ADVANCED TEXT CLASSIFICATION PROJECT

NLP-COURSE
AN-NAJAH NATIONAL UNIVERSITY

Prepared by: Farah Saleh

supervised by: Dr.Hamed Abdalhaq

GOAL

The goal of this text classification project is to construct a model that can effectively assign one of 91 predefined classes to a given document. These classes represent various categories or topics that the documents may belong to.

DATASETS

- *Training dataset: This dataset consists of a collection of 91 folders, each representing a distinct class or category. The documents within each folder are labeled according to their respective classes.*
- *Test dataset: Similar in structure to the training dataset*

0 3 METHODOLOGY

Data Preprocessing:

- **Tokenization:** Each document was tokenized into individual words or tokens to facilitate further processing.
- **Stop Words and non-alphabetic tokens Removal:** Commonly occurring words with little semantic value, such as "the," "is," and "and," were removed to focus on more meaningful content.
- **Normalization:** Text was converted to lowercase to ensure consistency in word representations.
- **Lemmatization** :Lemmatization was applied to reduce inflected words to their base or dictionary form (i.e., lemma). This process ensured enhancing the effectiveness of feature extraction and model performance.

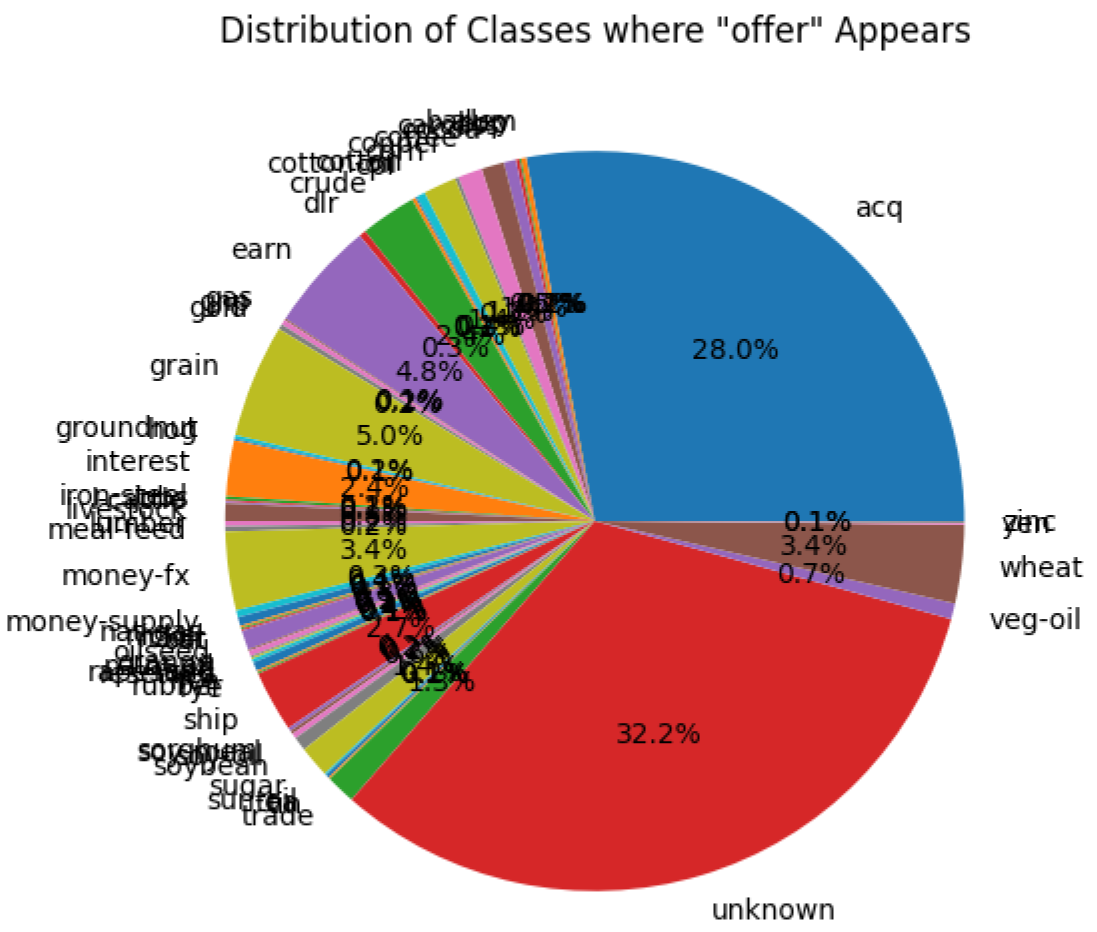
04 EDA :

