Explain you opinion (P5)
Project of Information Retrieval

Giulia Scarpa 26780A

A.A. 2023-2024

Abstract

This report analyzes a collection of Tweets to make an Aspect-Based Sentiment Analysis (ABSA). The sentiment analysis is performed using the Born Classifier[1], a text classification algorithm. The explanation features are extracted according to the Born algorithm. The purpose of the project is to exploit the Born explanation in order to use it for Aspect Based Sentiment Analysis. Using the list of words obtained from Born, we test different approaches for aspect extraction: dependency parsing, Word2Vec, and K-means clustering. Each model extracts candidate aspects and their respective sentences. After that, each sentence is associated to a new potential sentiment and used to compute the Born prediction and evaluate the goodness of the candidate aspects and the chose model.

Keywords Born Classifier, Spacy, Word2Vec, K-means

1 Introduction

Aspect-Based Sentiment Analysis (ABSA) has emerged as a crucial task in natural language processing, aimed at extracting sentiment information from text. This technique goes beyond document-level sentiment analysis by identifying and analyzing sentiment towards specific aspects or entities within the text. ABSA involves several steps including aspect extraction, sentiment classification, and often context modeling to capture the significative words in sentiment expression.

2 Dataset

The dataset is taken from *paperswithcode* website[2]. This is an entity-level Twitter Sentiment Analysis dataset. For each Tweet content, the task is to judge the sentiment of the entire sentence towards a given entity. The dataset contains 75682 rows with 4 columns:

[1] https://bornrule.eguidotti.com/

[2] https://paperswithcode.com/dataset/twitter-sentiment-analysis

- **tweet id:** containing the id of the users;
- **entity:** containing the categories of the Tweets (32 different categories);
- **sentiment:** the sentiment of the Tweets (Positive 20932, Negative 22624, Neutral 18393 and Irrelevant 13047);
- **content:** the text of the Tweets.

After removing rows with NaN values and duplicates, we are left with 59515 rows (Neutral 17879, Negative 21790 and Positive 19846). The 'Irrelevant' sentiment is dropped because it is not meaningful for the analysis.

3 Data Pre-processing

To enhance the performance of the analysis, the content of Tweets was pre-processed using the nltk library[3]. The pre-processing involved several steps to clean and standardize the text data.

1. All urls, user mentions and hashtags were removed from the text to focus on the text of the Tweets;
2. Special characters were dropped, and contractions were expanded (e.g., 'can't' becomes 'cannot') to normalize the text;
3. The text was verified to be encoded in ASCII to filter out any inconsistent Tweets;
4. All text was converted to lowercase to treat words uniformly;
5. The Tweets were tokenized (to obtain a list of words) for all;
6. Only alphabetic words were retained, removing tokens containing numbers or special characters;
7. Stopwords (common words that are usually ignored in text analysis) were removed using the nltk library's list of English stopwords;
8. The words were lemmatized, reducing them to their base or root form (e.g., "is" becomes "be"), which helps standardize the text and improve analysis quality;
9. Only words with a maximum length of 15 characters were kept to exclude unusually long words or potential noise.

After applying these preprocessing steps, the dataframe was cleaned by removing any entries with missing values NaN, obtained 48495 total entries.

[3] https://www.nltk.org/

4 Sentiment Analysis

Sentiment analysis, known as opinion mining, is the process of categorizing opinions expressed in text to determine the sentiment expressed by the author. It is the analysis of the polarity (positive, negative, or neutral) of a text, such as a review, social media post, or a customer feedback. Sentiment analysis uses natural language processing (NLP) techniques to extract information, providing insights into public opinion, user feedbacks, and trends in online contents.

