IT1244 Project Report: Airline Sentiment Clustering Dataset

Team 26: Gerald James Low Ming Tuck, Jayden Lim, Lim Zhao Xun Jerrell, Tan Min Kai, Edsel

Introduction

Understanding customer sentiment is essential for airlines to enhance service quality and customer satisfaction. In this project, we aim to classify sentiments expressed in tweets related to six major US airlines using Natural Language Processing (NLP) techniques and clustering algorithms. The goal is to identify and group positive, neutral, and negative tweets automatically, alongside differentiating between high and low confidence tweets. Sentiment analysis has been performed using rule-based approaches, which means use an existing corpus of words whose sentiments have already been determined to extract features from each data, then apply rules on these features to output a cluster (Coletta, Silva 2014).

Rule-based approaches for sentiment analysis, often limited by predefined keywords and lexicons, struggle with the dynamic nature of social media language. They are unable to capture slang, context, or emerging expressions effectively, and struggle with ungrammatical sentences (Zhang & Moldovan, 2018). Most importantly, rules-based approaches do not capture semantic relations between words, rather focuses only on the word itself and not its context. This makes them less suitable for analyzing evolving sentiment patterns in tweets. Models which attempt to capture semantic information have been proposed. For instance, Royyan & Setiawan (2022) introduced Feature Expansion with Word2Vec and TF-IDF to enhance sentiment analysis performance by creating word embeddings that better capture context and semantics, demonstrating the potential of using advanced NLP methods over traditional rule-based approaches for sentiment analysis and findings underline the benefits of using embeddings to represent words in vector space, facilitating more accurate clustering of sentiments based on semantic similarity. Still, on social media networks, much of the context is also carried through links and response chains. A tweet indicating agreement with a subject could arise from different sentiments, depending on the sentiment of the main tweet. While algorithms take into account links and connections that have been used (Qi, Aggarwal, Huang, 2014), in the absence of such information provided, these are just noise which greatly hampers the effectiveness of the model.

In this paper, we explore a mix of density-based clustering, hierarchical clustering, and distance-based clustering to determine the most appropriate way to split up the dataset of tweets. We also experimented with two word embedding algorithms, the TF-IDF and the Skip Grams before settling on the Skip Grams model for subsequent clustering and the TF-IDF model for noise detection.

Dataset

The dataset ‘Tweets.csv’ comprises of Twitter posts of airline passengers detailing their experiences with six different US airlines. It consists of 14,640 tweets and is separated into three columns: airline sentiment (positive, negative or neutral), confidence level (1 or less than 1) and the raw text of the Twitter posts. The posts contained noisy elements, such as handles, emojis, and URLs, which were removed to prevent the disruption of the training process. Latent links and references to other tweets were also missing from the data set which makes it difficult to ascertain the sentiment of some tweets, whose information is contained mainly in the other tweets it was referencing.

Data Preprocessing and Data Handling

Our preprocessing of the raw tweet text started by removing unnecessary elements like user handles, URLs, emojis, digits, and special characters. While we do recognize that hashtags may represent some sentiment, most of them did not seem to have any relevance. Hence, we removed them as well. A filter for airport destination codes was also used throughout the tweets. This helped to eliminate noise and focus on meaningful content. Text was converted to lowercase to ensure uniformity, which makes further analysis and feature extraction more consistent.

To facilitate evaluation using Nu**mPy** and **scikit** we converted the sentiment labels into numeric values—**positive** to 0, **negative** to 1, and **neutral** to 2. This transformation allowed us to easily compare and analyze the clusters' alignment with the actual sentiments of the tweets.

Methodology

Word Embedding

We trained a Skip Gram’s model on the cleaned tweet text to generate numerical representations of the words. The

Skip Gram's model generates a simple neural network which first aims to predict the words before and after a particular input word, and updates its weights based on the accuracy of this prediction. This gives a distinct advantage over the rules-based or purely frequency-based approach mentioned in the introduction. Each tweet was converted into a 400-dimensional vector, capturing the semantic relationships between words. (Existing studies have used embeddings from 50 to 500 dimensions, the choice of 400 was determined through repeated evaluation and a local search.) This enabled the clustering algorithms to group tweets based on the similarity of their word meanings, rather than merely on their surface text. An existing corpus of common English words, known as "Stop Words", along with the names of each US airline were left out of the model because these words do not give meaningful features on the embedded vector space.

