Text Mining

Alberto Odierna (907341), Biagio Spiezia (920172)

2025

Indice

1 Introduction

The BBC (British Broadcasting Corporation) is one of the world's leading news broadcasters, the exclusive concessionaire and publisher of public service broadcasting in the United Kingdom. It is also the oldest national broadcaster in the world, founded in 1922. Its archive constitutes one of the most extensive and varied resources for the study of texts. The dataset used in this paper is a subsample of its archive. In this work, two of the most widely used techniques in the field of text mining are described and applied in our case study: text classification and text summarisation. Text classification makes it possible to catalogue news items into predefined categories, facilitating the organisation and search of content, a technique that could be particularly useful for quickly identifying the subject matter of an article and facilitating archiving. Text summarisation, on the other hand, makes it possible to generate automatic summaries of texts, offering an overview of the content quickly and effectively. This technique is crucial in contexts where time is critical, allowing users to acquire essential information without having to read the entire article.

2 Dataset and Preprocessing

The BBC News Archive dataset consists of a sample of articles containing BBC news, collected to be used as a benchmark for machine learning research. It consists of 2225 articles produced between 2004 and 2005, extracted from the BBC website, and the following variables: news title, text file name, news content and news category. Each category corresponds to a specific thematic area, five in total: business, entertainment, politics, sports and technology. After importing the dataset as a preliminary step, a check for duplicates was performed. A total of 133 duplicate articles were identified and eliminated. During the pre-processing steps for the normalisation phase, the text was converted into lower case characters in order to standardise it. Subsequently, the text was divided into tokens, which represent the fundamental units for the subsequent analysis steps. Next, stop words - very frequent words that make it difficult to extract information about the context of the text - were excluded from the text. In addition to the words contained in the Python stopwords library, we defined a further set of words to be removed: ‘said’, ‘may’, ‘many’, ‘made’, ‘say’, ‘take’, ‘think’, ‘use’, as they are uninformative for distinguishing the context of an article and the respective subject areas. Non-alphanumeric characters, such as punctuation, were also eliminated. Finally, in the last step of lemmatisation: each word is reduced to its basic form in order to unify lexical variants.

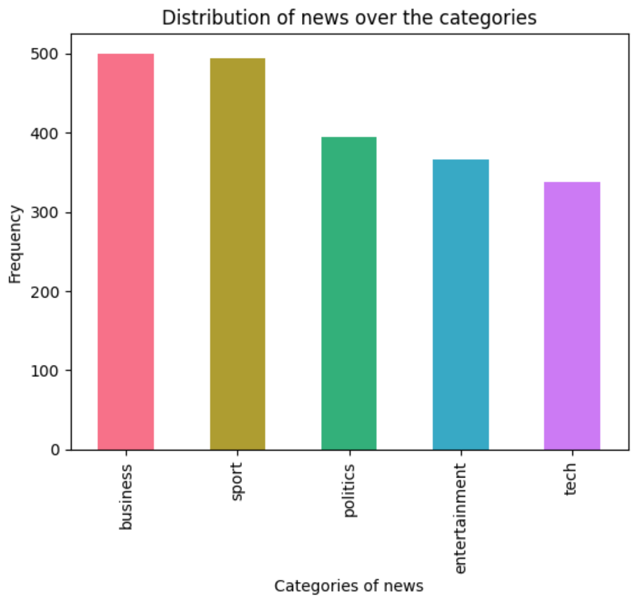

Figura 1: Barplot

3 EDA

The initial phase of any project pertaining to data involves the exploration of data. One of the initial challenges addressed was the issue related to imbalanced classes. As illustrated in the barplot 1, the categories under analysis are relatively balanced. Among these categories, the 'Business' category exhibits the highest number of observations, amounting to 500, while the 'Tech' category presents the lowest number, with 337 observations. Given the observed balanced distribution of the classes within the dataset, it is concluded that the implementation of techniques specifically designed to address class imbalance is unnecessary for subsequent stages of analysis.

