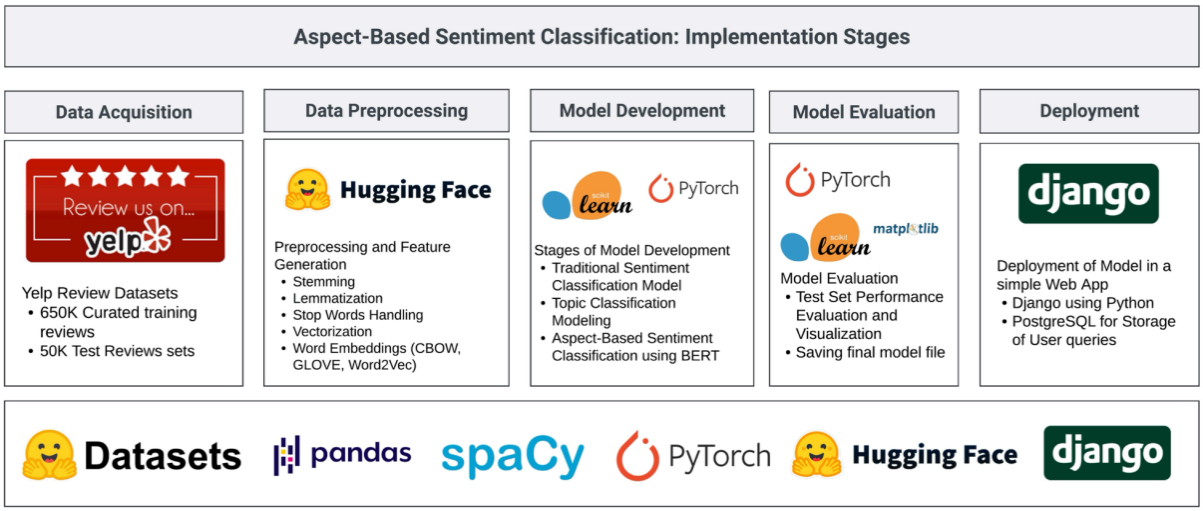
Figure 1 : Sample Hotel Review

This project aims to take traditional sentiment analysis a step further, to dissect customer reviews into specific sentiment details thereby providing granular and actionable insights into what aspects customers liked or disliked about the product and service.

2. Implementation

The implementation of the Aspect-Based Sentiment Classification project consist of five key stages,such as Data Acquisition, Data Preprocessing, Model Development, Model Assessment & Improvement, and Deployment.

In the following sections, we will discuss the implementation of each stage, highlighting interesting findings, the techniques employed, their outcomes, and the challenges faced that led to informed changes in our approach.



2.1 Data Acquisition

Data acquisition refers to the process of collecting and gathering data from various sources for further analysis, processing, or storage. It involves capturing data from sensors, instruments, devices, databases, or any other sources that produce or store information. The collected data can be in various forms such as analog signals, digital signals, text, images, videos, or any other format depending on the nature of the source.

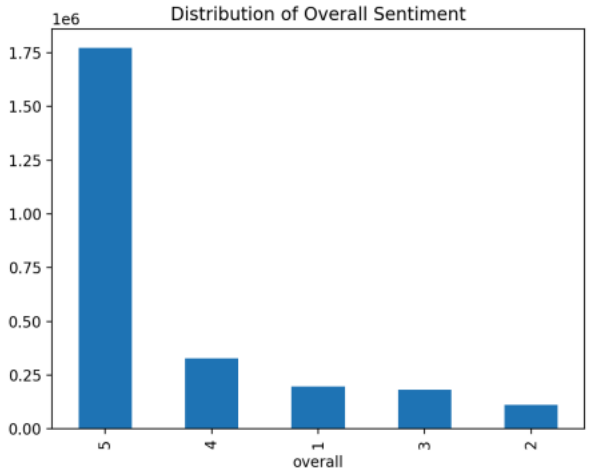
Initially, we decided to utilize the Amazon product reviews dataset as our primary source of data. This dataset is well-known for its accessibility and comprehensive coverage across a diverse product categories. The broad spectrum of categories presented within this dataset offer opportunity for us to develop robust models capable of addressing various aspects of reviews across multiple domains.

