Comparing and evaluating different techniques for Interview Response Evaluation

Team 26
Prachi Sarda, P Mohith Krishna, Mahitha Siri, Sreeya K
Mahindra University, Hyderabad, India.

1. Abstract

The lack of accessible, real-time and personalized feedback on interview responses poses a significant challenge for job seekers aiming to refine their content quality, structure, and confidence. Specifically, we focus on predicting whether an interview answer is of good, average, or poor quality, with the goal of providing a reliable tool for evaluating responses at a scale. The project utilizes a variety of textual features, including sentiment analysis, word count, sentence structure variability, and text complexity, all of which are extracted from the answers. We employ two machine learning models—Decision Tree and Support Vector Machine (SVM)—to perform the classification. To address data imbalance, the SVM model incorporates SMOTE (Synthetic Minority Over-sampling Technique), ensuring that the model generalizes well across underrepresented classes.

Keywords: Interview response, NLP-driven, Support Vector Matrix, Decision Trees

2. Introduction

Job interview preparation is often constrained by the absence of real-time, personalized feedback, leaving candidates unsure of how to improve their responses. Conventional tools either lack the depth required or are not easily accessible to all job seekers. This research addresses this challenge by utilizing Natural Language Processing (NLP) to provide real-time, tailored analysis of interview responses, making it both affordable and effective.

This paper presents a study focused on the development and evaluation of an automated system for classifying interview answers based on their quality. The system categorizes responses into three distinct classes—good, average, and poor—using a combination of textual features extracted from the answers. The features considered in this study include sentiment analysis, word count, sentence structure variability, and text complexity, which serve as indicators of the answer's overall quality. The primary goal of this research is to explore the potential of machine learning models, specifically Decision Tree and Support Vector Machine (SVM), in automating the qualitative assessment of interview responses. This study also discusses problems like data imbalance, under-represented classes, to mitigate which, synthetic methods like SMOTE were used in the SVM model.

Future work can further refine these models through hyperparameter tuning, additional feature engineering, and the incorporation of more advanced machine learning techniques to improve classification accuracy and robustness.

3. Prior Related Work:

Research on interview training has traditionally emphasized non-verbal communication and social behaviors. For example, the TARDIS project utilized virtual agents to help candidates improve facial expressions and body movements, with machine learning algorithms analyzing their emotional states (Anderson et al., 2013; Gebhard et al., 2018). Similarly, MACH—My Automated Conversation coach provides feedback on facial expressions, speech, and tone, focusing on emotional and social dynamics during interviews (Hoque et al., 2013). Another approach by Hartholt et al. (2019) involved a virtual reality system simulating diverse interview scenarios, enabling candidates to practice varying interviewer styles and settings in realistic environments.

Studies have also explored using multimodal data to assess interviewee traits. For example, Chen et al. (2017) employed a Bag-of-Words model to analyze textual responses and estimate personality traits. Hemamou et al. (2019a, b) developed HireNet, which used text, audio, and video data from interviews to predict hire-ability. Similarly, Nguyen & Gatica-Perez (2016) and Muralidhar et al. (2016) investigated how acoustic and visual cues in video resumes reflect communication skills linked to hiring decisions. Other papers that we refered to are in the references section.

4. Description of Dataset:

The dataset used in this research was meticulously created from scratch, building on top of a dateset found literature [1]. The dataset consists of 2,053 question-and-answer pairs

Dataset Overview

Total Rows: 2,053 Columns: 5

Columns:

A. Position/Role: This column includes the specific job title or role related to each question-and-answer pair.
B. Question: Each row includes a job-specific interview question
C. Answer: This column contains the corresponding answer for each interview question.
D. Interview Phase: This column indicates the stage of the interview process in which the question is typically asked.
E. Answer Quality: A critical component of the dataset, the Answer Quality column provides a manual rating of each answer.

5. Methodology: NLP Models and Framework Design

The methodology for this study is structured around three primary steps: data collection and preprocessing, feature extraction, and model development and evaluation. The focus is on automating the classification of interview answers as good, average, or poor using machine learning (ML) and natural language processing (NLP) techniques.

1. Data Collection and Preprocessing

1.1 Data Collection

The dataset used for this study consists of interview responses from a CSV file that contains multiple columns, including the interviewee's answer and the associated quality label. The answers are categorized into three classes: good, average, and poor. The target variable, Answer Quality, is encoded as follows: Good = 1; Average = 0; Poor = -1

1.2 Data Cleaning

The raw interview responses were cleaned to improve data quality and suitability for analysis. All answers were converted to lowercase to maintain consistency in the data. Non-ASCII characters, special characters, and digits were removed to eliminate irrelevant noise from the text. Common words such as "the", "and" "is", etc., were removed using a predefined list of stop words from the nltk library and English stop words from the sklearn library. Words were lemmatized using the WordNetLemmatizer from nltk to reduce inflectional forms to their root words. Any rows with 'bad' labels were replaced with 'poor' to standardize the dataset. Answers labeled as 'good' or 'bad' were also removed to balance the dataset.