A detailed examination was conducted on the distribution of word lengths across various news themes. The focus was on the variations in the length of news articles by category. As demonstrated in the boxplot, the 'Tech' and 'Politics' categories exhibit significant variability in article lengths, whereas the 'Entertainment' category has the shortest median article length.

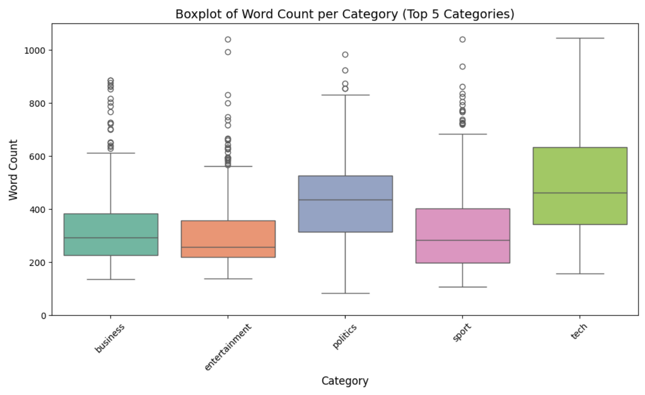

Figura 2: Boxplot

In all the categories analyzed, there are articles with atypical lengths, some of which exceed 1000 words. In the chart examined, the y-axis was fixed to facilitate comparison between the categories.

The report incorporates further visual analysis through the use of a word cloud, which serves as a graphical depiction of word frequencies within the texts evaluated. This tool emphasizes the words most commonly used, rendering them prominently visible to the observer. In this context, the word cloud facilitates the identification of prevailing words across various news categories. Analysis of the word cloud revealed the necessity to revise the list of stopwords, resulting in the exclusion of terms such as "said," "may," and "many." Despite their high frequency, these words do not significantly aid in distinguishing between the different categories. This modification enables a more precise analysis by focusing on more pertinent terms.

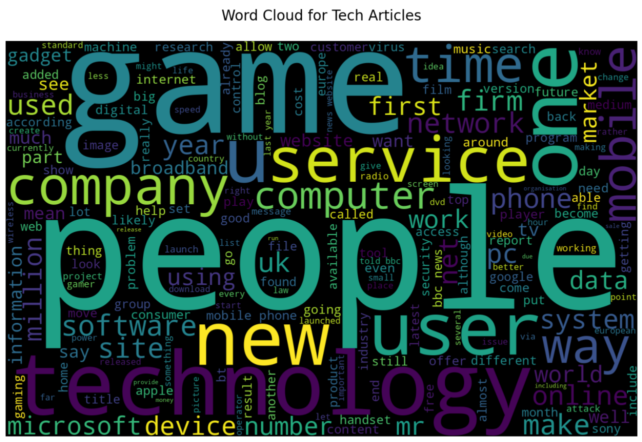

Figura 3: Word cloud tech

In Figure 3, the most frequent words in news articles belonging to the 'Tech' category are depicted. It is not surprising to find terms such as "game," "user," "company," and "computer," which are closely related to the prevailing themes in this domain. These words reflect the central aspects of the technology industry, highlighting the main areas of interest and ongoing discussions in the field.

4 Text Classification

The initial phase of our text mining project focuses on text classification, aiming to develop a predictive model that accurately categorizes news articles. To achieve this, we employ four machine learning algorithms: Logistic Regression (LR), Support Vector Machine (SVM), Random Forest (RF), and Multinomial Naive Bayes (MNB). These models are evaluated using two text representation techniques, namely Count Vectorizer and TF-IDF, with the objective of identifying the most effective classifier and representation method for this task. Additionally, we explore the use of Word2Vec, employing both the Continuous Bag of Words (CBOW) and Skip Gram (SG) models as alternative representation techniques. We also examine the use of a pre-trained model. These approaches are discussed in detail in Section 4.1.

The classification workflow begins by dividing the dataset into a training set and a testing set, essential for assessing the model's performance and its ability to generalize to new, unseen data. The training set comprises 70% of the data and serves as the foundation for model training. In this stage, the model is trained to understand the intricate connections between the input features, which are represented by textual vectors, and their corresponding labels that categorize the articles.