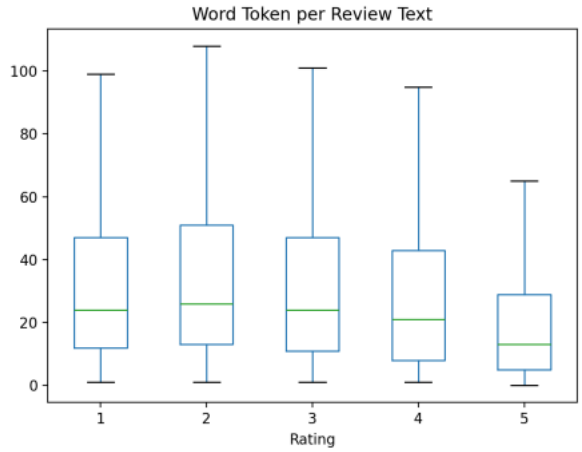
During our analysis of the Amazon Reviews dataset, we encountered a notable issue regarding class imbalance across all product categories. Specifically, approximately 85% of the reviews were categorized as positive, receiving either 4 or 5 stars. Neutral reviews, receiving 3 stars, accounted for roughly 6% of the dataset, while negative reviews, rated 1 or 2 stars, constituted the remaining approximately 9%.

This prevalence of positive reviews may suggest a trend where more engaged or satisfied customers are inclined to provide feedback. However, this imbalance poses a significant challenge to the development of a robust model. onversely, we found that the distribution of word tokens, normalized to account for outliers, demonstrated a relatively uniform pattern. Across reviews, there was a median

of approximately 20-25 words per review. Interestingly, positive reviews exhibited a slightly more concentrated distribution in comparison.

Amazon Review dataset:

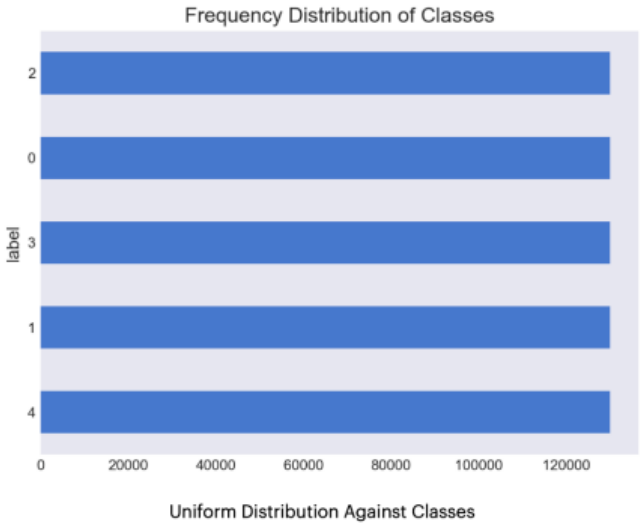




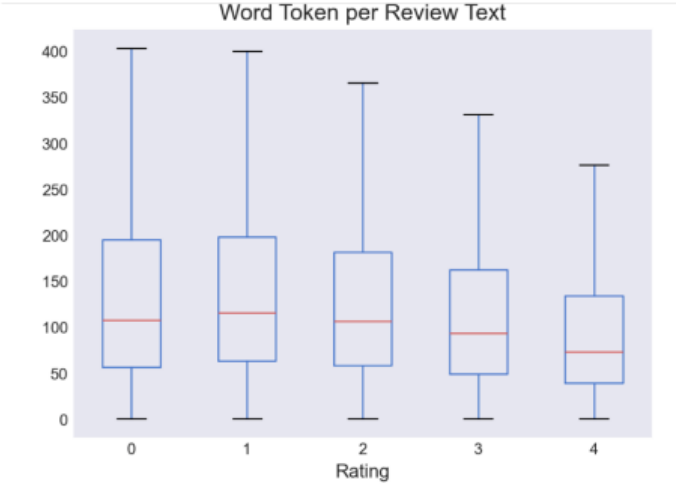
Acknowledging the hurdles presented by the overwhelming majority of positive reviews in the Amazon Review dataset, we embarked on a quest for an alternative dataset offering a more balanced distribution. Preferring to sidestep potential biases from artificial rebalancing methods, our exploration led us for the Yelp Reviews dataset.

Renowned for its balanced composition, the Yelp dataset has undergone a meticulous curation process, bolstering our confidence in the quality and dependability of our exploratory and analytical pursuits.

Yelp Review dataset:



Uniform Distribution Against Classes



Comparable Token to Review across Classes

3. Implementation

Our preprocessing pipeline is designed to clean the data and ready it for feature extraction. The key steps include understanding the dataset structure, analyzing the distribution of ratings, checking for punctuations and special characters, replacing numbers with words, creating a word cloud, lemmatizing, tokenizing, removing stop words, employing N-grams, using Bag of Words & CountVectorizer, and utilizing TF-IDF vectorizer..

-------An essential transformation involves replacing numerical values with their corresponding textual representations, preserving their semantic meaning within the text data. Additionally, employing word cloud visualizations offers a succinct representation of prevalent words, aiding in identifying dominant themes or topics.

