**Name : Rehaan Shaikh**

**Student no : 24082638**

**CMT122 Machine learning for nlp coursework**

**Index:**

1. **Introduction .......................................................................... Page 3**
2. **Preprocessing Steps and Justifications ................................... Page 3**
3. **Feature Extraction and Experimentation ................................ Page 4**
4. **Dimensionality reduction …………………………………………………… Page 5**
5. **Training and Testing the Models ............................................. Page 5**
6. **Performance and Evaluation Metrics ...................................... Page 6**
7. **Model Comparison and evaluation (Extra credit)…………………… Page 7**
8. **Reflection on Model Improvements and Bias Considerations .. Page 6**

**Introduction**:

This report presents a comprehensive approach to text classification using the BBC News dataset, which contains articles across five categories: tech, business, sport, politics, and entertainment. The goal is to develop a reliable classification model that accurately categorizes articles while identifying areas for potential improvement.

**1. Preprocessing Steps and Justifications:**

We began by loading the dataset, which consists of BBC news articles labelled into five different categories. We aimed to prepare this text data for machine learning classification by applying essential preprocessing steps with the help of nltk. The following steps were used during preprocessing:

**Lowercasing**: The text was converted the text to lowercase to ensure consistency by merging capitalized and small-case versions of the same word (e.g., "Apple" and "apple") into one. This reduces the total number of unique vectors, optimizing the feature space and making the model's learning process more efficient without altering the text's meaning.

**Tokenization**: Tokenization involved breaking each article into individual words (tokens). This step allows us to process and analyse the text in smaller, manageable parts, enabling models to understand and represent language effectively. By structuring text into tokens, it becomes easier for vectorizers to convert words into numerical representations.

**Stopwords Removal**: Common English stopwords (like “and,” “the,” “is”) were removed to focus on words that carry more meaning. This step reduces the number of unnecessary words in the vocabulary, making the process more memory and computationally efficient.

**Lemmatization**: Lemmatization was applied to transform each word into its base or root form, standardizing the text further. For instance, words like "running" and "ran" were converted to "run." This step helps reduce variations of the same word, making the vocabulary smaller and improving the model’s efficiency.

**Justification**: This preprocessing pipeline is effective for removing irrelevant or unnecessary information, ensuring that the remaining words in each article are useful for identifying its category. Each step contributes to reducing computational complexity and enhances the quality of features used in model training.

**2. Feature Extraction and experimenting on dev set:**

We first trained a model using the Bag of Words vectorizer, achieving a development set accuracy of 95.06%. Next, we experimented with TF-IDF vectorization, which improved the development set accuracy to 96.85%. Given the better performance of TF-IDF, we chose it for our final model evaluation on the test set.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Metric** | **Using BoW**  **(Logistic Regression)** | **Using TF-Idf**  **(Logistic Regression)** | **Using BoW**  **(SVM)** | **Using TF-IDF**  **(SVM)** |
| **Dev Set**  **Accuracy** | 0.95056 | 0.9685 | 0.94157 | 0.9685 |
| **Dev Set Recall** | 0.95 | 0.97 | 0.94 | 0.97 |

To represent articles(text) in a machine-readable format, we then chose these 3 features:

1. **TF-IDF (Term Frequency-Inverse Document Frequency)** : This feature captures the importance of words in each document relative to the entire dataset. TF-IDF works by assigning a weight to each word based on its frequency in a document (TF) and how unique it is across all documents (IDF). We used TF-IDF with a maximum of 1000 features to reduce dimensionality while preserving essential terms.
2. **N-Grams**: To capture context, we used bigrams, or two-word combinations, with a TF-IDF vectorizer, extracting up to 1000 features. Bigrams can recognize specific phrases (e.g., "global market" in business), capturing more context than single words (“golabl” and “market”) alone.
3. **Word2Vec Embeddings**: Word2Vec creates dense vectors for words that reflect their meanings. Words with similar meanings are represented by vectors that are close to each other By averaging these word vectors, we generate a single vector that represents the overall content of the article.

**Justification**: These features complement each other by capturing different aspects of the text.

* **TF-IDF**: Highlights important words in the article by considering both their frequency and rarity across the corpus, ensuring distinctive words carry more weight.
* **Bigrams**: Adds context by capturing pairs of consecutive words, helping the model understand relationships between terms within the article.
* **Word2Vec Embeddings**: Captures deeper semantic relationships, grouping similar meanings together, allowing the model to understand nuances beyond individual words.
  + Combining these features provides us with both statistical (TF-IDF) and semantic (Word2Vec, bigrams) information for more accurate predictions.

1. **Dimensionality Reduction via Feature Selection:**

Since the dataset is not inherently high-dimensional, advanced techniques like Principal Component Analysis (PCA) were unnecessary.

Dimensionality reduction was achieved using parameters within the TF-IDF and ngrams vectorization methods, such as max\_features =1000. This approach ensured that only the most important features (e.g., the top 1000 terms by relevance) were retained, reducing the feature space effectively.

This approach made the process memory- and computationally efficient.