The remaining 30% of the data forms the testing set, which is isolated from the model during training. This set serves as an independent collection of unseen examples, enabling the evaluation of the model's predictions. By analyzing its performance on the testing set, we gain crucial insights into the model's ability to generalize and its efficiency in classifying articles it has not previously encountered. Through this meticulous process, we aim to refine our text classification system, thereby improving its performance.

The next step is converting the textual information into numerical data, where each document is transformed into a numerical vector. More in detail, in this section we employ two distinct text representation techniques:

- **Count-vectorizer**: This is the simplest method to convert text into numerical vectors where each element represents the absolute frequency of a specific word in the document. The length of the vector is determined by the vocabulary size, thereby resulting in sparse representations since most words are absent in a single document.
- **TF-IDF**: This technique consists of two main components:
  - **Term Frequency (TF):** This calculates how often each word appears within a specific article. Words that occur more frequently are given higher numerical values, highlighting their importance within that particular document.
  - **Inverse Document Frequency (IDF):** This measures the relevance of each word throughout the entire data set. Words that are common across many articles receive lower numerical values, indicating their lesser significance in differentiating between articles.

  The overall TF-IDF score for each word in a document is derived by multiplying its TF and IDF values, effectively balancing the word's frequency against its ubiquity across all documents.

Our goal is to determine the most effective model with optimized hyperparameters to significantly enhance the accuracy of predicting the categories of news articles. To achieve this, cross validation with grid search methodology is employed. The former assesses the models' performance and their ability to generalize on new data: by repeatedly training and validating the models on different subsets of the data, we derive reliable estimates of their accuracy and robustness on previously unseen samples. Grid search enhables an exhaustive exploration of possible hyperparameter combinations for each classifier. By systematically testing these combinations, grid search helps identify the optimal parameter values that maximize model performance. After optimization, the test accuracy is used to evaluate the models' real-world predictive performance on unseen data.

We start by evaluating the performance of the Support Vector Classifier using 5-fold cross-validation to determine the most effective kernel function (linear, radial basis function, polynomial, and sigmoid). Next, we apply cross-validation and grid search to optimize the MNB classifier. The primary hyperparameter explored is alpha, the smoothing parameter, with tested values of 2.0, 1.5, and 3.0. In order to optimize the performance of the Random Forest classifier, the considered hyperparameter search space includes tree depths ranging from 2 to 20, minimum samples per leaf ranging from 5 to 20, the number of estimators ranging from 10 to 150, and both gini and entropy criteria. We also extend our analysis to the Logistic Regression classifier by testing various combinations of penalties (l1, l2, and elasticnet) and solvers (newton-cg, lbfgs, saga), including elastic net's l1_ratio.

Tables 1 and 2 summarize the results of hyperparameter optimization and the corresponding performance metrics for each classification model, evaluated using both the Count Vectorizer and TF-IDF Vectorizer, respecitvely. The results show Support Vector Classifier as one of the top-performing model across both vectorization techniques. Using the TF-IDF vectorizer, the SVM achieves the highest cross-validated and test accuracy (0.973 and 0.976, respectively). This highlights the strength of linear kernels in effectively separating classes within the feature space. The Logistic Regression model performs better, particularly when paired with the TF-IDF, achieving a test accuracy of 0.979.

Model	**Best Hyperparameters**	**CV Accuracy**	**Test Accuracy**
SVC	kernel=linear	0.967	0.966
RF	max_depth=20, min_samples_leaf=5, criterion=gini, n_estimators=100	0.954	0.944
LR	penalty=l2, solver=lbfgs	0.964	0.971

Tabella 1: Results of Hyperparameter Optimization and Performance (Count Vectorizer)

Model	**Best Hyperparameters**	**CV Accuracy**	**Test Accuracy**
SVC	kernel=sigmoid	0.973	0.976
RF	max_depth=20, min_samples_leaf=5, criterion=gini, n_estimators=150	0.954	0.952
LR	penalty=None, solver=newton-cg	0.972	0.979

Tabella 2: Results of Hyperparameter Optimization and Performance (TF-IDF Vectorizer)

Overall, while the combination of the SVC and TF-IDF vectorization method provides the best balance of accuracy, efficiency, and robustness for news article classification in this context, it's important to note that the differences in performance between the top models are not substantial.