4.1 Born Classifier model

The first task of this project is to perform sentiment classification using the Born Classifier model. Before this step, the texts were converted into an array of token (words) counts using the *'CountVectorizer'* function from the scikit-learn library[4]. This is a crucial pre-processing step to do before training a machine learning model. After that the dataset is randomically splitted into 80% training and 20% test set, obtaining 38796 entries for the training set and 9699 for the test set, with a vocabulary of 25132 words. After fitting the model, an accuracy of 77% was obtained. The classification report in Figure 1 describes

	precision	recall	f1-score	support
Negative	0.78	0.83	0.81	3669
Neutral	0.77	0.63	0.69	2693
Positive	0.76	0.82	0.79	3337
accuracy			0.77	9699
macro avg	0.77	0.76	0.76	9699
weighted avg	0.77	0.77	0.77	9699

Figure 1: Classification Report Born Classifier

the model evaluation metrics. As we can see in Figure 2, the diagonal of the confusion matrix presents a high number of labels. It means that most of the sentences were classified correctly.

4.2 Explanation Features

The Born classifier provides a function to obtain explanation features as a sparse matrix. Transforming this matrix into a dataframe results in a list of words along with their respective weights for each polarity category: Negative, Positive, and Neutral. The initial number of entries obtained is 25132. By filtering out the rows with zero weights and maintaining only the words that have a weight higher than the 80th percentile in at least one polarity category, the highest words remaining are 13055. In Table 1 we can see the words with the higher weight for each sentiment. These features will serve as an excellent starting point for extracting candidate aspects and identifying significant sentences where these

[4]https://scikit-learn.org/

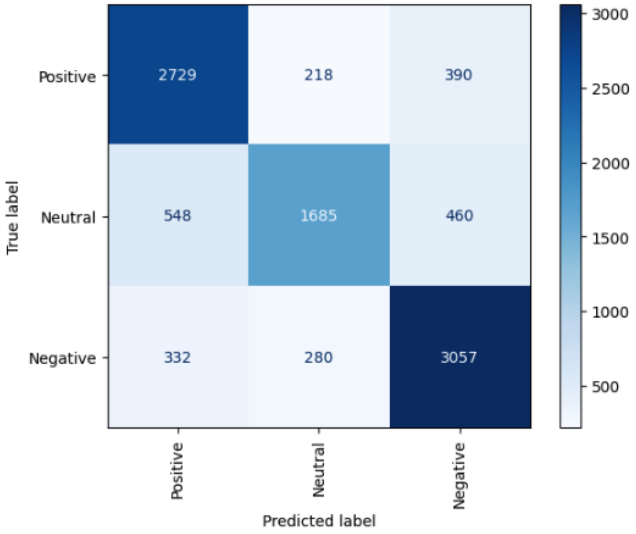

Figure 2: Confusion Matrix Born Classifier

aspects appear. In the next section, we will explore methods to extract candidate aspects taking in consideration these explanation features extracted from the Born classifier.

Positive	**Weight**	**Negative**	**Weight**	**Neutral**	**Weight**
love	0.044679	fix	0.056442	earned	0.049268
nice	0.036413	fuck	0.040436	check	0.037362
dope	0.034889	server	0.034199	achievement	0.027844
excited	0.034008	suck	0.030557	horrific	0.025757
wait	0.033493	trash	0.026053	vision	0.022568
beautiful	0.030758	wtf	0.024024	beep	0.021975
fun	0.030463	glitch	0.023769	love	0.021012
wow	0.030312	eamaddennfl	0.023463	orgrimmar	0.020387
thank	0.029645	broken	0.022154	summoners	0.019810
tuning	0.025124	shit	0.021913	panic	0.019745

Table 1: Most sentiment explanation features with their respective weights

5 Aspect detection

Aspect detection is a key component of Aspect-Based Sentiment Analysis (ABSA). It consists in identifying and extracting specific aspects or characteristics of an entity (such as a product, service or topic) mentioned in a text. These aspects are usually nouns or noun phrases that describe specific items of interest.