Further refinement is achieved through lemmatization, a process of reducing words to their base form, thereby standardizing vocabulary and reducing word variations. Tokenization, another critical step, dissects the text into individual tokens or words, forming the basis for subsequent analysis.

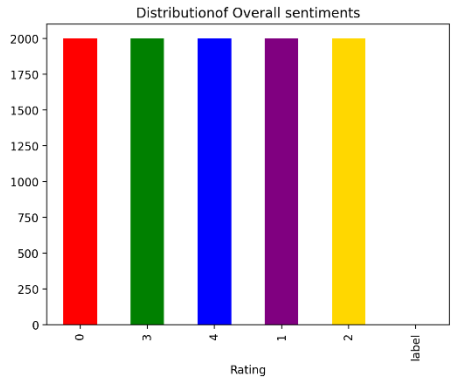
Removing stop words—common words lacking substantial meaning—curtails noise, refining the relevance of extracted features. Incorporating n-grams captures contextual information, enriching the text representation for tasks like language modeling or text classification.

For numerical feature generation, the Bag of Words approach with CountVectorizer tallies word frequencies to construct numerical feature vectors. Moreover, Term Frequency-Inverse Document Frequency (TF-IDF) vectorization assigns weights to words based on their frequency across documents, a technique widely embraced for text feature extraction.

In the appendix, we provide the complete implementation of our Preprocessing Class, showcasing the operationalization of these preprocessing steps within our workflow. This comprehensive approach ensures our data is meticulously curated and primed for subsequent analytical endeavors.

3.1 Understanding the dataset.

This step encompasses reading the file, understanding the dataset's shape, checking for information such as data types, checking for null values, duplicates and distribution of rating. These functionalities ensure a comprehensive exploration and initial assessment of the dataset's integrity.



3.2 Check for Punctuations & Special characters.

In the **Preprocessor** class, the method responsible for cleaning the text removes punctuations and special characters. It specifically replaces the ampersand ('&') with the word "and" and utilizes a regular expression to substitute any non-alphabetical character with a space. This process ensures that the text is devoid of punctuation, simplifying subsequent processing steps such as tokenization and lemmatization.

```
def punctuation_removal(self):
    """Removes punctuation from the text, replacing '&' with 'and'."""
    self.text = self.text.replace('&', 'and')
    self.text = re.sub("[^a-zA-Z]", " ", self.text)
```

3.3 Replace numbers with words.

Here we use inflect library. Inflect is a Python library that provides tools for generating plural forms of nouns, converting numbers to words, and performing various other text inflection tasks. It offers functionalities for linguistic operations such as pluralization, singularization, number-to-word conversion, and generating ordinal numbers.

```
import inflect

p= inflect.engine()


def replace_with_words(match):
    number = match.group(0)
    return p.number_to_words(number)

def number_to_words(text):
    # Replace all numbers in the text with their word representation
    return re.sub(r'\b\d+\b', replace_with_words, text)
```

3.4 Stemming & Lemmatizing.

The `lemmatize_text` method converts each token into its base or dictionary form (lemma) according to its part of speech. This is done using `nltk`'s `WordNetLemmatizer` and a part-of-speech tagger (`nltk.pos_tag`). Lemmatization helps reduce the inflectional forms of words, thus supporting the uniformity of the dataset.

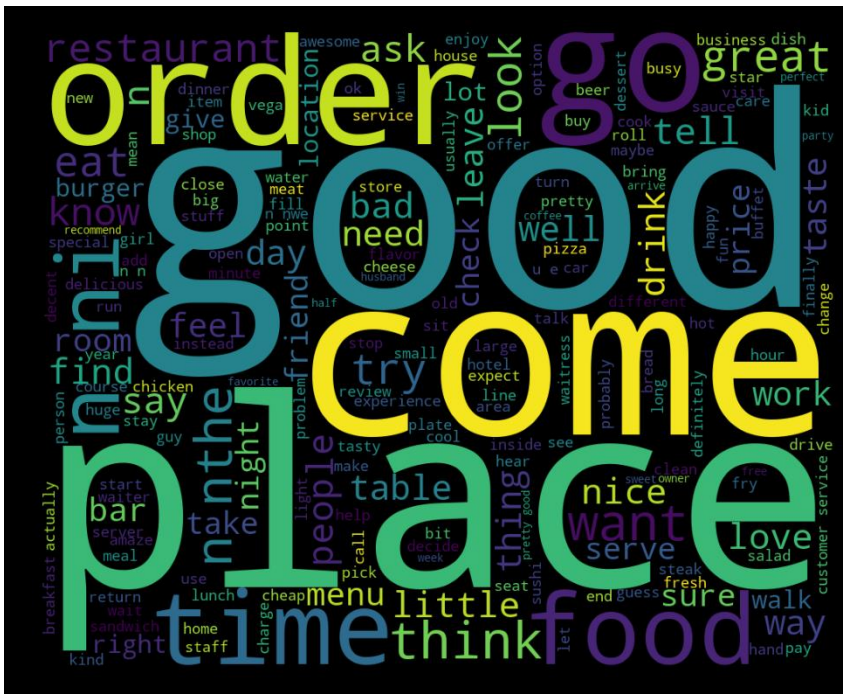
```
def lemmatize_review(text):
    new_sentence = []
    for token in nltk.pos_tag(nltk.word_tokenize(text)):
        # remove the stop words
        if token[0] not in stopwords.words('english'):
            pos = pos_dict.get(token[1][0].upper(), wordnet.NOUN)
```

```
            lemma = lemmatizer.lemmatize(token[0], pos=pos)
            new_sentence.append(lemma)

    return new_sentence
```