4.1 Word2Vec

In this section we investigate Word2Vec embeddings, building on our earlier analysis based on Count-vectorizer and TF-IDF representations. Word embeddings are a form of word representation in which words are mapped to vectors in a high-dimensional space. These vectors capture semantic relationships, with words that share similar meanings being located closer together. This enables the model to more accurately represent the underlying meaning of words.

For this project, we concentrate on Word2Vec, a neural network-based model that learns to predict a word based on its surrounding context. This approach is particularly useful for representing documents in the context of our classification task.

Specifically, our study considers three distinct methodologies:

1. **Pre-trained Word Embeddings**: As first approach, we use a pre-trained Word2Vec model, specifically the widely adopted *"word2vec-google-news-300"*, which represents each word as a 300-dimensional numerical vector.

2. **Continuous Bag of Words (CBOW) Architecture**: The second method focuses on the Continuous Bag of Words (CBOW) framework, a variant of Word2Vec that predicts a target word based on its surrounding context. Balancing model complexity with the size of our dataset, we use a 50-dimensional vector representation for each word, which demonstrated strong performance in our experiments.

3. **Skip-Gram Architecture**: Finally we also evaluate the Skip-Gram architecture, another Word2Vec variant which reverses CBOW by predicting context words from a given target word. A 100-dimensional vector is used to represent each word, again offering a compromise between computational efficiency and performance.

We compute the average of the word embeddings in each document to provide document-level embeddings, which successfully capture the document's overall semantic structure. The classifiers use the generated document embeddings as input features.

In order to evaluate the performance of Word2Vec representations, we applied the same machine learning pipeline described in the previous section, which includes hyperparameter tuning using grid search and cross-validation, followed by an evaluation of the models on the test set. Also the hyperparameter grid and evaluation strategy remained consistent with those used for the Count Vectorizer and TF-IDF Vectorizer experiments. The results of the hyperparameter tuning and model evaluation for the Word2Vec-based approaches are summarized in Tables 3, 4 and 5.

Word2Vec-based representations demonstrate competitive performance. Pre-trained embeddings achieve strong results, with SVC and Logistic Regression reaching test accuracies of 0.968 and 0.966, respectively, leveraging rich semantic information. Skip-Gram outperforms CBOW, with SVC achieving 0.966, indicating its strength in capturing nuanced word relationships. CBOW performs adequately but falls short, with SVC achieving only 0.894.

Model	**Best Hyperparameters**	**CV Accuracy**	**Test Accuracy**
SVC	kernel=poly	0.964	0.968
RF	criterion=entropy, max_depth=20, min_samples_leaf=5, n_estimators=150	0.945	0.941
LR	penalty=None, solver=newton-cg	0.963	0.966

Tabella 3: Results of Hyperparameter Optimization and Performance (Pre-trained Word2Vec Embeddings)

Model	**Best Hyperparameters**	**CV Accuracy**	**Test Accuracy**
SVC	kernel=linear	0.894	0.894
RF	criterion=entropy, max_depth=15, min_samples_leaf=5, n_estimators=150	0.894	0.86
LR	penalty=None, solver=newton-cg	0.933	0.933

Tabella 4: Results of Hyperparameter Optimization and Performance (CBOW Representation)

Model	**Best Hyperparameters**	**CV Accuracy**	**Test Accuracy**
SVC	kernel=linear	0.957	0.966
RF	criterion=gini, max_depth=20, min_samples_leaf=5, n_estimators=50	0.961	0.953
LR	penalty=None, solver=newton-cg	0.961	0.961

Tabella 5: Results of Hyperparameter Optimization and Performance (Skip-Gram Representation)

While Word2Vec effectively captures semantic relationships between words, it is the combination of TF-IDF with Logistic Regression (LR) that delivers the highest performance, achieving an impressive accuracy of 0.979. This demonstrates TF-IDF's ability to extract highly discriminative class-specific features for classification tasks.

