VISA: Journal of Visions and Ideas
Vol 4 No 3 (2024) 1359 – 1371 E-ISSN *2809-2058* P-ISSN 2809-2643
DOI: 47467/visa.v4i3.2835

Logistic Regression Classification with TF-IDF and FastText for Sentiment Analysis of LinkedIn Reviews

Nabila Sya'bani Wardana[1], Firza Prima Aditiawan[2]*, Anggraini Puspita Sari[3],
[1,2,3]Informatika, Fakultas Ilmu Komputer, UPN Veteran Jawa Timur, Indonesia
Rungkut Madya No.1, Gunung Anyar, Surabaya, Indonesia
[1]20081010114@student.upnjatim.ac.id, [2*]firzaprima.if@upnjatim.ac.id,
[3]anggraini.puspita.if@upnjatim.ac.id

ABSTRACT

Social media and professional networking platforms like LinkedIn have become crucial platforms for individuals to interact, share information, and build professional networks. Despite the significant benefits LinkedIn has provided to its users, there are still some limitations such as account restriction ambiguity, synchronization issues, and the emergence of spam and irrelevant content. Therefore, it is important to understand users' responses to the application. Previous research has shown that sentiment analysis can be an effective tool in understanding user reviews of applications. This study will continue previous research by analyzing the sentiment of user reviews of the LinkedIn application using the Logistic Regression method, taking into account the use of TF-IDF Feature Extraction and FastText Feature Expansion. Logistic Regression was chosen because it is effective in handling binary sentiment classification problems and has relatively high training speed. This method will be tested to address data imbalance and improve classification performance. This research demonstrates that this approach can provide optimal results in measuring accuracy, recall, precision, and F-Score. The research findings will provide valuable insights for LinkedIn application developers to enhance service quality. Based on the evaluation metrics, it can be observed that the first testing scheme with default parameters achieved an accuracy of 91.86%, a precision of 94.05%, a recall of 91.99%, and an F1-Score of 93.01%. The percentage values obtained already surpass 90%.

Keywords: *Expert System, Classification, Parenting, Early Childhood, Forward Chaining.*

ABSTRAK

Media sosial dan platform jaringan profesional seperti LinkedIn telah menjadi platform penting bagi individu untuk berinteraksi, berbagi informasi, dan membangun jaringan profesional. Terlepas dari manfaat signifikan yang diberikan LinkedIn kepada penggunanya, masih ada beberapa keterbatasan seperti ambiguitas pembatasan akun, masalah sinkronisasi, dan munculnya spam dan konten yang tidak relevan. Oleh karena itu, penting untuk memahami respons pengguna terhadap aplikasi tersebut. Penelitian sebelumnya telah menunjukkan bahwa analisis sentimen dapat menjadi alat yang efektif dalam memahami ulasan pengguna terhadap aplikasi. Penelitian ini akan melanjutkan penelitian sebelumnya dengan menganalisis sentimen ulasan pengguna terhadap aplikasi LinkedIn menggunakan metode Regresi Logistik, dengan mempertimbangkan penggunaan Ekstraksi Fitur TF-IDF dan Ekspansi Fitur FastText. Regresi Logistik dipilih karena efektif dalam menangani masalah klasifikasi sentimen biner dan memiliki kecepatan *training* yang relatif tinggi. Metode ini akan diuji untuk mengatasi ketidakseimbangan data dan meningkatkan kinerja klasifikasi. Penelitian ini menunjukkan bahwa pendekatan ini dapat memberikan hasil yang optimal dalam mengukur akurasi, *recall*, presisi, dan F-Score. Temuan

Vol 4 No 3 (2024) 1359 – 1371 E-ISSN *2809-2058* P-ISSN 2809-2643
DOI: 47467/visa.v4i3.2835

penelitian ini akan memberikan wawasan yang berharga bagi para pengembang aplikasi LinkedIn untuk meningkatkan kualitas layanan. Berdasarkan metrik evaluasi, dapat diamati bahwa skema pengujian pertama dengan parameter *default* mencapai akurasi sebesar 91,86%, presisi sebesar 94,05%, recall sebesar 91,99%, dan F1-Score sebesar 93,01%. Nilai persentase yang diperoleh sudah melampaui 90%.

Kata kunci: Sistem Pakar, Klasifikasi, Pengasuhan Anak, Anak Usia Dini, *Forward Chaining*.