In my exploratory data analysis , I delved into understanding the nature of my dataset, particularly focusing on identifying similarities and distinctions between classes. One prominent observation was the similarity in document content across different classes, despite variations in class names. To investigate this further, I selected two classes—'acq' and 'unknown'—where my model encountered challenges in accurate classification.

Upon examining the documents within these classes, I discovered recurring keywords such as ('negotiate,' 'offer,' and 'acquisition.') These terms appeared frequently across documents in both the 'acq' and 'unknown' classes. Notably, the presence of 'acq' or 'acquisition' served as a strong indicator that a document belonged to the 'acq' class.

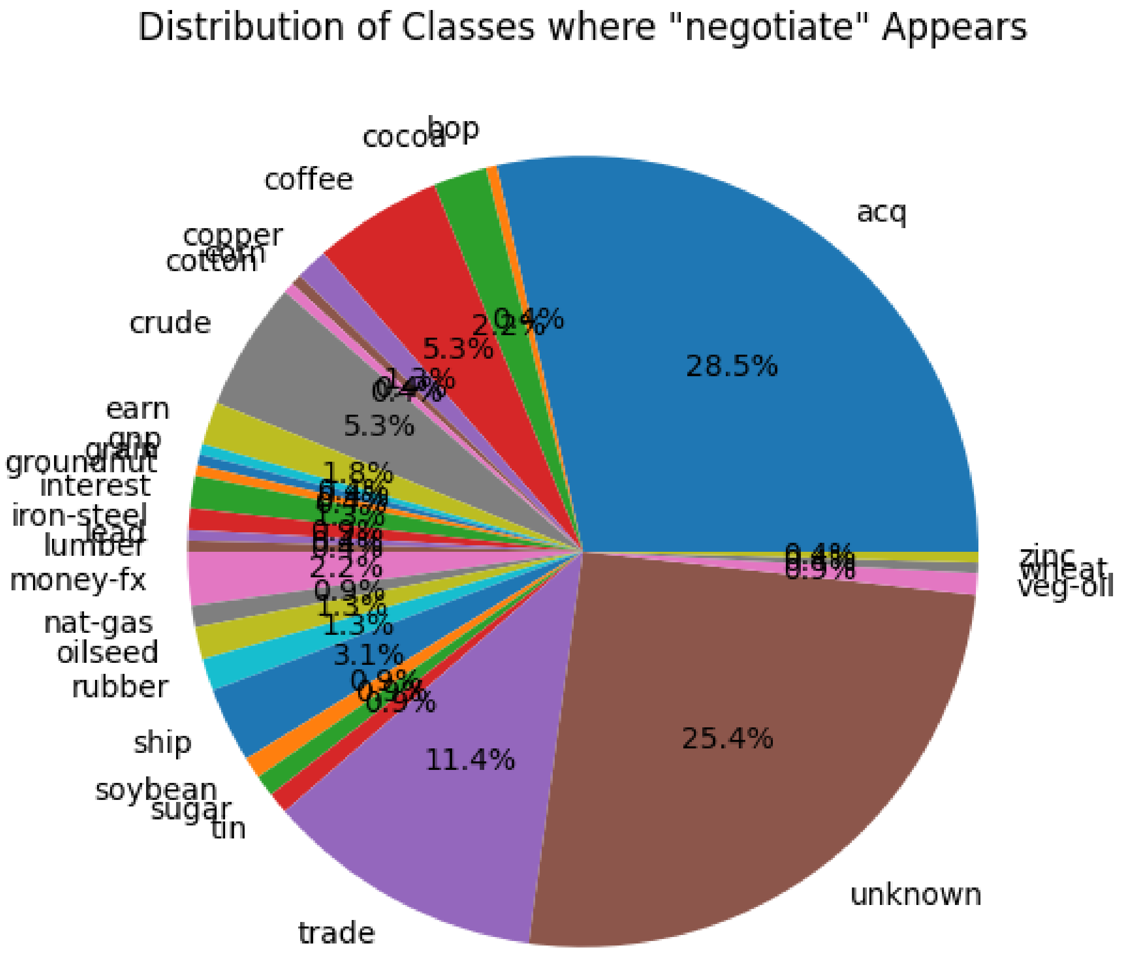
To visualize and validate these findings, I employed regular expressions (regex) to search for occurrences of these keywords within the document content