Despite that the Support Vector Classifier using pre-trained Word2Vec stands out as a good choice for balancing efficiency and effectiveness. Infact, using a TF-IDF representation in a deployment scenario could lead to quicker model degradation due to well-known issues associated with this technique. Specifically, the challenge of handling new words can necessitate frequent updates to the vocabulary, leading to an increase in the vector's dimensionality. This, in turn, raises computational costs for both training the model and making predictions. See 6. In conclusion, although we have a preference for pretrained embeddings, the choice should be based on an evaluation of the deployment project's priorities, including efficiency needs, resource availability, transparency requirements, and the complexity of textual relationships.

Metric	**Pretrained (s)**	**TF-IDF (s)**
Training Time	0.095	5.572
Classification Time	0.034	1.007
Total Time	0.129	6.579

Tabella 6: Comparison of Execution Times for Pretrained and TF-IDF Models

5 Text Summarization

Two approaches were implemented for the Text Summarisation task: extrcative and abstractive. The extractive approach is based on identifying and selecting the most relevant sentences directly from the main text without editing them. This approach is relatively simple to implement and guarantees text consistency and the absence of grammatical errors as it uses already existing sentences. However, it may be less flexible and may not always be able to capture the overall meaning of the text, especially when important information is distributed in several places. The abstractive approach, on the other hand, aims to understand the content of the text and generate new summaries using its own language. This approach is more complex, requires advanced models such as those based on neural networks, and may be more prone to grammatical or coherence errors. It offers the advantage of being more creative and condensing information better, allowing the content to be reformulated in a more concise and meaningful way. However, it is a more computationally intensive method and requires a longer runtime on average. The extractive approach used in the project was implemented using the Latent Semantic Analysis (LSA) model, and consists of the following steps :

- The text is processed using the PlaintextParser parser, whose tokenizer is set to English.
- The LSA technique synthesises the content by identifying a predefined number of sentences, in our case 5.
- The sentences are then concatenated and the final summary is produced.

For both approaches, the evaluation metrics ROUGE-1, ROUGE-2 and ROUGE-L were calculated by comparing the generated summaries with the original titles. Although comparison with titles is not an optimal choice for assessing the quality of summaries, in the absence of abstracts this solution was opted for. Indeed, it is important to note that titles, being extremely short and concise, may not be an ideal reference for assessing summaries, especially abstractive ones, which tend to reformulate the content in a more creative and generalised manner. Therefore, while the extractive approach might seem more effective in capturing key elements of the title, the abstractive approach might offer qualitative advantages that do not fully emerge from the ROUGE metrics. These metrics allow the quality of the abstract to be measured in terms of similarity with the reference by assessing the correspondence of uni-grams, bi-grams and the overlap of longer sequences between texts, respectively.

In the abstractive approach, the pre-trained deep learning model we chose was the BART (Bidirectional and Auto-Regressive Transformers) developed by Facebook.

Techniques	**Metrics**	**Mean**	**Standard Deviation**
Extractive	ROUGE-1	0.055	0.026
	ROUGE-2	0.008	0.014
	ROUGE-L	0.046	0.022
Abstractive	ROUGE-1	0.090	0.045
	ROUGE-2	0.013	0.023
	ROUGE-L	0.076	0.039

Tabella 7: Metrics results for extractive and abstractive approach

The results show that although the abstractive approach achieves higher average values of ROUGE-1 and ROUGE-L than the extractive approach, this does not necessarily imply a clear superiority in capturing the key elements of the securities. ROUGE-2, in particular, turns out to be the least significant in both approaches, probably because short titles do not contain enough bi-grams for a robust evaluation. This explains the low and similar values for ROUGE-2 in both approaches. Finally, ROUGE-L is like a more indicative metric for evaluating headlines, as it takes into account both content and word order, which is crucial for representing the coherence and effectiveness of the message. Despite the higher average scores for the abstractive approach, it is important to consider that it introduces more variability (higher standard deviation), which may reflect a lower reliability than the extractive approach.