5.1 Dependency Parsing

The first approach for aspect extraction uses dependency parsing with text segmentation. The content of Tweets is pre-processed using the spacy library that creates a document object for each Tweet which contains linguistic annotations. There are two steps:

1. **Aspect Identification:** The function identifies aspects by examining specific grammatical relationships in each sentence. Firstly, it looks for tokens that serve as adjectival modifiers (denoted as amod) of nouns. When such a token is found, the head noun (the noun that the adjective modifies) is considered an aspect. For example, in the phrase "huge fan", "huge" is an adjectival modifier of the noun "fan," so "fan" would be identified as an aspect. Next, the function looks for tokens that are nominal subjects (denoted as nsubj) of verbs. When it finds such a token, the subject itself is considered an aspect. Additionally, it includes any direct objects (denoted as dobj) of the verb in the aspect segment. For instance, in the sentence "game is amazing", "game" is the nominal subject of the verb "be" and "amazing" is the direct object. Therefore, "game" is identified as an aspect, and "amazing" is included in the aspect segment.

2. **Aspect Segmentation:** Once the aspects have been identified, the function determines the span of text that covers the entire aspect segment. It calculates *aspect_start* as the starting index of the earliest token in the segment and *aspect_end* as the ending index of the latest token. Using these indices, the function extracts the relevant portion of the sentence, referred to as *aspect_sentence*.

The extracted sentence segment (*aspect_sentence*) is validated to ensure it contains more than one word, providing meaningful context. The function returns

Entity	**Aspect**	**Count**
RedDeadRedemption(RDR)	redemption	263
johnson&johnson	johnson	208
RedDeadRedemption(RDR)	game	137
TomClancysRainbowSix	game	133
Amazon	amazon	117
Borderlands	borderland	115
Overwatch	overwatch	106
TomClancysGhostRecon	game	103
Facebook	facebook	102
WorldOfCraft	warcraft	93

Table 2: Aspects computed with Dependency Parsing

a list of tuples, which contains the aspects and its respective sentences. After that we check if the candidate aspect is contained in the explanation features list

computed by Born. If present, it associates a new potential sentiment according to the aspect. This is computed by taking the maximum weight between the polarities of that aspect, that Born returns with the sparse matrix of the explanation features . The model was performed using the tqdm library[5] to monitor the computational time, that required 7.16 minutes. The result is a new dataset with 19590 entries, and in the Table 2 we see the most common aspects extracted according to their membership entity.

5.2 Word2Vec

The second approach employed for the aspect detection is the Word2Vec model. Word2Vec helps in capturing the semantic relationships between words, enabling a more accurate identification of aspects and their contexts in the text. First of all, the pre-processing function is responsable for cleaning and tokenizing the input text to improve the performance of the model and the aspect extraction process. It lemmatizes the words, removes tokens that are not alphabetic, stop words or punctuation and it selects only nouns. Focusing on nouns is important, because they often represent key entities and aspects in the text. The aspect extraction function identifies and extracts specific aspects from the text along with their corresponding sentence segments. For the aspects identification, the model uses the explanation features list, generated from Born, to filter out some potential aspects and only include those that have been successfully learned by the Word2Vec model. The resulting list of aspects contains only the features that the model recognizes, which are then used for aspect extraction in the subsequent analysis. The function for the aspect extraction initializes a dictionary, which maps each aspect to a list of sentences that contains the aspect. For each sentence, it checks if the sentence contains any of the predefined aspects. Moreover dependency parsing is used to find the syntactic relation between the

Entity	**Aspect**	**Count**
Borderlands	borderland	392
Facebook	facebook	359
RedDeadRedemption(RDR)	redemption	338
FIFA	fifa	309
Amazon	amazon	302
Overwatch	overwatch	300
WorldOfCraft	warcraft	292
TomClancysRainbowSix	game	256
RedDeadRedemption(RDR)	game	230
AssassinsCreed	creed	219

Table 3: Aspects computed with Word2Vec

aspect and the sentence, and identifies the subtree of tokens associated with the aspect. Finally, it extracts and stores the relevant sentence segments those

[5]https://tqdm.github.io/

contain the aspects. This approach ensures that the extracted sentences are contextually relevant and accurately capture the aspects . In the end, it associates a new potential sentiment, according to the aspect, computed the same way as before (see 5.1). The computation took 45.54 minutes and the result is a new dataset with 55765 entries. In the Table 3 the most common aspects extracted according to their membership entity are shown.

