International Journal of Computer Applications (0975 - 8887)
Volume 174 - No.14, January 2021

A Novel Approach towards Rating Free-Text Responses in Job Recruitment

Sarvesh Relekar
Data Analytics
Shortlist Professionals Pvt. Ltd.
Mumbai, India

Sayak Ray
Data Analytics
Shortlist Professionals Pvt. Ltd.
Kolkata, India

ABSTRACT

The use of chatbots has become mainstream in the field of staffing and recruitment. It has been observed that candidates tend to be much more at ease when interacting with chatbots. The responses provided by the candidates to the questions posed to them via chatbots are evaluated on the basis of various parameters by human evaluators who may have a subjective bias towards what defines a good response. This is especially the case, when it comes to evaluating free-text responses to open-ended questions that have little to no domain constraints. To overcome this hurdle involving human bias, we propose an alternate approach that utilizes modern techniques in the fields of Natural Language Processing and Deep Learning to develop an algorithm that rates free text responses in an impartial manner on the basis of the mood/sentiment expressed by the candidate, the grammatical accuracy of the answer and the relevance of the response w.r.t. the question asked while penalizing it for the presence of any negation or grammatical error, thus acting as a baseline model that aims to achieve the stated task. This algorithm thus sets a common standard towards what can be considered a good response thereby overcoming the hurdle arising from human perspective and establishing a criterion for evaluating free text responses.

Keywords

Automated Free-Text Grading, Natural Language Processing, Deep Learning, Data Science

1. INTRODUCTION

Shortlist as an organization has been trying to address the challenge of rating answers provided by potential candidates on the proprietary platform with respect to:

(1) The mood of the candidate while answering questions.

(2) The grammatical correctness and the sentence structures of the answer.

(3) The relevance of the answer w.r.t the question asked.

This helps Shortlist create acting guidelines for the Operations Team to reduce bias while reviewing answers, and make their job easier. Also, for Connect^{TM} jobs, this acts as a nice offering to clients while they review the profiles themselves.

1.1 Candidate Mood

When a potential candidate answers the questions relevant to the job for which they are applying for on the platform, the recruitment team is unaware of the prevailing environment that the candidate is in. For eg. the candidate may be tentative while answering certain questions, confident while answering some others. Some questions might bring them joy, while for answering some others, they have to put on their analytical hats.In order to give the recruitment team an additional flavor of how the candidate mentally perceives certain questions while answering, a technique to capture the mood of the candidate has been incorporated within the evaluation process.

1.2 Grammatical Correctness and Sentence Structure

When it comes to attending a job assessment, clear written communication skills are a necessary quality for all candidates. Written communication for Shortlist is a two-way process and the success of a candidate depends a lot on their abilities to effectively communicate their points to the evaluator. A grammar rating score will help the team choose the right candidate fit for the role.

1.3 Answer Relevance and Quality of Content

The strongest indicator for good fit for a particular role that the recruitment team aims for is how relevant the answers provided by the candidate are w.r.t the question asked. Generic answers might not help the team understand if the candidate has the requisite experience or not. A relevance checking algorithm sorts the answers of candidate to help avoid any bias from the Recruitment Operations team while reviewing answers while a content rating algorithm can help the reviewer gain an idea about how expressive the content is.

2. RELATED WORK

2.1 Text Similarity Calculation Methods

A variety of approaches have been tried and tested for the task of short answer or essay scoring. A majority of these approaches involved testing for similarity. One such case is discussed in [1] where a system is proposed that compares the input answer with several answers provided beforehand for reference and computes a similarity score on the basis of weights obtained from common words between the input answer and the reference answer. A similar approach is taken in [2] which uses a modified version of the BLEU algorithm for evaluation w.r.t the reference answers and gen-

erates a similarity score. An alternate method could be developing a Knowledge Base, as suggested in [3] that allows one to compute the various correlation measures between words. However, this approach lacks coverage of synonyms and may be inadequate for free text responses. A combination of two topic modelling algorithms for similarity, namely ERB, a modified version of the BLEU algorithm and Latent Semantic Analysis are used in unison and their combined results are used to evaluate students' free-text answers in [4]. Another comparative approach is discussed in [5] that involves the usage of knowledge based similarity measures utilising the WordNet implementation for word-to-word similarity metrics and their comparison against corpus-based measures such as LSA and Explicit Semantic Analysis (ESA) in which the dimensions of the vector are directly equivalent to abstract concepts.

2.2 Regression-based Evaluation Methods

A linear regression based approach is explained in [6] where various modules for syntactic, discourse and topic analyses provide a holistic score that focuses on the marking the writing skills of the author instead of the content. An enhanced version of this approach is explained in [7] by the name of C-rater that builds a canonical representation of a given response by extracting the underlying structure of the response, resolving pronoun reference, normalizing across inflected words and finally, recognizing the use of similar terms and synonyms.