6 Conclusion

In conclusion, our exploration of the BBC news dataset has been characterized by the successful use of a variety of powerful tools and techniques. Through meticulous preprocessing, we enhanced the quality of our textual data, ensuring it provided a strong foundation for further analysis.

To recapitulate the conclusion gathered in the text classification task, we have explored a variety of text representation techniques and machine learning approaches. We are greatly satisfied with the results, having observed similar outcomes across both simpler and more complex techniques. Despite a dedicated section on Word2Vec, capable of capturing the semantic meaning of words, this approach did not yield a performance increase over TF-IDF. However, the

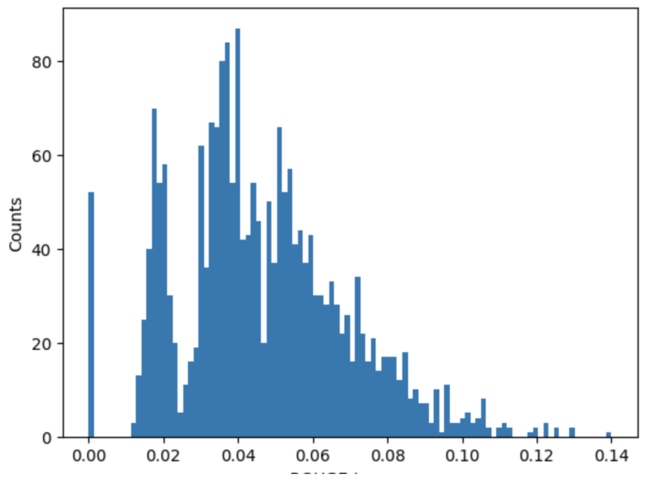

Figura 4: Rouge-L extractive

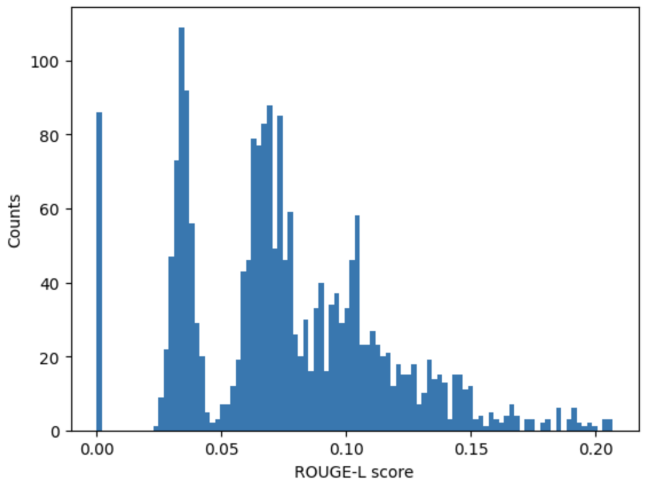

Figura 5: Rouge-L abstractive

latter's requirement to frequently update the vocabulary due to new words can lead to increased dimensionality and higher computational costs in deployment scenarios.

For future analyses, it would be practical to conduct a thorough comparison of model performance during testing phases, particularly focusing on how each model degrades over time. This degradation analysis would help in understanding the long-term viability of the models, especially the TF-IDF approach, in real-world applications. This could lead to insights on maintaining model accuracy and efficiency as data evolves.