1.3 Data Splitting

The dataset was then split into training and testing sets using an 80-20 split, ensuring that 80% of the data was used for training the models and 20% for testing the models. This split is crucial for evaluating the performance of the models on unseen data.

2. Feature Extraction

2.1 Textual Features

Several linguistic and statistical features were extracted from the interview responses to provide the model with meaningful information about the quality of the answers: The total number of words in each interview answer was computed. The sentiment polarity of each answer was computed using the TextBlob library, which provides a score between -1 (negative) and 1 (positive), which helped capture tone. The Flesch-Kincaid grade level was computed for each answer using the textstat library, providing an estimate of the readability of the response.

2.2 TF-IDF Vectorization

The Term Frequency-Inverse Document Frequency (TF-IDF) method was applied to the cleaned text data to transform the text into numerical representations (vectors) that capture the importance of words in relation to the entire dataset.

2.3 Data Scaling
Given that the features extracted from the text (e.g., word count, sentiment, text complexity) have different scales. This scaling operation standardizes the features, ensuring they have zero mean and unit variance, which is important to standardize the data.

3. Model Development

3.1 Decision Tree Classifier
A Decision Tree Classifier was chosen as one of the primary models due to its simplicity, interpretability, and ability to handle both numerical and categorical features. The model was trained using the features extracted from the text data, with the class labels (i.e., Answer Quality) as the target variable. To handle class imbalance, the class_weight='balanced' parameter was used to adjust weights inversely proportional to class frequencies.
Because Decision tree can handle imbalance, smote was not used on the data in this model.

3.2 Support Vector Machine (SVM)
A Support Vector Machine (SVM) classifier was also implemented to compare performance. SVM is known for its robustness in high-dimensional spaces and is particularly effective for text classification tasks. The SVM was trained using the scaled features and the TF-IDF vectorized data. Additionally, the Synthetic Minority Over-sampling Technique (SMOTE) was employed to address the class imbalance by oversampling the minority class in the training set.

3.3 Random Forest Classifier (for Feature Importance)
To complement the SVM, a Random Forest Classifier was also used to evaluate feature importance, to identify which features contributed most to the classification task.

4. Model Evaluation

4.1 Performance Metrics

- Accuracy: The percentage of correct predictions made by the model.
- Precision, Recall, and F1-Score: These metrics were computed for each class (good, average, poor) to assess the balance between false positives and false negatives, as well as the overall effectiveness of the classifier.
- Confusion Matrix: A confusion matrix was generated to visually inspect the number of correct and incorrect predictions across all classes.

6. Experiments:

Several experiments were conducted to explore the performance of various machine learning models and feature extraction techniques in predicting the quality of interview responses. The following experiments were carried out:

Baseline Experiment: Unprocessed Text Classification
The first experiment served as a baseline to understand the impact of basic text preprocessing and feature extraction on model performance. The baseline experiment provided an initial accuracy score and helped assess the raw performance of machine learning models on the unprocessed text.

Experiment 1: Impact of Preprocessing on Classification

This experiment aimed to determine the effect of various preprocessing steps on model accuracy. The preprocessing steps included: Stop word Removal, Text Normalization, Lemmatization

Experiment 2: Sentiment and Complexity Features
The next experiment involved incorporating sentiment analysis and text complexity features into the model. The following features were extracted:

- Sentiment: Using the TextBlob library, sentiment polarity was computed for each interview answer (ranging from -1 for negative to 1 for positive).
- Text Complexity: The Flesch-Kincaid grade level was calculated to measure the readability of each answer.

Experiment 3: Sentence Structure Variability

This experiment sought to determine the effect of sentence structure variability on model performance. The hypothesis was that answers with varied sentence structures might indicate higher-quality responses, and that this could serve as a useful feature for classification. Using SpaCy's POS tagging, the sentence structure variability (the number of unique parts of speech used) was computed for each answer.

Experiment 4: Class Imbalance Handling (SMOTE)

The effect of handling class imbalance was tested. The dataset had a disproportionate number of answers labeled as "good" compared to "average" and "poor". This imbalance could have negatively impacted model performance, especially for predicting the minority classes. To address this, the Synthetic Minority Over-sampling Technique (SMOTE) was applied to the training set to generate synthetic examples for the minority classes. The SVM model was trained on the resampled data and evaluated against a non-sampled test set. Performance was compared with and without SMOTE to assess its impact on model accuracy, especially for predicting the minority class (e.g., "poor" responses).

Experiment 5: Comparison of Classifiers

This experiment focused on comparing the performance of different classifiers on the same preprocessed and feature-engineered dataset. The classifiers used were:

Decision Tree Classifier
Support Vector Machine (SVM)
Random Forest Classifier
BERT Based Model

Each classifier was trained and tested using the same feature set, which included TF-IDF vectors, sentiment scores, text complexity, sentence structure variability, and word count. Performance metrics such as accuracy, precision, recall, and F1-score were compared across classifiers to determine the best performing model for this task.