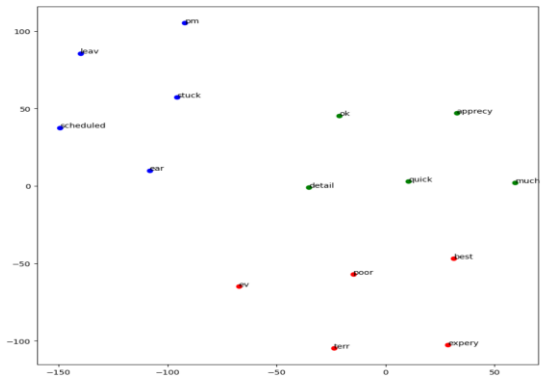

Fig 1. Word2Vec Embedding for Positive Words (Green), Negative Words (Red), Words Relating to Punctuality (Blue)

In addition, we used TF-IDF (Term Frequency-Inverse Document Frequency) to represent tweets as numerical vectors. This method returns a weighted frequency count of words that emphasizes the importance of words that are frequent within a tweet but rare across the entire dataset, making it useful for distinguishing between different clusters. For parity, the TF-IDF embedding was also scaled down to 300 dimensions using Latent Semantic Analysis (LSA), which is effective in capturing semantic relationships between words. Nguyen et. al. (2021) determined that TF-IDF is advantageous for clustering as it converts text data into meaningful vectors, capturing the significance of words within a document relative to their rarity across the corpus. By emphasizing unique terms within tweets, TF-IDF helps cluster similar tweets while reducing noise from common words. This representation aligns well with clustering methods like K-Means and DBSCAN, which rely on numerical differentiation to separate clusters. The paper asserts that TF-IDF's ability to highlight distinct words makes it particularly suitable for separating tweet sentiments in an unsupervised context.

Clustering Algorithms

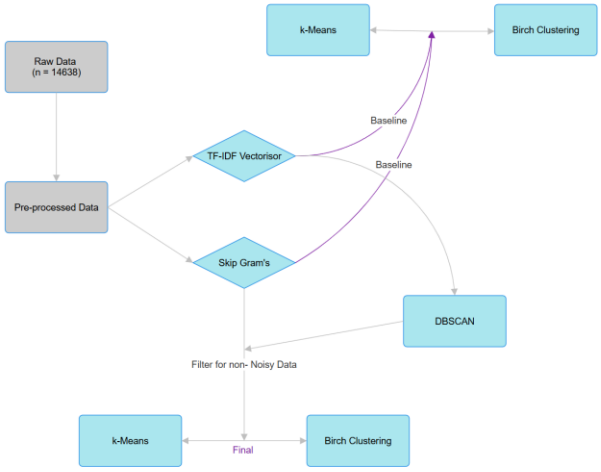

Fig 2. Flowchart of our Two-Step Clustering Process

Taking reference from Kremers' et al (2021), whose team used a combination of k-means and Density-Based Spatial Clustering of Applications with Noise (DBSCAN) to reduce the dimensionality and size of a data set, we have also decided to use a two-step clustering algorithm.

Firstly, we perform a clustering using DBSCAN, a density-based algorithm. Density based algorithms work by looking at the neighborhood of each data set and expanding the cluster if the neighborhood has a high density of points. It is effective because the clusters can now take on any shape and data points which are too sparse to be meaningful can be taken as noise. DBSCAN was applied to both the TF-IDF and Skip Grams Embeddings while varying the noise parameters "eps" and "min_samples". These parameters control how sparse the data points must be before they are classified as noise. It was found the TF-IDF routinely performed better (see the next section on "Metrics") than the Skip-Gram's as the noise parameters varied. One reason for this is that the vectors generated from TF-IDF more easily capture features relating to the broad topic of the tweets rather than sentiments. This helps to filter out tweets about rarer or difficult to understand topics. Semantic analysis using Skip Gram's is not needed at this stage. Doing a local search for the optimum parameters for TF-IDF vectors, we settled on (eps = 0.4, min_samples = 16).

Because the number of clusters is not controlled in DBSCAN, the 3 sentiments cannot be directly read off from the output. One approach is to enrich all the non-noisy data with their DBSCAN cluster labels, then perform k-means on each subcluster, and then merge the sub clusters together. However, without additional information on the nature of the clusters, it is unclear how the merging should proceed. Instead, we decide to re-group all the points which were not classified as noise and perform a single final clustering on them as a whole. The other "noisy" points will be assigned to the cluster whose centroid is closest to it. At the optimum noise parameters mentioned, about 10% of the original dataset was used for the final clustering.