INTRODUCTION

In the continuously evolving era of digitalization, social media and online professional networking have become integral parts of daily life. One prominent platform in professional networking is LinkedIn, which aims to facilitate information exchange, professional communication, and network building. With over 774 million members worldwide, LinkedIn has become a primary platform for individuals to promote themselves, seek career opportunities, and expand their professional networks [1]. However, alongside its growth and popularity, LinkedIn also faces several challenges, including ambiguity in account restrictions, synchronization issues, and the presence of spam and irrelevant content. One way to understand user responses to platforms like LinkedIn is through sentiment analysis of the reviews they provide. Sentiment analysis is a technique used to understand the opinions or feelings contained within text, whether positive, negative, or neutral [2]. By analyzing user reviews, developers and service providers can gain valuable insights into the strengths and weaknesses of their applications, as well as improve the overall user experience. Currently, text processing with machines has advanced rapidly through the use of machine learning and deep learning. One popular example of widely used deep learning is Recurrent Neural Networks (RNN), which excel at recognizing complex sequential patterns from gradient-based [3]. On the other hand, logistic regression is a machine learning method often used for simple text classification with high efficiency. Both logistic regression and RNNs rely on gradient-based optimization methods to train their models. Previous research has shown that the Logistic Regression method, which is an effective classification algorithm for binary sentiment classification problems, can be used to accurately analyze the sentiment of user reviews [4]. Furthermore, the use of TF-IDF Feature Extraction and FastText Feature Expansion techniques has also been proven effective in improving the performance of sentiment classification models [4]. TF-IDF (Term Frequency-Inverse Document Frequency) is a method for extracting features from text commonly used in text analysis, while FastText is a text representation learning method that takes into account the sub-word structure of words. Both techniques can provide additional insights into the sentiment contained within LinkedIn user reviews. Therefore, in this study, we aim to combine the Logistic Regression method with TF-IDF Feature Extraction and FastText Feature Expansion to analyze the sentiment of user reviews of the LinkedIn application. We will test this approach to understand user opinions about this platform and provide recommendations that can help developers improve the quality of their services. By conducting this research, we hope to contribute to the

VISA: Journal of Visions and Ideas
Vol 4 No 3 (2024) 1359 – 1371 E-ISSN *2809-2058* P-ISSN 2809-2643
DOI: 47467/visa.v4i3.2835

development of more responsive application systems that meet the needs and desires of users.

METHODS

The methodology for this study involves several key steps to design and implement the sentiment analysis system effectively. The workflow encompasses data scraping, sorting, and labeling, followed by text preprocessing, TF-IDF feature extraction, and FastText word embedding. Subsequently, the data is split into training and testing sets, and classification is performed using a logistic regression model tested with the testing data. Testing is conducted with predefined scenarios based on parameters, and the results are evaluated using a confusion matrix to assess model performance. Parameters such as accuracy, precision, recall, and F1-score are used to measure overall sentiment analysis performance. The design of methodology can be seen in