05 EDA :



The presence of (offer) seems to play a crucial role in determining that the class is either “acq” or “unknown”

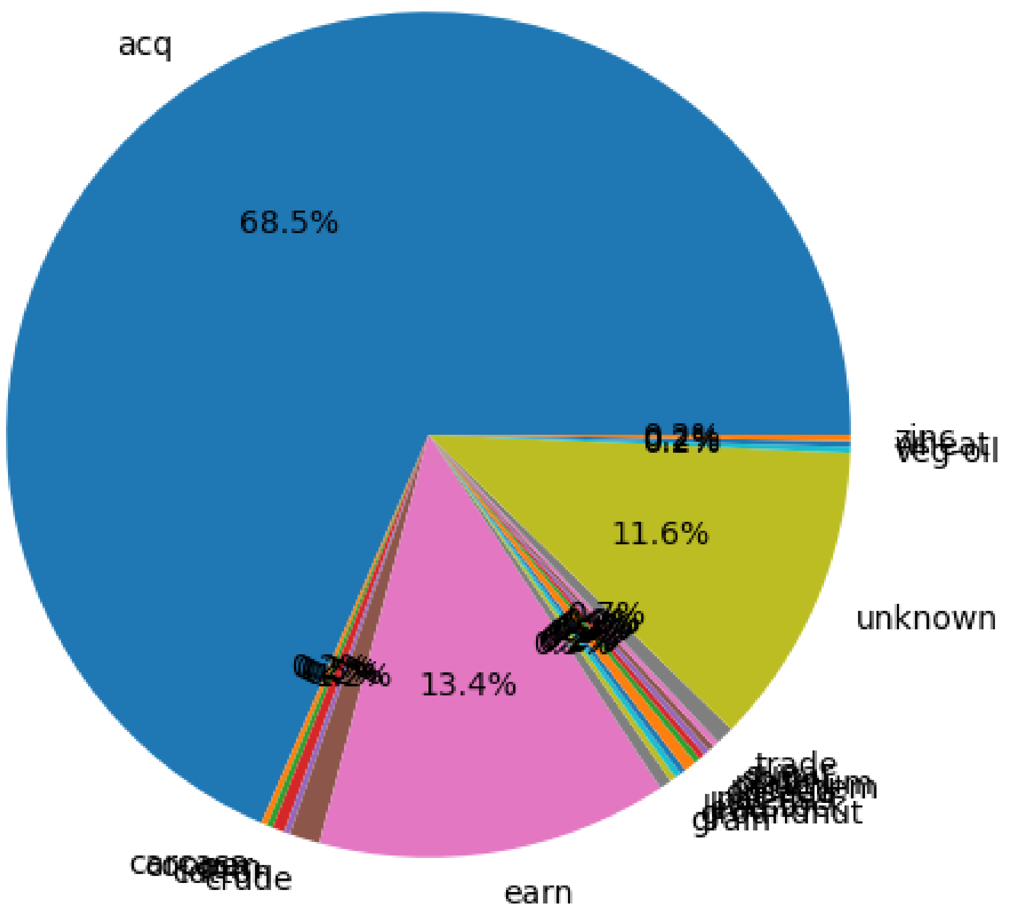
06 EDA :



The presence of (negotiate) seems to play a crucial role in determining that the class is either “acq” or “unknown”

07 EDA :

Distribution of Classes where "acq" Appears



But for determining whether it’s “acq” or “unknown” the presence of term (acquisition or acq) is a strong evidence that the document belongs to class “acq”

08 METHODOLOGY

Feature Encoding:

- Vectorization: The text data was transformed into numerical vectors using techniques like TF-IDF (Term Frequency-Inverse Document Frequency) and word embeddings (fasttext , glove , word2vec) to make it suitable for machine learning algorithms.