Two candidates were considered for the final clustering algorithm, k-means and Birch clustering. K-means is a distance-based algorithm that assigns cluster labels based on the cluster centroid which is closest to the data point. The centroids are then re-computed after each assignment, and repeats until the cluster labels stay mostly constant. However, cosine similarity was used instead of Euclidean distance as it is widely preferred in text-based clustering. Cosine similarity measures the directionality of two vectors rather than magnitude, hence avoiding large distances that could affect our algorithms.

$$S_C(A, B) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

Birch clustering is a hierarchical algorithm that builds a tree structure from data. Each node in the tree is a cluster and clusters are merged continuously based on a set threshold. Its benefit over k-Means is that it is much more efficient with large datasets and clusters of varying sizes, which makes it scalable. It also tends to produce more consistent results because the clustering is not dependent on the initialization state (k-Means can produce different results depending on where the centroids are first set).

Metrics

The metrics used to evaluate the goodness of our clusters can be divided into informed and uninformed metrics. Informed metrics are calculated using the labels of the data while uninformed metrics measure the goodness of the model without reference to the true labels. Informed metrics include:

Homogeneity: Assessing the extent to which a cluster has only members belonging to a single class.

Completeness: Assessing the extent to which a class has members which are all contained in a single cluster.

V-measure: The harmonic mean between homogeneity and completeness.

Adjusted Rand Index: Measures the similarity between the actual labels and the predicted labels.

For these informed metrics, the range of scores is from 0 to 1. A score of 0 means that the model has performed at its poorest while a score of 1 indicates perfection.

For uninformed metrics, we used the **Silhouette Coefficient,** which is calculated with the mean intra-cluster distance and the mean nearest-cluster distance, for each sample. The scores range from –1 to 1. Negative values indicate that samples are wrongly assigned, while positive values indicate that the samples are correctly assigned. Values near 0 mean that the clusters are overlapping.

Entropy: The distances of a datapoint from the centroids produced in k-Means and Birch clustering was calculated and passed through a soft-max function. This gives a heuristic for the likelihood that a point belongs to a cluster, based on the relative distances from the centroids. The entropy of this prediction is hence a measure of how close these probabilities are to a "pure" vector (1,0,0) or (0,1,0) or (0,0,1). For the purposes of this project, the data points whose entropy fall in the lowest 25% amongst all data points are given the label: "Confident". These metrics, with the exception of entropy, were calculated and recorded using the **scikit** metrics library.

Results & Discussions

We conducted our prior model by feeding all the data into a k-Means algorithm and the Birch algorithm, without filtering. The results are summarized in the table below.

(n = 10 runs)	k-Means	Birch
Homogeneity	0.0628	0.0155
Completeness	0.0544	0.0757
V-Measure	0.0576	0.0277
Adj. Ran Index	0.2834	0.3378
Silhouette Coef.	0.0307	0.0384

Table 3. Metrics Before DBSCAN Clustering

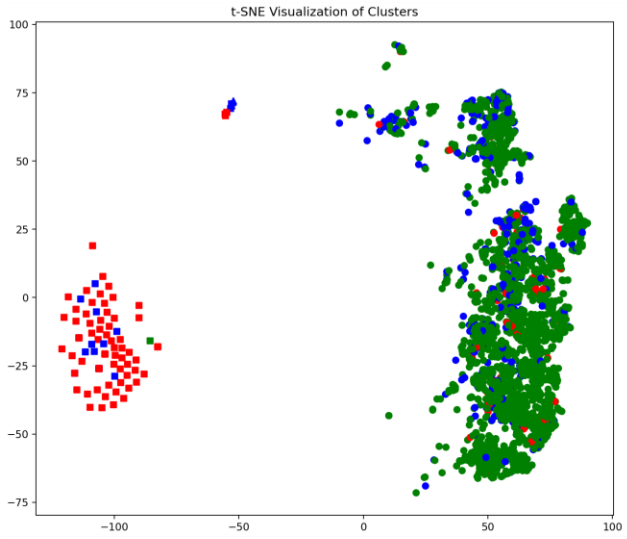

Fig 4. Clusters Showing Only the Confident Data Points using K-Means (After DBSCAN)