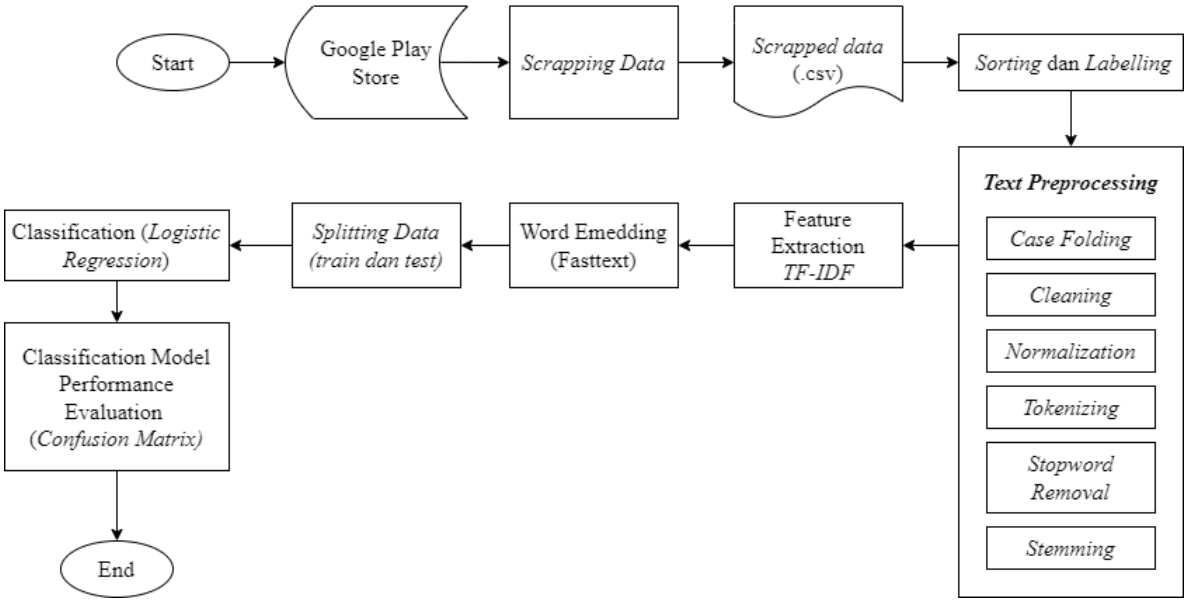

Figure 1. Design Method

In user reviews of the LinkedIn mobile application sourced from the Google Play Store platform serve as the primary data. Employing web scraping techniques facilitated by the 'google_play_scrapper' package [9], the data extraction process retrieves these reviews, which are subsequently stored in CSV format. Following extraction, a meticulous sorting and labeling procedure is conducted to categorize the reviews into positive and negative sentiments, ensuring the data's preparedness for subsequent analysis. The core methodology of this research involves the utilization of Term Frequency-Inverse Document Frequency (TF-IDF) for feature extraction. Through TF-IDF, the textual user reviews undergo transformation into numerical representations, wherein the significance of each word is quantified based on its frequency across documents. This process comprises four distinct stages, ensuring a comprehensive and nuanced understanding of the corpus described in Figure 2.

VISA: Journal of Visions and Ideas
Vol 4 No 3 (2024) 1359 – 1371 E-ISSN *2809-2058* P-ISSN 2809-2643
DOI: 47467/visa.v4i3.2835

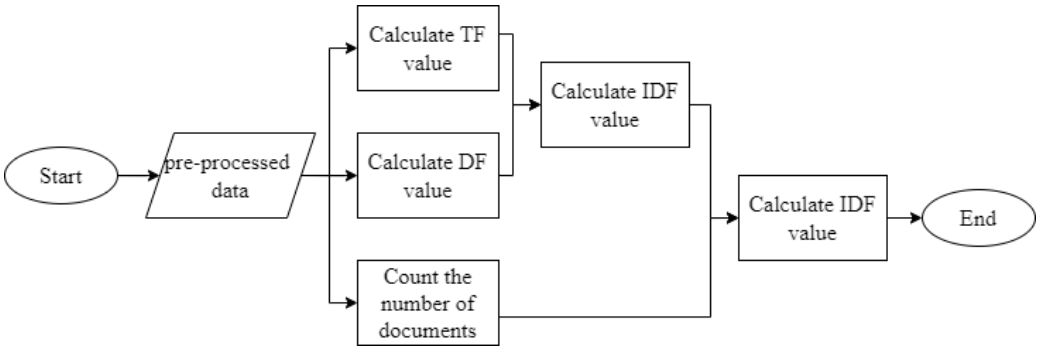

Figure 2. TF-IDF Methodology

To further enhance the representation of word vectors, TF-IDF representations are integrated with FastText word embeddings. This integration process begins with the formation of n-grams from each document at the sub-sentence level [10]. By capturing sentiment contexts at a granular level, this approach aims to enrich the subsequent analyses with deeper insights into user sentiment. Upon completing the feature extraction and integration steps, the data is split into training and testing sets, allowing for the subsequent phases of model training and evaluation. The FastText model is trained using the processed data to learn vector representations for each word and sub-word within the sentiment context. Following training, FastText generates word embedding vectors, which are seamlessly integrated with TF-IDF vectors to create comprehensive representations of sentiment.

In this sentiment analysis of LinkedIn user reviews, the research uses a combination of TF-IDF and FastText for feature extraction, with all stages of the research process utilizing default parameter settings as defined during model creation. This approach aims to provide insights into the efficacy of Logistic Regression classification models when supported by these combined vector feature representations. The objective is to determine whether this combination can enhance the model's performance in detecting sentiment in LinkedIn reviews compared to using individual vector representation methods. After the vector representations are combined, the Logistic Regression algorithm is applied for classification. Logistic Regression is chosen for its simplicity and effectiveness in binary classification tasks [11]. It estimates the probability that a given input belongs to a certain class, making it suitable for sentiment analysis where the goal is to classify reviews as positive or negative. The classification results are compared against actual labels, with a confusion matrix aiding in visualizing the classification outcomes [12]. Various evaluation metrics, including accuracy, precision, recall, and F1-score, are computed to provide a comprehensive evaluation of the model's effectiveness in detecting sentiment within LinkedIn reviews. Through meticulous testing and evaluation, this research aims to elucidate whether the combined use of TF-IDF and FastText can enhance the model's sentiment analysis capabilities compared to using TF-IDF alone.