5.3 K-means

The third approach combines Word2Vec embeddings and semantic clustering to extract meaningful aspects from the data. This method leverages the strengths of both vector representation and clustering to enhance aspect extraction. Word2Vec embeddings is used to represent a list of features of interest extracted from the dataset, taking in consideration the Born explanation features list. This numerical representation allows us to compute the semantic similarity between words effectively. K-means clustering is applied to group these Word2Vec embeddings based on their semantic similarity. For this task, the number of clusters is 32, accordingly to the number of entities in the original dataset. K-means clustering partitions the data into clusters, where each cluster contains words that are semantically similar. This step is crucial, as it organizes the feature space into meaningful groups, facilitating aspect extraction. After computing the clusters, the aspect extraction follows what has already been done previously for the Word2Vec model (see 5.2). The aspects are the representative words within each cluster, indicative of the topics. For the sentences associate with that aspect took 37.14 minutes and the final dataframe had 22576 rows, each representing an aspect, its associated sentence, and sentiment information. We can see in the Table 4 the most common aspects extracted according to their membership entity.

Entity	**Aspect**	**Count**
Borderlands	borderland	392
Facebook	facebook	359
FIFA	fifa	309
Amazon	amazon	302
AssassinsCreed	creed	219
WorldOfCraft	earned	200
WorldOfCraft	achievement	118
ApexLegends	apex	114
PlayerUnknownsBattlegrounds(PUBG)	banned	111
PlayerUnknownsBattlegrounds(PUBG)	ban	109

Table 4: Aspects computed with K-means

6 Results

The sentiment analysis prediction applied to the dataset obtained from the first approach demonstrates a robust performance with an accuracy of 77%. It excels in identifying negative sentiments with high precision and maintains good performance for neutral sentiments. While the model shows good performance for positive sentiments, its precision in this class could be improved. The Word2Vec approach shows a solid performance in sentiment analysis with an accuracy of 76%. The class-wise results show that the model is particularly effective in identifying negative sentiments, performs adequately for neutral sentiments, and has a good but improvable performance for positive sentiments. The K-means shows strong and balanced performance in sentiment analysis, with an accuracy of 80%, higher than the previous predictions. The class-wise results reveal that the model is particularly effective in identifying negative sentiments and performs adequately for neutral and positive sentiments. The macro and weighted average metrics further confirm the model's robustness and reliability in sentiment classification tasks. All methods identify common and relevant aspects, but Word2Vec and K-means methods extract a larger variety of aspects. K-means method provides the highest count for individual aspects, suggesting better identification of key aspects. K-means method achieves the highest coherence score (0.65), indicating that the aspects extracted are more semantically related and contextually meaningful, Word2Vec also performs well with a coherence score of 0.60, while dependency parsing has the lowest coherence score (0.45), implying less semantic consistency within the aspects.

7 Conclusions

In this report, we have conducted a comprehensive analysis of Tweets using Aspect-Based Sentiment Analysis (ABSA) with the Born Classifier. The Sentiment Analysis conducted by Born has shown good performance. The dependency parsing had limitations in capturing the semantic richness of the text, which affected the variety and accuracy of aspect extraction, probably for the simplicity of the approach, that in fact required a less computational time to compute. Dependency Parsing is not so good in coherence terms but can still be useful for more straightforward aspect extraction tasks. Using Word2Vec embeddings, we captured semantic relationships between words, enhancing the quality of aspect extraction. K-means Clustering emerges as the most effective method for aspect extraction, yielding the highest coherence scores and extracting an high range of relevant aspects. This method produced the highest accuracy of 80% in sentiment prediction. The clustering process ensured that the aspects were semantically coherent, enhancing the relevance and quality of the analysis. Overall, our analysis indicates that integrating Word2Vec with K-means clustering significantly improves the performance of ABSA. This combination captures both the semantic richness and structural relationships within the text, leading to more accurate and meaningful aspect extraction.