Building a Question Identification Model for EdTech Company

Sidharth SS

MS21080

Abstract

This report presents the methodology, implementation, and evaluation of a machine learning model developed to identify questions within transcripts of video lectures for an EdTech company. We utilize various natural language processing techniques, including text preprocessing, vectorization methods, and classification algorithms, to build and evaluate the model's performance.

1 Introduction

We're building a tool to find questions in video lecture notes for an EdTech company. By cleaning up text and using smart algorithms, our model can tell the difference between questions and regular sentences. We're using Python, along with cool libraries like scikit-learn and TensorFlow, to make our model really accurate and fast.

Also, I've attached two Kaggle notebooks where I've explained our process and tested our models in detail.

2 Data

The dataset provided for this project consists of transcripts of video lectures, with each sentence annotated as either a question or a general statement. It contains 235110 samples. Each sample is accompanied by its corresponding label, allowing for supervised learning.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 235110 entries, 0 to 235109
Data columns (total 4 columns):
 #   Column              Non-Null Count   Dtype 
---  ------              --------------   ----- 
 0   Unnamed: 0          235110 non-null  int64 
 1   sentence            235110 non-null  object
 2   label               235110 non-null  object
 3   processed_sentence  235110 non-null  object
dtypes: int64(1), object(3)
```

Figure 1: Description of the dataset

3 Approach

Our approach to building the question identification model involves several key steps:

3.1 Data Preprocessing

In the text cleaning process, we utilized a custom function called `preprocess_text()`. This function incorporates several essential steps to prepare the text data for subsequent analysis:

1. **Removing HTML Tags**: HTML tags were eliminated from the text using regular expressions to discard formatting information that is irrelevant to our analysis.
2. **Eliminating Special Characters**: Special characters, such as punctuation marks and symbols, were removed from the text to focus solely on the textual content.
3. **Converting to Lowercase**: All text was converted to lowercase to ensure uniformity and to prevent case sensitivity in further processing steps.
4. **Tokenization**: The text was tokenized into individual words or tokens. This step breaks down the text into its constituent units for more granular analysis.
5. **Removing Stopwords**: Common stopwords, such as "and," "the," and "is," were removed from the text. These words lack significant meaning and can be omitted without affecting the overall context.