VISA: Journal of Visions and Ideas
Vol 4 No 3 (2024) 1359 – 1371 E-ISSN *2809-2058* P-ISSN 2809-2643
DOI: 47467/visa.v4i3.2835

RESULT AND DISCUSSIONS

This chapter elucidates the discussion of the research conducted in accordance with the methodology stages previously outlined. The discussion encompasses the implementation of each methodological step, starting from data extraction to model evaluation.

A. Data Acquisition

The data acquisition stage involves retrieving a dataset comprising application reviews from the Google Play Store platform, specifically targeting the LinkedIn application. This process is facilitated using the `google-play-scraper` library available in Python. Initially, the application ID, located at the end of the URL following the parameter "?id=", is obtained. The objective is to acquire review data from the LinkedIn application available on the Google Play Store using the `google-play-scraper` library in Python. The initial step involves installing the `google-play-scrapper` package, followed by importing necessary libraries and modules such as `google_play_scraper`, `pandas`, and `numpy`. Subsequently, the program invokes the `reviews` function to fetch reviews from the LinkedIn application with the specified ID `'com.linkedin.android'`, setting the language (`lang`) and country (`country`) parameters to Indonesian. Reviews are sorted by highest relevance (`Sort.MOST_RELEVANT`) and retrieved up to 10,500 reviews (`count=10500`). The obtained review data is then converted into a DataFrame using `pandas`, retaining only three columns: `'score'`, representing the review rating; `'at'`, indicating the timestamp of the review's creation; and `'content'`, containing the actual review comments. The DataFrame is subsequently sorted based on the review date (`'at'`) in descending order, starting from the most recent reviews. Finally, the DataFrame is exported to a CSV file named "skripsi_scrappedData_LinkedIn_10500.csv". The result of data acquisition can be seen at Figure 3.

Data Scrapping Linkedin : 10500 Data

	score	at	content
0	5	2024-01-09 16:42:45	like
1	1	2024-01-09 16:19:56	Verifikasi paan cok pilih yg menghadap ke atas...
2	1	2024-01-09 12:31:24	Saya disuruh verifikasi keamanan dan sudah dil...
3	1	2024-01-09 12:18:06	Kenapa verifikasi manusia nya ngulang terus? G...
4	5	2024-01-09 11:17:09	Good

	score	at	content
10495	5	2018-09-13 11:31:27	Baru nyoba jadi belum tahu
10496	5	2018-09-13 07:42:58	Satu kata; Bagus. Satu kalimat; Profesional ha...
10497	5	2018-09-13 02:07:45	Semoga bisa membantu
10498	4	2018-09-12 17:28:26	Penasaran ama aplikasi ini,buat apa fungsinya
10499	5	2018-09-12 08:28:48	Sangat membantu

VISA: Journal of Visions and Ideas
Vol 4 No 3 (2024) 1359 – 1371 E-ISSN *2809-2058* P-ISSN 2809-2643
DOI: 47467/visa.v4i3.2835

Figure 3. Data Scrapping Result

B. Data Preprocessing

The labeling process involves determining two types of sentiments: reviews indicating positive sentiment and those indicating negative sentiment. Label assignment is conducted manually by interpreting reviews based on their negative or positive sentiments. According to the Kamus Besar Bahasa Indonesia (KBBI), negative sentiment refers to reactions or attitudes that lower the value of someone or something, while positive sentiment refers to reactions or attitudes that enhance the value of someone or something [13]. Therefore, data will be labeled as negative with a value of 0 if the reviews contain critical or dissatisfied words such as 'poor', 'unsatisfactory', 'difficult', and other expressions indicating dissatisfaction with the application. Conversely, data will be labeled as positive with a value of 1 if the reviews contain words of praise such as 'like', 'good', 'helpful', and other expressions indicating satisfaction with the application. Sentiment annotations for each review are added to a new column named 'label', labeling the data that can be seen at Figure 4.

label	score	at	content
1	5	2024-01-09 16:42:45	like
0	1	2024-01-09 16:19:56	Verifikasi paan cok pilih yg menghadap ke atas...
1	5	2024-01-09 11:17:09	Good
1	5	2024-01-09 11:09:28	Aplikasi Profesional paling relevan buat kaum ...
1	5	2024-01-09 07:31:05	Bagus banget dan canrik