Experiment 6: Feature Importance and Dimensionality Reduction

The feature importance was evaluated using a Random Forest Classifier. After training the model, the most important features were identified based on their contribution to the classification decision. Features with low importance were removed, and the model was retrained with the reduced feature set. Additionally, the impact of dimensionality reduction techniques, such as Principal Component Analysis (PCA), was explored to see if reducing the number of features while maintaining performance could improve model efficiency.

Experiment 7: Hyperparameter Tuning

Hyperparameter tuning was conducted to find the optimal settings for both the Decision Tree Classifier and the SVM model. Grid search and random search techniques were used to tune the following hyperparameters:

Experiment 8: Evaluation on Real-World Data

Lastly, the models were evaluated on a real-world interview dataset, collected from a sample of candidate responses to interview questions across different industries. This dataset had a more diverse range of answers, including complex, informal, and varying quality responses. The goal was to determine the generalizability of the trained models and assess their ability to classify responses in real-world settings.

7. Results

The performance of two machine learning models, **Support Vector Machine (SVM)** and **Decision Tree (DT)**, was evaluated on the task of predicting the quality of interview responses (labeled as "good", "average", or "poor"). The models were trained and tested on the preprocessed dataset with multiple features, including text-based features (TF-IDF), sentiment, complexity, word count, and sentence structure.

1. Support Vector Machine (SVM)

The *SVM model* was evaluated with the following results:

- *Accuracy*: 53.77%
- *Classification Report*:

Class	Precision	Recall	F1-Score	Support
-1 (Poor)	0.33	0.79	0.47	38
0 (Average)	0.18	0.40	0.25	62
1 (Good)	0.90	0.53	0.67	311
Accuracy	-	-	0.54	411
Macro Avg	0.47	0.58	0.46	411
Weighted Avg	0.74	0.54	0.59	411

The SVM model exhibited good precision for the "Good" class (1), but struggled with the "Poor" (-1) and "Average" (0) classes. Notably, the model's *recall for the "Poor" class* was high (0.79), indicating that it was able to identify most of the poor-quality answers, but it had low precision (0.33), resulting in many false positives. For the "Good" class, the model demonstrated high precision (0.90) but lower recall (0.53), showing a bias toward predicting "Good" answers.

2. Decision Tree (DT)

The *Decision Tree model* was evaluated with the following results:

- *Accuracy*: 68.0%
- *Classification Report*:

Class	Precision	Recall	F1-Score	Support
-1 (Poor)	0.41	0.53	0.46	38
0 (Average)	0.28	0.39	0.32	62
1 (Good)	0.85	0.76	0.80	311

Accuracy	-	-	0.68	411
Macro Avg	0.51	0.56	0.53	411
Weighted Avg	0.72	0.68	0.70	411

The *Decision Tree model* outperformed the SVM model in terms of overall *accuracy (68%), with particularly strong performance on the "Good" class (1), achieving an F1-score of 0.80. However, the model's **precision for the "Poor" class* was moderate (0.41), and it had a *lower recall* (0.53), indicating that it was not as effective at identifying poor-quality responses. The "Average" class suffered from low precision (0.28) and recall (0.39), which suggests that the Decision Tree had difficulty distinguishing between average and poor responses.

3. Comparative Analysis

- *SVM* showed a *bias toward the "Good" class*, with high precision for good-quality responses, but it struggled with the other two classes, especially in terms of recall.
- *Decision Tree, while having a higher overall accuracy, demonstrated stronger recall for the "Good" class and a **balanced approach* towards different classes. It still faced challenges with distinguishing between the "Poor" and "Average" responses.

8. Conclusion

The Decision Tree model performed better in terms of overall accuracy (68%) compared to the SVM model (53.77%), particularly in identifying high-quality responses (1). However, both models exhibited challenges with predicting lower-quality responses (0 and -1), with poor precision and recall for the "Average" and "Poor" classes. The next step could involve further optimization, including hyperparameter tuning, class balancing techniques, or trying other classifiers to improve the identification of average and poor-quality responses.

9. References:

[1] https://github.com/Anas436/Chatbot-for-Interview-Preparation

[2] Verrap, R., Nirjhar, E., Nenkova, A., & Chaspari, T. (2022, December). “Am I Answering My Job Interview Questions Right?”: A NLP Approach to Predict Degree of Explanation in Job Interview Responses. In Proceedings of the Second Workshop on NLP for Positive Impact (NLP4PI) (pp. 122-129).

[2] Srihari, P., Shivani, S., Nimmalapudi, T. L., Sirisha, P., & Pooja, T. (2022). Survey On Interview Performance Prediction and Analysis using Audio Features and NLP. *NeuroQuantology*, *20*(9), 6567.

[3] Purohit, J., Bagwe, A., Mehta, R., Mangaonkar, O., & George, E. (2019, March). Natural language processing based jaro-the interviewing chatbot. In 2019 3rd international conference on computing methodologies and communication (ICCMC) (pp. 134-136). IEEE.