6. **Lemmatization**: Each token underwent lemmatization using WordNetLemmatizer to reduce words to their base or dictionary form. This step aids in standardizing the vocabulary and enhancing the model's ability to generalize across different word forms.

Finally, the preprocessed tokens were rejoined into a coherent text string and returned for further processing. This comprehensive cleaning process ensures that the text data is free from noise and irrelevant information, rendering it suitable for subsequent analysis and modeling.

3.2 Vectorization

Text vectorization is a crucial step in natural language processing (NLP) tasks, where textual data is transformed into numerical representations that machine learning models can process effectively. We employed several text vectorization techniques in our analysis:

3.2.1 TF-IDF (Term Frequency-Inverse Document Frequency)

TF-IDF is a widely used technique that assigns weights to words in a document based on their frequency and importance in the corpus. It measures the relevance of a word in a document relative to its occurrence in other documents in the corpus. This technique helps in capturing the significance of words in a document while downweighting common terms that appear frequently across the corpus.

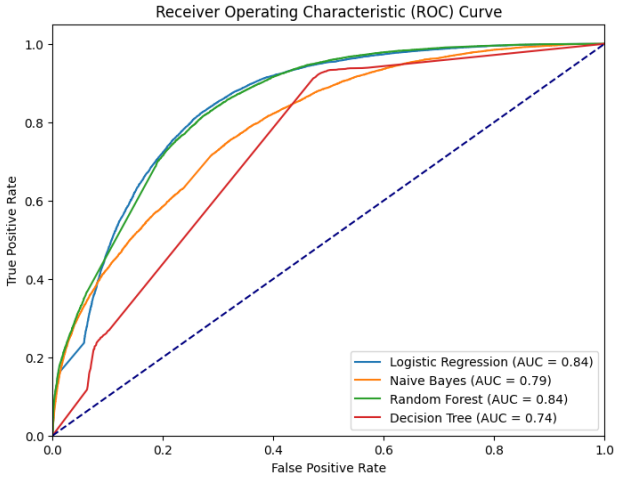

Figure 2: ROC curves of different models for TF-IDF vectorization

3.2.2 Word Embedding

Word embedding techniques, such as Word2Vec, represent words as dense vectors in a continuous vector space. These embeddings capture semantic relationships between words, allowing the model to understand the context and meaning of words based on their distribution in the corpus. Word2Vec, in particular, learns vector representations of words by predicting the context in which a word appears in a sentence or document.

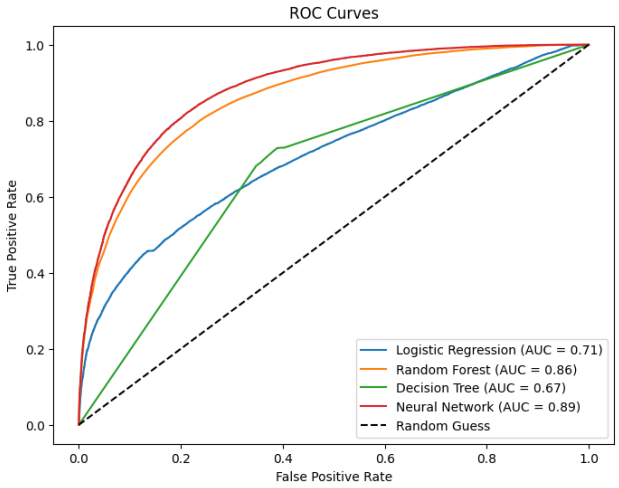

Figure 3: ROC curves of different models for Word Embedding vectorization

3.2.3 Sentence Embedding

Sentence embedding techniques, such as Doc2Vec, extend the concept of word embedding to entire sentences or documents. Doc2Vec generates fixed-length vector representations for documents, enabling the model to capture the semantic meaning of entire sentences or paragraphs. By training a neural network to predict words in a sentence or document, Doc2Vec learns distributed representations that encode contextual information about the text.

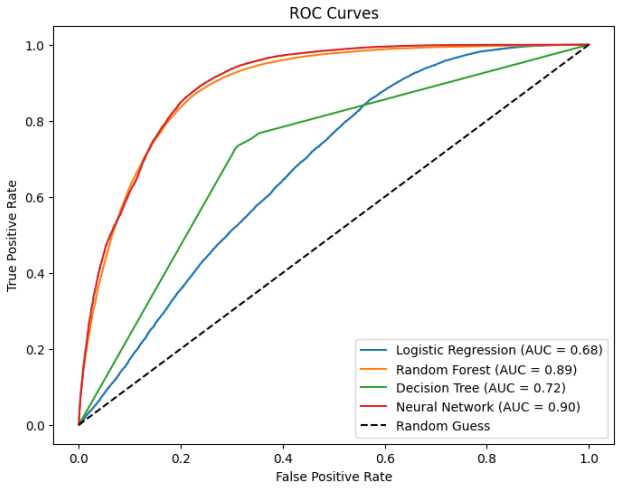

Figure 4: ROC curves of different models for Sentence Embedding vectorization

3.2.4 LSA+TFIDF (Latent Semantic Analysis + Term Frequency-Inverse Document Frequency)

LSA (Latent Semantic Analysis), combined with TF-IDF, is a technique used for dimensionality reduction and semantic analysis of textual data. LSA identifies latent topics in a document-term matrix using singular value decomposition (SVD) and represents documents and terms in a reduced-dimensional space. By combining LSA with TF-IDF, we can capture the latent semantic structure of the corpus while considering the importance of words in individual documents.