Figure 4. Data Preprocessing

The data that has undergone preprocessing will be further analyzed by conducting a word analysis using a word cloud to visually represent the frequency of words contained in the processed dataset from previous stages. By utilizing a word cloud, analysis can be conducted to provide a general overview of whether there are still inappropriate words or noise that may have been overlooked in the cleaning stage and could potentially interfere with the subsequent word processing by the machine [14]. Based on the representation of the word cloud, words with a high frequency of occurrence will be displayed in a larger size. The more frequently a word appears in the processed dataset, the larger its representation will be in the word cloud, and vice versa. For the LinkedIn application review data, visually, there are several frequently occurring words, including 'aplikasi', 'saya', 'tidak', 'bagus', 'verifikasi', 'sangat', 'bantu', and 'LinkedIn'. Based on the word cloud, it can also be observed that there are no longer unnecessary words such as slang or misspelled words, as they have undergone normalization during the preprocessing stage.

VISA: Journal of Visions and Ideas
Vol 4 No 3 (2024) 1359 – 1371 E-ISSN *2809-2058* P-ISSN 2809-2643
DOI: 47467/visa.v4i3.2835

Figure 5. Frequency of Review Words

Next, the process involves computing TF-IDF by creating a function called 'compute_tfidf' with the input parameter 'docs', which represents a collection of documents. The 'compute_tfidf' function consists of several lines of code. Firstly, an empty list is initialized to store the TF-IDF values of each document, and these values are stored in the variable 'tfidf'. The next step involves calculating the IDF (Inverse Document Frequency) values for the document collection using the function 'compute_idf(docs)' that was previously defined. Then, an iteration is performed over each document ('doc') in the document collection ('docs'). Within each iteration, the TF (Term Frequency) value for each document ('doc') is calculated using the function 'compute_tf()'. After computing the TF value, TF-IDF is calculated for each word in the document by multiplying the TF value with the IDF value stored in each respective dictionary. The computed TF-IDF values for all words in each document are stored in the variable 'tfidf_doc' and then appended to the empty list 'tfidf' defined earlier. Finally, the 'compute_tfidf' function returns the list 'tfidf'.

```
TF-IDF features for document 15:
        'susah': 0.1597495779798887
        'bgt': 0.2485362072840649
        'buat': 0.14260708700807 76
        'login': 0.14747639361010131
        'sama': 0.20152437284446817
        'proses': 0.2871990224878113
        'verifikasi': 0.12702614239249702
        'lot': 0.3102645758118793

TF-IDF features for document 16:
        'saya': 0.08083911595891449
        'suka': 0.16711728427150832
        'aplikasi': 0.06223074749419919
        'nya': 0.11660228643951774
        'mudah': 0.1334167626925171
        'di': 0.09329690366154839
        'erti': 0.24338119269204794
        'dan': 0.0785708818923616
        'lain': 0.3451587587214127
```

Figure 6. TF-IDF Result

VISA: Journal of Visions and Ideas
Vol 4 No 3 (2024) 1359 – 1371 E-ISSN *2809-2058* P-ISSN 2809-2643
DOI: 47467/visa.v4i3.2835