3.5 Word Cloud

A word cloud is a visualization technique used to represent text data in a visually engaging and intuitive manner. It displays the most frequently occurring words in a corpus of text, with the size of each word proportional to its frequency. Words that appear more frequently in the text are typically displayed with larger font sizes, making them more prominent and noticeable in the word cloud.



The largest word in the center is "**good,place,come,order**" suggesting it's a central theme or most frequently occurring term. Other prominent words include "**food**," "**drink**," "**service**," and "**restaurant**." The colors range from yellow, blue, green to purple, adding visual diversity and highlighting different words. The background of the image is black which makes the colorful words stand out more prominently. Words related to hospitality and dining experiences such as "**menu**," "**eat**," "**burger**," and "**pizza**" are visible, indicating these are key aspects of the corpus.

3.6 Tokenizing

The `tokenize` method splits the input text into individual words or tokens using the `nltk.word_tokenize` function. Tokenization is a fundamental step in text preprocessing as it converts the continuous text into discrete components (tokens), which are easier to analyze and process in subsequent steps such as lemmatization or spelling correction.

```
def tokenize(self):
    """Tokenizes the text for further processing."""
    self.word_tokenize = nltk.word_tokenize
    return self.word_tokenize(self.text)
```

3.7 Stop words

Stop words are common words that are often filtered out during text preprocessing in natural language processing (NLP) tasks. These words are typically extremely common and do not contribute much to the overall meaning of the text. Examples of stop words include "the", "and", "is", "in", "on", "at", "of", "to", "for", "a", "an", "with", "as", "are", "be", and so on.

The primary purpose of removing stop words is to reduce noise in the text data and focus on the more meaningful words that carry the main message or information.

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

sentence = "the restaurant at the city served amazing food"
print("With Stop words:", sentence, "\n" )

without_stopwords = [word for word in word_tokenize(sentence) if word
not in stopwords.words('english')]
print('Without Stop words:', ' '.join(without_stopwords))
```

3.8 N-grams

N-grams are contiguous sequences of n items, typically words, extracted from a text corpus. They are widely used in natural language processing (NLP) and text analysis tasks by considering sequences of words rather than individual words in isolation, N-grams help preserve the syntactic and semantic relationships between words in the text.

```
from nltk.util import ngrams
from nltk.tokenize import word_tokenize

document = "New York is truly an amazing city to live in"

[ ' '.join(gram) for gram in ngrams(word_tokenize(document), 1) ]
```

3.9 Bag of Words

The Bag of Words (BoW) model is a fundamental technique used in natural language processing (NLP) for text representation. It represents text data as a collection of unique words, ignoring grammar and word order, and focuses solely on the presence or absence of words in a document.

3.10 CounterVectorizer & TF-IDF vectorizer

Both CountVectorizer and TF-IDF Vectorizer are techniques used to convert text data into numerical representations suitable for machine learning algorithms. CountVectorizer is a text preprocessing technique that converts a collection of text documents into a matrix of token counts. It represents each document as a vector where each dimension corresponds to a unique word in the vocabulary, and the value represents the count of that word in the document.

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf_vectorizer = TfidfVectorizer( analyzer='word',          # Word
level vectorizer
                                    lowercase=True,           #
Lowercase the text
                                    tokenizer= word_tokenize) # Use
this tokenizer)

tfidf_vectorizer.fit(corpus)
tfidf_features = tfidf_vectorizer.transform(corpus).toarray()
```

4. Model Development

5. Model Deployment

6. References

1.