09 METHODOLOGY

Model Selection and Training:

- Random forest
- Ensemble (KNN , random forest , decision tree classifier , Naive Bayes)
- SVM
- LSTM

METHODOLOGY

Random Forest (TF-IDF):

Data Representation

I represented the textual data using the Term Frequency-Inverse Document Frequency (TF-IDF) method. a matrix where each row represents a document and each column represents a unique term in the corpus then convert it into dense array. This representation captures the discriminative power of words in distinguishing between different classes.

Model Selection

- The chosen model is a Random Forest classifier .
- Random Forest is selected for its ability to handle high-dimensional data and its robustness against overfitting.
- It works well with TF-IDF representations as it can effectively capture the nonlinear relationships between features.

Evaluation Metrics

I chose the macro F-score as evaluation metric because the classes in the dataset are imbalanced. This metric provides a balanced assessment of the classifier's performance across all classes, which is crucial in scenarios where class distribution is uneven.

METHODOLOGY

Random Forest (TF-IDF):

Results :

Macro F-score on Test Data : 0.1875

Macro F-score on new Data : 0.203

while Random Forest with TF-IDF is a reasonable choice for modeling text data, the macro F-score on both test and new data is low , one of the reasons behind it maybe the data , classes are imbalanced meaning that one or more classes have significantly fewer instances compared to others. This can pose challenges for machine learning models, so I used Class Weighting technique. By assigning higher weights to minority classes and lower weights to majority classes, the model is encouraged to pay more attention to the minority classes during training.

The results after :

Macro F-score on Test Data : 0.2325

Macro F-score on new Data : 0.3177

METHODOLOGY

Random Forest (embedding):

Data Representation

I represented the textual data using embeddings because they excel at capturing semantic information. I experimented Word2Vec, GloVe, and FastText embeddings. Among these, FastText consistently outperformed Word2Vec, while GloVe performed better than both Word2Vec and FastText in our classification experiments.

Model Selection

Random Forest Classifier was chosen as the classification model for its ability to handle high-dimensional data and capture non-linear relationships.

Evaluation Metrics

I chose the macro F-score as evaluation metric because the classes in the dataset are imbalanced. This metric provides a balanced assessment of the classifier's performance across all classes, which is crucial in scenarios where class distribution is uneven.

I conducted experiments using three different word embedding techniques - Word2Vec, FastText, and GloVe - each coupled with a Random Forest classifier.

METHODOLOGY

The macro-averaged F1 scores obtained for each combination are as follows:

- **Word2Vec**: 0.1393
- **FastText**: 0.1119
- **GloVe**: 0.1834

the differences in F1 scores between the embeddings are relatively small. This indicates that while GloVe performed slightly better on average, all three embeddings yielded fairly similar classification results when combined with the Random Forest classifier.

Furthermore, when the model predicted on a different dataset, the macro-averaged F1 scores for all embeddings showed a substantial improvement:

- **Word2Vec**: 0.1620
- **FastText**: 0.1313
- **GloVe**: 0.2151

In this case, the differences in F1 scores between the embeddings are minimal, suggesting that the choice of embedding technique had less impact on classification performance with this dataset. Instead, other factors such as the characteristics of the data or the effectiveness of the classifier may have played a more significant role in determining performance.

.

Random Forest (embedding):

i observed an improvement in performance when the model was evaluated on new data , this improvement suggests that the model generalize well and capable of performing effectively on unseen data. While the exact reasons for this improvement may vary, it underscores the importance of evaluating model performance on diverse datasets to ensure robustness and reliability.

In conclusion, the methodology provides a framework for text classification using embeddings , TF-IDF and Random Forest Classifier. While the obtained results suggest room for improvement, they also highlight the importance of methodological choices in achieving accurate classification. Further experimentation and refinement of the methodology are warranted to advance the state-of-the-art in text classification.

METHODOLOGY

Ensemble using KNN, decision tree classifier, Random forest, and Naive Bayes models:

Data Representation

I represented the textual data as (TF-IDF) dense array.