2.3 Classification-based Evaluation Methods

A unique method is implemented in [8] where a Bayesian Network classifies a sample essay on the basis of Information Gain. The text itself has been subjected to basic preprocessing techniques such as stemming and stopword removal into three categories, namely Inappropriate, Partial and Appropriate. A similar ordinal scheme of classification of free text responses into three rating classes, namely low, medium and high has ben used for rating the content of a response in this paper. Another usage of Bayesian classifiers and their comparison with K-Nearest Neighbors classifiers was studied in [9]. These classifiers were combined with a variety of other text similarity measures and were observed toperform well. However, the KNN algorithm proved to be vastly inferior to Bayesian Classifiers unless it was paired with some other corpus based measure such as Latent Semantic Indexing.

2.4 Previously Attempted Approaches at Shortlist

Using previously shortlisted candidates' information (such as assessment score, question answers, boost count, job status and exclude count) against employers information (such as budget for the role, required experience, job title, industry, and sector) A Random Forest Model was developed that predicts the probability of a candidate to be shortlisted. The Random Forest model learns patterns from the data. Given the task to predict whether a candidate must be shortlisted or not, the model learns two major sets of patterns to meet the objective. A similarity based approach was also tested wherein a score between 0-1 is assigned for the relevance between the question posed to the employer and the candidate's response. The distance metrics used to quantify relevance included Euclidean distance (shortest distance between two points), Levenshtein distance (number of insertions and deletions required to change one text to another) and Cosine Similarity. The predominant challenges faced while using these approaches were:

(1) The lack of annotations to define a good answer.

(2) The distance metrics were bidirectional, but were not robust enough to account for open ended free flowing english text.

3. PROPOSED APPROACH

3.1 Candidate Mood Analysis

The mood of the candidate expressed from the answer can tell us about their thought process. It can be used to infer the:

(1) Level of confidence a candidate has while writing an answer.

(2) Level of knowledge they have about that field. (job description in this case)

(3) Analytical skills of the candidate.

The IBM Watson Tone Analyzer [10], a third party API has been used to detect the various tones present within the response. It is a service that analyses the tone of the input content. The IBM Watson Tone Analyzer is a part of IBM Watson, IBM's own platform to provide cloud based services to users. It helps in accelerating businesses using AI based solutions. It allows the user, a monthly quota of 2,500 API calls at no cost and has separate pricing lists for premium user plans. Any tone that gets a score of 0.5 or greater within a text document is considered within the final output by the tone analyzer. The final output is a JSON file that comprises both the tones reflected from the document as well as the tones deduced from each of the individual sentences.
The Tone Analyzer leverages cognitive linguistic analysis to identify a variety of tones at both the sentence and document level. It detects 3 types of tones, namely:

(1) **emotion** *(anger, disgust, fear, joy and sadness)*

(2) **social propensities** *(openness, conscientiousness, extroversion, agreeableness, and emotional range)*

(3) **language styles** *(analytical, confident and tentative)*

3.2 Grammatical Correctness & Sentence Structure

Grammatical accuracy is evaluated in two phases; In the first phase, each sentence in the response is split into tokens (words) and every token is evaluated for the accuracy of its spelling and a count of incorrectly spelled words is maintained. In the second phase, Grammar Bot [11], a third-party API is used to evaluate the grammatical accuracy of a response.

3.2.1 Spelling Checker Module

A spelling checker algorithm that utilises Peter Norvig's Spell Checker [12] is used to check the accuracy of spellings. This algorithm, based on the Levenshtein Distance metric is used to find permutations within a minimum Edit distance of 1 and maximum of 2 from the original word. It then compares all permutations (insertions, deletions, replacements, and transpositions) to known words in a word frequency list. Those words that are found more often in the frequency list are more likely the correct results.
This module takes a preprocessed free text response as input. The steps involved in preprocessing include:

(1) Tokenization on whitespaces.

(2) Punctuation removal from list of tokens.

(3) Removal of tokens that begin with capital letters.