1. **Training and testing of the model:**

We chose a logistic regression model and a support vector machine (SVM) classifier for classification. These models are effective for text classification tasks and have well-established performance metrics. To assess model performance, we split the data into training, development, and test sets:

* **Train-Dev-Test Split**: We split the dataset by reserving 20% for testing and using the remaining 80% for training. Of the training data, 75% was used for training the model, and 25% was set aside for Dev set (tuning model parameters). This division allowed us to tune model parameters on the development set before final evaluation on the test set.
* **Justification**: Splitting data this way helps to evaluate generalization, minimizing overfitting and ensuring consistent performance by giving sufficient sample for training and testing.

**5. Report on Performance and Evaluation Metrics :**

The performance of the logistic regression model was evaluated on both the development and test sets, yielding highly promising results. Here's a summary:

* **Test Set Accuracy:**  
  On the test set, the model improved it’s accuracy by reaching up to 98.20%, demonstrating its strong generalization ability to unseen data.

**Macro-Averaged Precision, Recall, and F1-Score:**

1. **Development Set Results**

• Accuracy: 96.85%

**Macro-averaged metrics:**

* Precision: 0.97
* Recall: 0.97
* F1-score: 0.97

1. **Test Set Results:**

* Accuracy: 98.20 %

**Macro-averaged metrics :**

* Precision: 0.98
* Recall: 0.98
* F1-score: 0.98

**Key Observations:**

* The model achieved an accuracy of 96.85% on the development set, reflecting the effectiveness of our preprocessing and feature selection strategies.
* On the test set, the model improved it’s accuracy by reaching up to 98.20%, demonstrating its strong generalization ability to unseen data.
* The sport category had perfect scores (precision, recall, and F1) on both the development and test sets, signifying unambiguous classification for this category.

**7**. **Model Comparison and Evaluation: (Extra Credit)**

In this section, we compared three machine learning models—Logistic Regression, Support Vector Machine (SVM), and Random Forest. Each model's performance isevaluated using accuracy, precision, recall, and F1-score metrics on the development and test sets. Below is a detailed comparison and analysis.

**Performance Comparison:**

The table below summarizes the performance metrics for each model based on the test set:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Test Set Accuracy** | **Precision (Macro Avg)** | **Recall (Macro Avg)** | **F1-Score (Macro Avg)** |
| Logistic Regression | 0.982 | 0.98 | 0.98 | 0.98 |
| SVM | 0.969 | 0.97 | 0.97 | 0.97 |
| Random Forest | 0.978 | 0.98 | 0.98 | 0.98 |

**Analysis of Results:**

1. **Logistic Regression**: Logistic Regression had the highest accuracy (98.2%) on the test set. It performed well across all categories, showing its ability to handle balanced datasets effectively. This model is simple to use and easy to understand, making it a great choice. However, it might not work as well for more complex patterns in data.
2. **Support Vector Machine (SVM)**: SVM achieved an accuracy of 96.9% and delivered strong precision and recall. Although it didn’t perform as well as Logistic Regression or Random Forest, SVM is great for handling high-dimensional data like text. The downside is that it can be slower and harder to scale for larger datasets.
3. **Random Forest**: Random Forest performed very well, with a test accuracy of 97.8%. It had similar precision, recall, and F1-scores to Logistic Regression. Random Forest is powerful for handling complex data but is harder to interpret compared to Logistic Regression.

**Strengths and Limitations:**

* **Logistic Regression**:
  + Strengths: Simplicity, interpretability, and high accuracy.
  + Limitations: Limited ability to capture non-linear relationships.
* **SVM**:
  + Strengths: Handles high-dimensional data effectively.
  + Limitations: Computationally expensive, slightly lower performance on this dataset.
* **Random Forest**:
  + Strengths: Excellent accuracy, robust to overfitting, handles complex relationships well.
  + Limitations: Less interpretable compared to Logistic Regression.
* **Conclusion:**

While Logistic Regression provides the highest accuracy, Random Forest offers a competitive alternative with similar precision and recall. SVM, though slightly lagging in accuracy, remains a viable option due to its effectiveness in high-dimensional spaces. Overall, the choice of model depends on the specific use case, with Logistic Regression being ideal for interpretability and Random Forest for handling more complex tasks.

**6. Reflection on Model Improvements and Bias Considerations:**

While this approach achieves reasonable accuracy, several improvements could enhance performance:

* **Advanced Embedding Models**: Using transformer-based models like BERT can improve understanding of context and semantics in text.
* **Hyperparameter Tuning**: Optimizing model parameters, especially for SVM (like regularization and kernel type), could optimize classification.
* **Dimensionality Reduction**: Techniques like Principal Component Analysis (PCA), t-SNE and Feature Importance from Random Forest can help reduce redundant features, improving efficiency and preventing overfitting.

**Bias Considerations**: Potential biases may cause several issues such as :

* **Class Imbalance**: Addressing uneven category representation through oversampling or adjusting class weights can reduce bias.
* **Labeling Subjectivity**: Clear guidelines for article categorization can ensure more consistent training data.