The data used to train the FastText model is the data in the 'hasil_Stemming' column of the pre_data DataFrame. Each string in the data column needs to be converted into a list of words to be processed by the model. String separation is done using the 'split' function based on commas and spaces (", ") and removing the "[" and "]" characters, which are separators between documents, using the 'strip()' function, and also removing the single quote "'". After converting all data into a list of words, the data is stored in the 'sentences' variable, which will be used as input parameters for the data to be trained on the FastText model. Training is done by calling the 'Fasttext()' function provided by Gensim. The trained FastText model is stored in the 'model_Fasttext' variable and saved in a file format named 'fasttext_model.model'.

```
model_fasttext.wv['verifikasi']

array([ 0.39678225, -0.34526864, -0.05295744,  1.3134226 ,  0.4351108 ,
       -0.30916813, -0.36871606,  1.0538807 ,  0.49265906, -0.6403095 ,
       -1.1315659 ,  0.13436782,  0.539006  , -0.39049825, -0.09255583,
        0.27613544, -0.47757083,  0.13179924, -1.3440266 , -0.30507803,
       -1.2877127 , -0.84163064,  0.6039313 , -0.08688612,  1.0492644 ,
       -0.08902604,  0.32746223,  0.83947426, -1.0534353 , -1.2193087 ,
       -0.93667287, -0.21410228,  0.13074577, -0.50468266,  0.42763713,
        0.26333216, -0.13200939, -0.65905595, -1.5392004 ,  1.0604261 ,
        0.9607566 ,  0.3646379 ,  0.21109754, -1.706075  , -0.82524717,
       -0.6095488 , -0.32596952,  0.8976899 ,  1.3065988 , -0.39141268,
        0.39420527, -1.0218469 , -1.7247758 , -0.5987452 , -0.48618218,
       -0.21731041,  0.01224182,  0.46165228,  0.5735539 , -0.5896704 ,
       -1.5519055 , -0.8034898 ,  1.0444779 , -0.62873566, -0.7745226 ,
        1.1267651 ,  0.0103724 ,  0.13525285, -0.18034492, -1.3361412 ,
       -0.21177769, -2.015471  , -0.86429816, -0.20878589, -0.5936336 ,
       -0.4888741 ,  0.34948936, -0.06436379,  0.42847568, -0.5838738 ,
       -0.18056798, -0.32894808, -0.1306629 , -0.2696216 , -0.33069596,
        0.06094408,  0.2627477 ,  0.6369771 , -0.18374673,  1.1992452 ,
        0.00531885,  0.6148113 ,  0.09985705, -0.32508656, -0.5639888 ,
       -0.98154515,  0.17347042,  1.2124621 , -0.2032139 , -0.33005068],
      dtype=float32)
```

Figure 7. FastText Model Results

The result of the model training is the vector representation of the words learned from the training dataset. In this case, each word vector will be represented with a vector dimension of 100, according to the default value of the vector size parameter. Besides being able to view the word vector of a specific word, the FastText model can also represent vectors of several words that are close or similar to the searched word, with higher values indicating greater similarity between the displayed word vectors. The vector representations of similar words indicate that the FastText model can learn the semantics of words quite well. The quality of the trained model's results is also influenced by the parameters set to control the training process. For now, training is done with default parameters. The vector representations generated by the FastText model from this training are expected to help improve the machine's performance in processing data for further stages of processing. Once the data has been processed and represented in vector form, it can be used as input for the classification model. Before processing the data, the classification model needs to be created and defined to be trained using the available data. Through training, the model will learn and evaluate its performance on the input data. The classification model used in this study employs the Logistic Regression algorithm.

Vol 4 No 3 (2024) 1359 – 1371 E-ISSN *2809-2058* P-ISSN 2809-2643
DOI: 47467/visa.v4i3.2835

a. Combining vector representation

The process of combining word vector representations from TF-IDF calculations with word vectors from FastText involves merging both vectors into a single representation to enrich the document information and improve model performance [15]. First, an empty list named 'combined_features' is created to store the combined feature vectors. An iteration is performed over the index of each document in the list of words from all documents, 'sentences'. In each iteration, the TF-IDF vector for the i-th document is retrieved from the 'tfidf_df' DataFrame and stored in the 'tfidf_vector' variable. Additionally, the average FastText vector for each word in the document is calculated using the word vectors generated by 'model_fasttext' and stored in the 'fasttext_vector' variable. The two vector representations are then concatenated into a single combined feature vector using the 'np.concatenate()' function and saved into the 'combined_vector' variable. The combined vectors from all documents are added to the 'combined_features' list, which is then converted into a DataFrame, 'combined_features_df', with rows corresponding to the number of documents and columns to the combined features. The combined feature vectors are now ready for classification using machine learning algorithms.

b. Logistic Regression

The logistic regression classification model is defined as a class named 'LogisticRegression', which contains several functions. The first function is to create the constructor of the 'LogisticRegression' class with '__init__', which has three parameters: 'learning_rate' (the learning rate for updating model weights), 'num_iterations' (the number of training iterations for the model), and 'verbose' (to print the training process of the model). These three parameters are optional inputs, and the current default values will be used when the model is utilized. This function is useful for determining and adjusting the model for the classification process based on the initialization of the attributes within the object. The 'fit' function is defined for the model to perform the learning process within the 'LogisticRegression' class by adjusting the model weights based on the training data. The learning process initializes the initial weights with zero values and iterates as many times as the specified 'num_iterations'. In each iteration, several calculations are performed, including predicting using the sigmoid function, calculating the error value, updating the weights, and displaying the training progress at each iteration. After the weight calculations, the logistic regression model, which has been trained with the data, can be used to make predictions. The 'predict' function is defined by inputting the logistic regression equation, whose calculations also use the sigmoid function. The prediction results are rounded to binary values (0 or 1) according to the label categories of the data. Additionally, the loss value is calculated to measure how well the model performs based on the predicted values compared to the actual values. The

VISA: Journal of Visions and Ideas
Vol 4 No 3 (2024) 1359 – 1371 E-ISSN *2809-2058* P-ISSN 2809-2643
DOI: 47467/visa.v4i3.2835

last function calculates the model's accuracy to measure the accuracy level of the model in predicting new data (other than the data trained by the model). This calculation is done by comparing the model's predictions with the actual labels and computing the average percentage of correct predictions.