With references on the analyses conducted on text summarization shows that the abstractive approach achieves higher ROGUE scores on average than the extractive, although with greater variability results. However the specific context of titles, which are characterized by extreme conciseness, may limit the ability of ROUGE metrics to fully capture the quality of the summary. In addition, it should be considered that the abstractive approach incurs a significantly higher computational cost than the extractive, as it requires advanced deep learning models, often based on transformative neural networks, with longer inference times and higher consumption of resources. To improve the accuracy of the evaluation, a possible future development could be to use more extended abstracts instead of titles, allowing for a richer and more detailed comparison. In addition, it might be useful to explore query-based summary models, which generate customized summaries based on specific aspects of interest, improving the relevance of the extracted content. A further step could be the integration of more advanced metrics, such as BERTScore or other semantic assessments, for more accurate measurement of summary quality. Finally, optimization of the models used, such as by fine-tuning on specific datasets, could further improve the consistency and fidelity of the generated summaries, better balancing quality and computational cost.

Riferimenti bibliografici

[1] Dadgar, Seyyed Mohammad Hossein, Araghi, Mohammad Shirzad, and Farahani, Morteza Mastery. A novel text mining approach based on TF-IDF and support vector machine for news classification. In *2016 IEEE International Conference on Engineering and Technology (ICETECH)*, pages 112–116, 2016. IEEE.

[2] Dogru, Hasibe Busra, Tilki, Sahra, Jamil, Akhtar, and Hameed, Alaa Ali. Deep learning-based classification of news texts using Doc2Vec model. In *2021 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA)*, pages 91–96, 2021. IEEE.

[3] Gaurav, Prateek. NLP: Zero to Hero [Part 1: Introduction, BOW, TF-IDF Word2Vec]. *Medium*, 2022.

[4] Karani, Dhruvil. Introduction to word embedding and word2vec. *Towards Data Science*, volume 1, 2018.

[5] Kurnia, Rafly, Tangkuman, Y., and Girsang, A. Classification of user comment using Word2Vec and SVM classifier. *International Journal of Advanced Trends in Computer Science and Engineering*, volume 9, number 1, pages 643–648, 2020.

[6] Dadgar, Seyyed Mohammad Hossein, Araghi, Mohammad Shirzad, and Farahani, Morteza Mastery. A novel text mining approach based on TF-IDF

and support vector machine for news classification. In *2016 IEEE International Conference on Engineering and Technology (ICETECH)*, pages 112–116, 2016. IEEE.

[7] Dogru, Hasibe Busra, Tilki, Sahra, Jamil, Akhtar, and Hameed, Alaa Ali. Deep learning-based classification of news texts using Doc2Vec model. In *2021 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA)*, pages 91–96, 2021. IEEE.

[8] Gaurav, Prateek. NLP: Zero to Hero [Part 1: Introduction, BOW, TF-IDF Word2Vec]. *Medium*, 2022.

[9] Karani, Dhruvil. Introduction to word embedding and word2vec. *Towards Data Science*, volume 1, 2018.

[10] Kurnia, Rafly, Tangkuman, Y., and Girsang, A. Classification of user comment using Word2Vec and SVM classifier. *International Journal of Advanced Trends in Computer Science and Engineering*, volume 9, number 1, pages 643–648, 2020.

[11] Nenkova, Ani, and McKeown, Kathleen. Automatic summarization. *Foundations and Trends in Information Retrieval*, volume 5, numbers 2–3, pages 103–233, 2011.

[12] Gambhir, Mahak, and Gupta, Vishal. Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*, volume 47, number 1, pages 1–66, 2017.

[13] Gong, Yihong, and Liu, Xin. Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 19–25, 2001.

[14] Steinberger, Josef, and Ježek, Karel. Using Latent Semantic Analysis in Text Summarization and Summary Evaluation. In *Proceedings of ISIM*, 2009.

[15] See, Abigail, Liu, Peter J., and Manning, Christopher D. Get To The Point: Summarization with Pointer-Generator Networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.

[16] Lewis, Mike, Liu, Yinhan, Goyal, Naman, Ghazvininejad, Marjan, Mohamed, Abdelrahman, Levy, Omer, Stoyanov, Veselin, and Zettlemoyer, Luke. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of ACL*, 2020.

[17] Lin, Chin-Yew. ROUGE: A Package for Automatic Evaluation of Summaries. In *Proceedings of Workshop on Text Summarization Branches Out*, 2004.