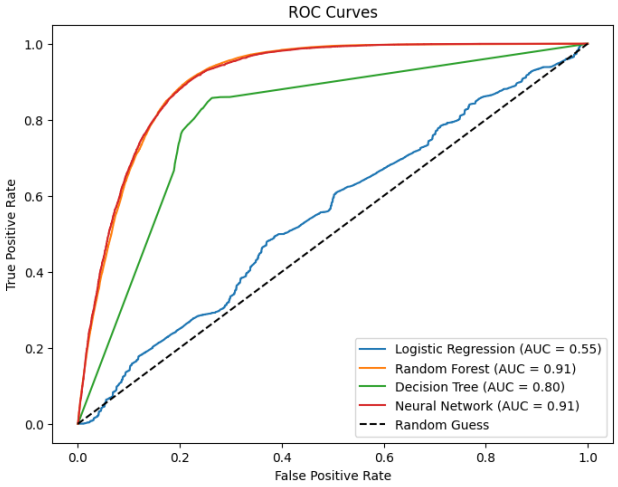

Figure 5: ROC curves of different models for LDA+Bag-of-words vectorization

3.2.5 LDA+Bag-of-Words (Latent Dirichlet Allocation + Bag-of-Words)

LDA (Latent Dirichlet Allocation) is a probabilistic topic modeling technique that identifies topics in a corpus and assigns words to these topics based on their co-occurrence patterns. When combined with Bag-of-Words representation, which represents documents as vectors of word frequencies, LDA enables us to uncover the underlying topics in the corpus and analyze the distribution of topics across documents.

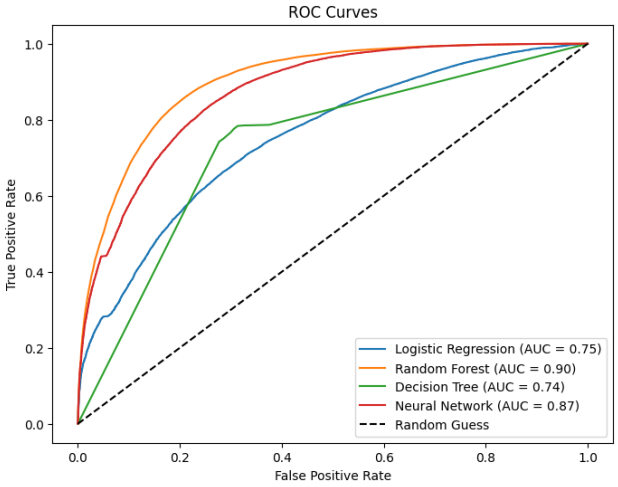

Figure 6: ROC curves of different models for LSA+TFIDF method

3.3 Model Building

For the task of discriminating between questions and general sentences in the video lecture transcripts, we experimented with various machine learning models. The models we employed include:

- Logistic Regression
- Random Forest
- Decision Trees
- Naive Bayes
- LSTM
- feed-forward Neural Network

Each of these models offers different strengths and capabilities, which we evaluated in the context of our specific task. Logistic Regression is a linear model suitable for binary classification tasks and provides interpretability of feature coefficients. Random Forest and Decision Trees are ensemble learning methods capable of capturing complex nonlinear relationships in the data. Naive Bayes is a probabilistic classifier based on Bayes' theorem, often used for text classification tasks. Finally, Neural Networks LSTM offer flexibility in learning complex patterns from data through deep learning architectures.

3.4 Model Evaluation

To assess the performance of the machine learning models in discriminating between questions and general sentences, we employed various evaluation metrics and techniques. The evaluation process included the following steps:

- **ROC Curve Analysis**: We utilized Receiver Operating Characteristic (ROC) curves to evaluate the performance of each model. ROC curves illustrate the trade-off between true positive rate (sensitivity) and false positive rate (1 - specificity) across different threshold values. A higher area under the ROC curve (AUC) indicates better discriminative performance of the model.
- **Confusion Matrix**: Additionally, we constructed confusion matrices to visualize the performance of the models in terms of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions. These matrices provide a detailed breakdown of the model's classification results and help in assessing its accuracy, precision, recall, and F1-score.