The data splitting will be performed on two types of data: the application review data already transformed into combined feature vector representations as the 'combined_features_df' attribute and the data in the label column of the 'pre_data' dataframe as the 'labels' target, representing two sentiment class types (positive: 1 and negative: 0). The data splitting is done using the 'train_test_split' function from 'scikit-learn' with a 'test_size' value of 0.3, meaning the data split ratio is 70:30 with 70% training data and 30% testing data. The 'random_state' parameter is set to 24 to ensure consistent dataset splitting every time the program is rerun. The data splitting process will result in four types of datasets: 'X_train' as the feature data for training, 'X_test' as the feature data for testing, 'y_train' as the class label data corresponding to 'X_train', and 'y_test' as the class label data corresponding to 'X_test'. With a data split ratio of 70:30, the training data size will be 3611, and the testing data size will be 1548. The next step is to use the created classification model to be trained with the training data, and after the model finishes learning from the data, testing can be performed by predicting labels with the testing data. The prediction results are then used for calculating the accuracy performance of the model in processing the data. The training of the model is done by creating a model object and calling the 'LogisticRegression()' class previously created. The 'verbose' parameter is set to True to print the training progress of the model. The model is trained by calling the 'fit' method with the training data ('X_train' and 'y_train'). After training, the model is used to predict the labels of the test data 'X_test' using the 'predict' method. The final step is to measure the performance of the model with the actual test data 'y_test' using the 'accuracy' method until 300 iteration.

```
Accuracy: 0.9186046511627907
Precision: 0.9405162738496072
Recall: 0.9198682766190999
F1 Score: 0.93007769145394
```

Figure 8. Logistic Regression Model Accuracy Result

Based on the evaluation metrics, it can be observed that the first testing scheme with default parameters achieved an accuracy of 91.86%, a precision of 94.05%, a recall of 91.99%, and an F1-Score of 93.01%. The percentage values obtained already surpass 90%.

CONCLUSIONS AND RECOMMENDATION

Vol 4 No 3 (2024) 1359 – 1371 E-ISSN *2809-2058* P-ISSN 2809-2643
DOI: 47467/visa.v4i3.2835

In conclusion, the study successfully applied a combination of methodologies to analyze sentiment in user reviews of the LinkedIn application. Initially, data acquisition involved scraping user reviews from the Google Play Store using the `google-play-scraper` package. These reviews were then sorted, labeled, and preprocessed to prepare them for analysis. The TF-IDF feature extraction method was employed to transform the preprocessed text data into numerical representations, followed by the incorporation of FastText word embeddings to enrich the feature vectors. This process enhanced the model's ability to capture contextual meaning from the text data. Subsequently, Logistic Regression, a robust classification algorithm, was utilized to classify the sentiment of the user reviews. The model was trained and evaluated using scenarios by combining TF-IDF with FastText word embeddings. The evaluation results demonstrated that the model achieved high performance in both scenarios, with accuracy, precision, recall, and F1-Score metrics exceeding 90%. This indicates the effectiveness of the combined approach in accurately classifying sentiment in LinkedIn user reviews. Overall, the study highlights the importance of employing advanced natural language processing techniques, such as TF-IDF and FastText, in sentiment analysis tasks. By leveraging these methodologies, developers and service providers can gain valuable insights into user feedback, enabling them to enhance the quality of their applications and services.