Model Selection

The ensemble model comprises four classifiers: K-Nearest Neighbors (KNN), Decision Tree, Random Forest, and Naive Bayes (NB). Each classifier is trained using the TF-IDF vectors of the training data. KNN is trained with 5 neighbors, while both Decision Tree and Random Forest models are trained with class weights to handle imbalanced data. The Naive Bayes model is trained separately using custom functions to calculate prior and likelihood probabilities.

Predictions are generated from each individual classifier using the TF-IDF vectors of the test data.
For Naive Bayes, predictions are obtained by calculating class probabilities and selecting the class with the highest probability for each document.
The majority voting ensemble method is employed to combine predictions from all classifiers. For each test sample, the final prediction is determined by selecting the class with the most occurrences among the predictions of all classifiers.

METHODOLOGY

Ensemble using KNN, decision tree classifier, Random forest, and Naive Bayes models:

Results :

- Macro-averaged F1 score on the test data: 0.2525

- Macro-averaged F1 score on the new data: 0.3089

- The ensemble model achieves a relatively low macro-averaged F1 score on the test data, indicating modest performance.
- However, the F1 score improves on new data, suggesting that the model generalizes better to unseen samples.
- The increase in F1 score on new data may imply that the ensemble model captures underlying patterns in the data more effectively than individual classifiers alone.

METHODOLOGY

SVM Classifier :

Data Representation

I represented the textual data as glove embedding.

Model Selection

- The choice of the SVM classifier is a deliberate decision made based on its effectiveness in handling high-dimensional data, such as text data represented by word embeddings (e.g., Word2Vec).
- SVMs are known for their ability to find optimal hyperplanes that separate data points into different classes while maximizing the margin between them. This makes them well-suited for classification tasks with complex decision boundaries.
- A parameter grid is defined to tune the hyperparameters of the Support Vector Machine (SVM) classifier using grid search.
- The grid includes different choices for the kernel function ('poly' and 'rbf'), regularization parameter C (0.1, 1, and 10), and gamma values ('scale' and 'auto') for the 'rbf' kernel.
- The grid search is performed using 5-fold cross-validation and optimized for macro F1 score.

METHODOLOGY

SVM Classifier :

- The best hyperparameters: {C=10, gamma='scale', kernel='poly'}.
- These hyperparameters are then used to initialize the SVM classifier for further training and evaluation.

Results :

- Macro-averaged F1 score on the test data: 0.269
- Macro-averaged F1 score on the new data: 0.3126

This improvement in performance on new data compared to the test data indicates that the SVM classifier generalizes well and effectively captures underlying patterns in unseen samples and the chosen hyperparameters (C=10, gamma='scale', kernel='poly') seem to lead to a well-generalized SVM classifier that performs robustly across different datasets.

METHODOLOGY

LSTM :

Data Representation

- Text data preprocessing begins with tokenization, where each document's content is split into individual tokens (words).
- This is achieved using the Tokenizer class from TensorFlow's preprocessing module, which converts text into sequences of integers.
- To ensure uniform input size for the RNN model, sequence padding is applied using the pad_sequences function.
- The resulting tokenized and padded sequences serve as the input data for training the RNN model.

Model Selection

he RNN model architecture includes an embedding layer, LSTM layer, dense hidden layer, and output layer. The embedding layer converts integer-encoded tokens into dense vectors, facilitating semantic understanding. The LSTM layer captures temporal dependencies in sequential data. A dense hidden layer adds non-linearity for learning complex patterns. The output layer predicts class probabilities using softmax activation. Together, these components create a powerful RNN model for processing and classifying sequential text data.

.

METHODOLOGY

LSTM :

Training Progress and Performance:

- During training, the model's accuracy and loss are monitored at each epoch.
- The training progresses over 10 epochs, with notable improvements observed in both accuracy and loss.
- The model's accuracy increases steadily from around 23% to approximately 77%, while the loss decreases from 4.9 to 0.7 .
- Validation accuracy and loss also show consistent improvement, indicating that the model is learning useful representations and generalizing well to unseen data.
- Macro F1 Score: 0.11927

The RNN model demonstrates significant learning capability and generalization performance over the course of training. The steady increase in accuracy and decrease in loss suggest that the model is effectively capturing the underlying patterns in the text data.