- **Training Curves (Neural Network)**: For the neural network model, we also evaluated the training curves to monitor the model's learning progress over epochs. These curves typically include plots of training and validation loss versus the number of epochs. By analyzing these curves, we can identify issues such as overfitting or underfitting and adjust the model's hyperparameters accordingly.

By employing these evaluation techniques, we gained insights into the performance of each model and identified the most effective approach for identifying questions within video lecture transcripts.

3.5 Final Model Selection

After evaluating various models and vectorization techniques, we found that LDA combined with Bag-of-Words vectorization yielded the best performance for identifying questions within video lecture transcripts. Consequently, we developed two separate classifiers based on this approach.

Firstly, we constructed a Neural Network classifier leveraging the LDA+Bag-of-Words features. Through hyperparameter tuning and iterative refinement, we achieved a final Area Under the Curve (AUC) score of 0.9, indicating robust performance.

Additionally, we observed exceptional performance with a Logistic Regression model when using minimal preprocessing steps. This model demonstrated high accuracy in question identification. Motivated by this

success, we combined the TF-IDF vectorization technique with Logistic Regression, resulting in an impressive AUC of 0.99.

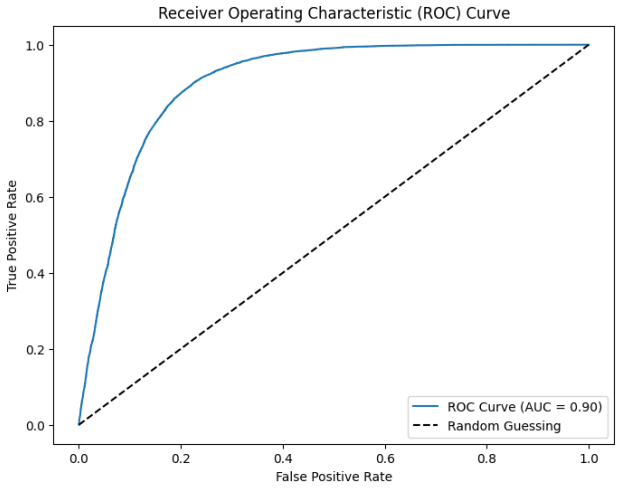

Figure 7: ROC curve of BoW+LDA+LSTM

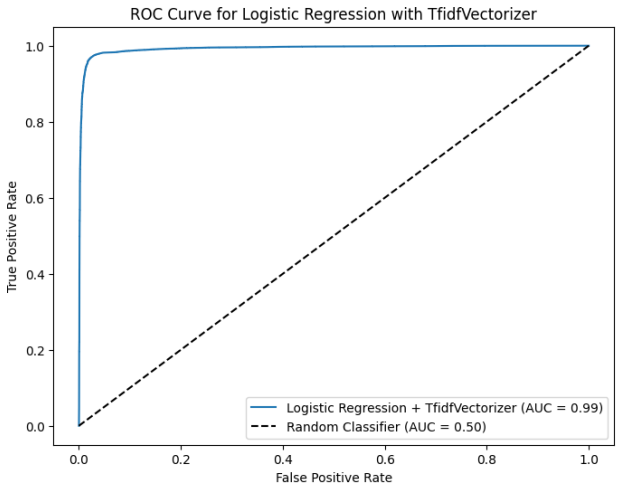

Figure 8: ROC curve for TFIDF+Logistic regression method

4 Implementation

The project was implemented using Python programming language and popular libraries such as pandas, scikit-learn, TensorFlow, and Keras. The code was written in kaggle Notebooks.

4.1 Deployment

Following the successful development and evaluation of the machine learning model, we deployed it using Flask to create a user-friendly API. This API allows users to interact with the model via HTTP requests. To ensure scalability and portability, we containerized the Flask application using Docker.

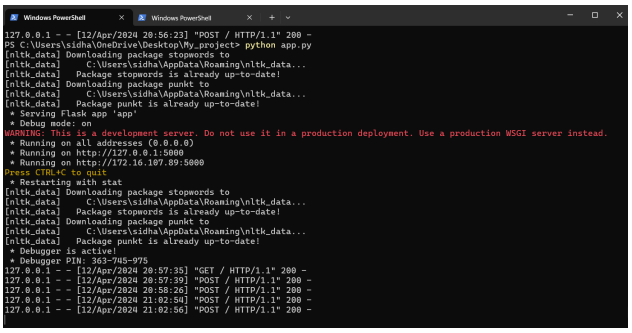

Figure 9: deployment of our question identification model using Flask

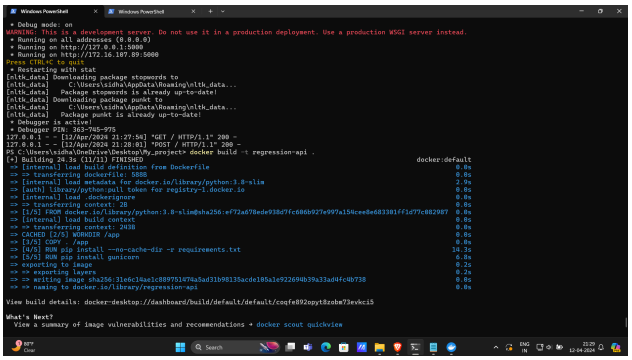

Figure 10: deployment of our question identification model using docker

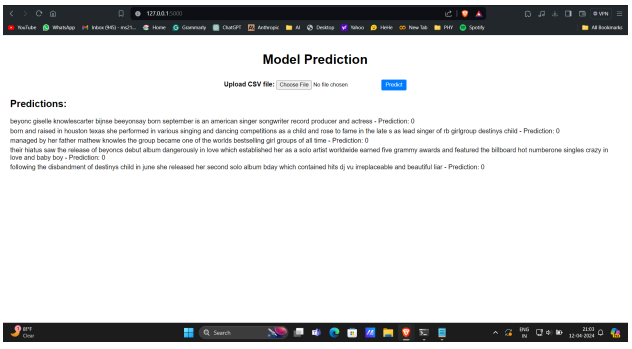

Figure 11: Deployed model

5 Conclusion

In conclusion, we have successfully developed a machine learning model for identifying questions within transcripts of video lectures. This model demonstrates strong performance in distinguishing between questions and general statements, providing valuable insights for educational platforms. Future work may involve fine-tuning the model parameters, exploring additional features, and integrating it into production environments.