REFERENCES

A. D. Anggraeni, M. Farhansyah, M. R. P. Hermawan, G. W. Wicaksono, and C. S. K. Aditya, "Comparison of Classification Methods on Twitter Sentiment Analysis of PDAM Tugu Tirta Kota Malang," *JUITA J. Inform.*, vol. 11, no. 1, p. 67, 2023, doi: 10.30595/juita.v11i1.15485. [8]

A. M. Taufiqi and A. Nugroho, "Sentimen Pengguna Twitter Mengenai Isu Kebocoran Data Dengan Algoritma Naïve Bayes," *J. Nas. Ilmu Komput.*, vol. 4, no. 1, pp. 1–11, 2023, doi: 10.47747/jurnalnik.v4i1.1091. [13]

A. Puspita Sari, H. Suzuki, T. Kitajima, T. Yasuno, D. Arman Prasetya, and A. Rabi', "Deep convolutional long short-term memory for forecasting wind speed and direction," *SICE J. Control. Meas. Syst. Integr.*, vol. 14, no. 2, pp. 30–38, 2021, doi: 10.1080/18824889.2021.1894878. [3]

Arif Widiasan Subagio, Anggraini Puspita Sari, and Andreas Nugroho Sihananto, "Klasifikasi Lexicon-Based Sentiment Analysis Tragedi Kanjuruhan pada Twitter Menggunakan Algoritma Convolutional Neural Network," *J. Ilm. Sist. Inf. dan Ilmu Komput.*, vol. 4, no. 1, pp. 166–177, 2024, doi: 10.55606/juisik.v4i1.759. [14]

B. K. Widodo, N. H. Matondang, and D. S. Prasvita, "Penerapan Algoritma Naive Bayes Untuk Analisis Sentimen Penggunaan Aplikasi Jobstreet," *Techno.Com*, vol. 21, no. 3, pp. 523–533, 2022, doi: 10.33633/tc.v21i3.6361. [12]

D. Arianto and I. Budi, "Analisis Sentimen Berbasis Aspek dan Pemodelan Topik pada

Vol 4 No 3 (2024) 1359 – 1371 E-ISSN *2809-2058* P-ISSN 2809-2643
DOI: 47467/visa.v4i3.2835

Candi Borobudur dan Candi Prambanan," *Multinetics*, vol. 8, no. 2, pp. 141–150, 2023, doi: 10.32722/multinetics.v8i2.5056. [11]

D. E. Cahyani and I. Patasik, "Performance comparison of tf-idf and word2vec models for emotion text classification," *Bull. Electr. Eng. Informatics*, vol. 10, no. 5, pp. 2780–2788, 2021, doi: 10.11591/eei.v10i5.3157. [6]

D. H. Safitri, H. M. Az-Zahra, and M. C. Saputra, "Evaluasi Usability Sosial Media Profesional 'LinkedIn' Menggunakan Metode Usability Testing," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 6, no. 12, pp. 6019–6028, 2022, [Online]. Available: http://j-ptiik.ub.ac.id [1]

H. MacHhour and I. Kassou, "Concatenate text embeddings for text classification," *2017 Int. Conf. Internet Things, Embed. Syst. Commun. IINTEC 2017 - Proc.*, vol. 2018-Janua, pp. 206–210, 2017, doi: 10.1109/IINTEC.2017.8325940. [15]

H. R. Alhakiem and E. B. Setiawan, "Aspect-Bas1ed Sentiment Analysis on Twitter Using Logistic Regression with FastText Feature Expansion," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 6, no. 5, pp. 840–846, 2022, doi: 10.29207/resti.v6i5.4429. [4]

J. Pardede and I. Pakpahan, "Analisis Sentimen Penanganan Covid-19 Menggunakan Metode Long Short-Term Memory Pada Media Sosial Twitter," *J. Publ. Tek. Inform.*, vol. 2, no. 3, pp. 12–25, 2023. [10]

M. F. Rizki, W. Pramusinto, M. Hardjianto, and S. Subandi, "Implementasi Algoritma K-Nearest Neigbors Untuk Analisis Sentimen Aplikasi Jobstreet," *Pros. Semin. Nas. Mhs. Fak. Teknol. Inf.*, vol. 2, no. 1, pp. 267–276, 2023. [2]

M. S. Kabir and M. S. Arefin, "Google Play Store Data Mining and Analysis," *Vol*, vol. 12, no. 26, pp. 1–5, 2019, [Online]. Available: https://www.ijais.org/archives/volume12/number26/kabir-2019-ijais-451839.pdf [9]

R. Ahuja, A. Chug, S. Kohli, S. Gupta, and P. Ahuja, "The impact of features extraction on the sentiment analysis," *Procedia Comput. Sci.*, vol. 152, pp. 341–348, 2019, doi: 10.1016/j.procs.2019.05.008. [5]

Siti Khomsah, Rima Dias Ramadhani, and Sena Wijaya, "The Accuracy Comparison Between Word2Vec and FastText On Sentiment Analysis of Hotel Reviews," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 6, no. 3, pp. 352–358, 2022, doi: 10.29207/resti.v6i3.3711. [7]