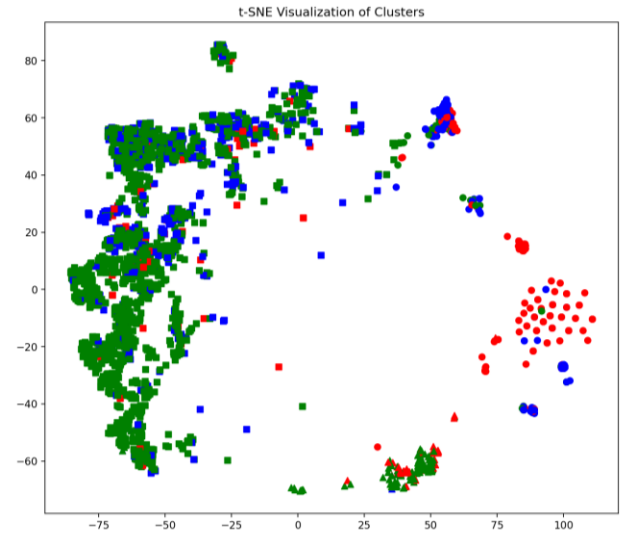

Fig 5. Clusters Showing Only the Confident Data Points using BIRCH (After DBSCAN)

Fig 4 shows the plot of all "Confident" data points after the final k-Means algorithm. Each color represents the *true sentiments* as provided with the data, and each shape represents a predicted cluster. Fig 5 shows the same plot for the Birch algorithm. We can see that if we only filter out the "Confident" points, there is a clear separation between the red point (true positive) and the green and blue points (true negative and true neutral). This also shows much of the difficulty of this task lies in differentiating between neutral and positive sentiments. Table 6 shows the same scores after performing one pass of DBSCAN on the data. Generally, both models showed improvement in the V-measure after DBSCAN filtering. Although the DBSCAN did produce a higher V-measure, this is partly because it generated more clusters and hence resulted in greater separation of data points. It is interesting to note that Birch model had a much worse performance before filtering, which could be because cluster features are harder to form when the data set is large and noisy.

(n = 10 runs)	DBSCAN	k-Means	Birch
Homogeneity	0.0826	0.0600	0.0548
Completeness	0.1103	0.0721	0.1096
V-Measure	0.0982	0.0695	0.0712
Adj. Ran Index	0.1301	0.4532	0.4053
Silhouette Coef.	-0.0854	0.1249	0.5279

Table 6. Metrics After DBSCAN Clustering

The final model chosen is the Birch Algorithm due to its slightly higher V-measure and completeness score.

Final model

The best 2-step BIRCH clustering model that was trained out of all the test runs was saved using the **pickle** library. The model and the clusters it produced can be found in the "birch_model.pkl" file and we also included the Kmeans algorithm in "kmeans_model.pkl" for reference. The Appendix provides details on this model's performance and the cluster visualization.

Impacts on Society

For better or for worse, our model is not yet as accurate as a human manually reading tweets to determine their sentiments. Still, it can be argued that even a sub-par accuracy makes the model worthwhile due to the sheer overwhelming amount of data that makes manual processing of all of them impossible. Models can help in the prediction of "certain" points and leave uncertain data points for humans. The development of such powerful technology to determine user sentiments poses a threat to privacy because users may unwittingly disclose information that is extracted as features in such algorithms to the detriment of their well-being (such as targeted political advertisements).

References

Coletta, L. F. S.; Silva, N. F. F.; Hruschka, E. R.; Hruschka E. R. Jr. 2014. *Combining classification and clustering for tweet sentiment analysis. Proceedings of the Brazilian Conference on Intelligent Systems (BRACIS), 89-95.* https://doi.org/10.13140/2.1.1060.928.

Nguyen, T. H.; Pham, T. K. N.; Bui, T. H. M.; Nguyen, T. Q. C. 2021. *Clustering Vietnamese conversations from Facebook page to build training dataset for chatbot. Nha Trang University. Preprint, arXiv:2112.15338v2*.

Qi, G. J.; Aggarwal, C. C.; Huang, T. 2012. *Community detection with edge content in social media networks*. Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, and IBM T.J. Watson Research Center.

Royyan, A. R.; Setiawan, E. B. 2022. Feature Expansion Word2Vec for Sentiment Analysis of Public Policy in Twitter. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *6*(1), 78 - 84. https://doi.org/10.29207/resti.v6i1.3525.

Tian, Z.; Ramakrishnan, R.; Livny, M. 1996. *BIRCH: An efficient data clustering method for very large databases*. Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data (SIGMOD '96), 103-114. Montreal, Canada: Association for Computing Machinery. https://doi.org/10.1145/233269.233324.

Zhang, L.; Moldovan, D. 2018. Rule-based vs. Neural Net Approaches to Semantic Textual Similarity. In *Proceedings of the First Workshop on Linguistic Resources for Natural Language Processing*, pages 12–17, Santa Fe, New Mexico, USA. Association for Computational Linguistics. https://aclanthology.org/W18-3803.