A variable that stores the count of incorrect spellings is initialized to zero is maintained to store the count of incorrectly spelled words. Each word is passed to the module for spell checking and the

counter is incremented for each incorrect spelling. A list of suggestions is generated if the word is incorrect. If the list of suggestions has a positive number of items, then the counter is incremented by one. Once every word has been evaluated, a percentage value of spelling accuracy is calculated using the following formula:

$$Percentage = 100 \times \left(1 - \frac{Count\ of\ Errors}{Number\ of\ Tokens}\right) \quad (1)$$

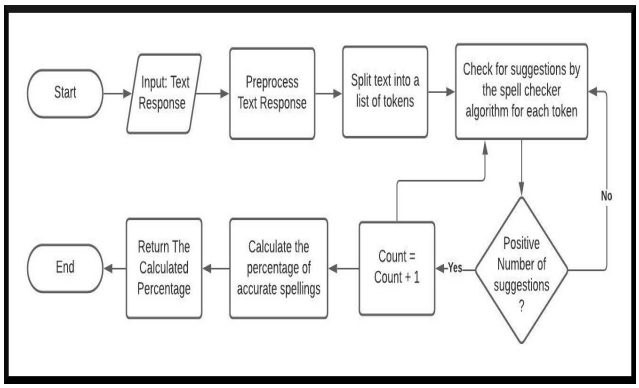

Fig. 1. Flowchart for the Spelling Checker Module

3.2.2 Gramatical Correctness Module

The task involving the evaluation of the grammatical accuracy of a response is performed using the GrammarBot API, another third party API that evaluates the sentence for its grammatical accuracy by returning a number of matches in terms of the grammatical mistakes occurring in the sentence. The GrammarBot API allows a quota of 4,500 API calls per month and has a variety of other pricing plans available for users as per their requirements. It is currently hosted on the RapidAPI Platform as a semi-premium API service. The module takes a raw free text response as input. No preprocessing is required. Similar to the Spelling Checker Module, a counter initialized to zero is maintained to store the count of matches generated by the GrammarBot API for each response. Upon making the API call, A JSON response containing the matches of grammatical errors within the given text is generated and their count is stored. This count is used to generate the percentage of grammatical accuracy of the response calculated using the formula given below:

$$Percentage = 100 \times \left(1 - \frac{Count\ of\ Errors}{Number\ of\ Tokens}\right) \quad (2)$$

3.3 Relevance of the Answer & Quality of Content

The relevance of a response w.r.t the job details and the question asked as well as the quality of the content within the response are among the most important parameters evaluated by this system. The task of calculating the relevance of the response w.r.t the question asked and the quality of the content is split into two major modules that perform both the tasks independently. Another minor module named as the Positivity Factor Generation Module is used to generate a value that offsets some erroneous ratings assigned by the Content Rating Module.

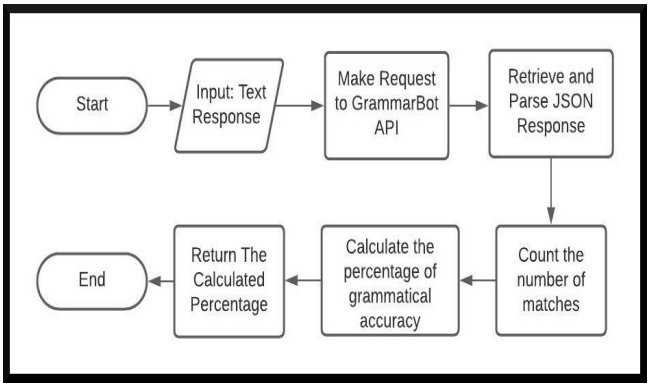

Fig. 2. Flowchart for Grammatical Correctness Module

3.4 Relevance Checking Module

The Relevance Checking Module calculates the Cosine Similarity between the concatenation of the question text & relevant job details w.r.t the free text response provided for the given question by a particular candidate.

3.4.1 Data Set

The data set used for this task is Shortlist's own private data comprising of fields such as **Job description** & **Job Summary** and the **Must-haves** for the given job. Each job has a flow of open ended questions associated with them. There is also another feature named **Job Card (Body)** that dictates the essential skills or qualities required from a candidate that is associated with each job. The data set contains 65,926 unique responses from 28,467 candidates. This data set is also used as our Test data set for the Content Rating Module explained further in the paper. Another data set containing 92 responses annotated by the Operations Team is used for calculating the relevance of the answer as well as the quality of content expressed in it.

3.4.2 Data Preprocessing

Data preprocessing for this module involves the execution of the following steps in the specified order:

(1) Replacement of UTF-8 encodings with corresponding punctuation or miscellaneous characters.
(2) Removal of any UTF-8 encoded characters that cannot be replaced.
(3) Separation of camelcased words into separate words at the capital letters within such words.
(4) Replacement of certain punctuation marks ('.', '!', ',', '?') with whitespaces.
(5) Removal of extra whitespaces.
(6) Tokenization and removal of any garbage characters from the list of tokens
(7) Lemmatization of each token.
(8) Concatenate tokens with whitespaces for separation

Shortlist has a unique method of maintaining job details. Job details comprise of four main components namely:

(1) **Job Description**
(2) **Job Summary**
(3) **Job Card (Body) Text**

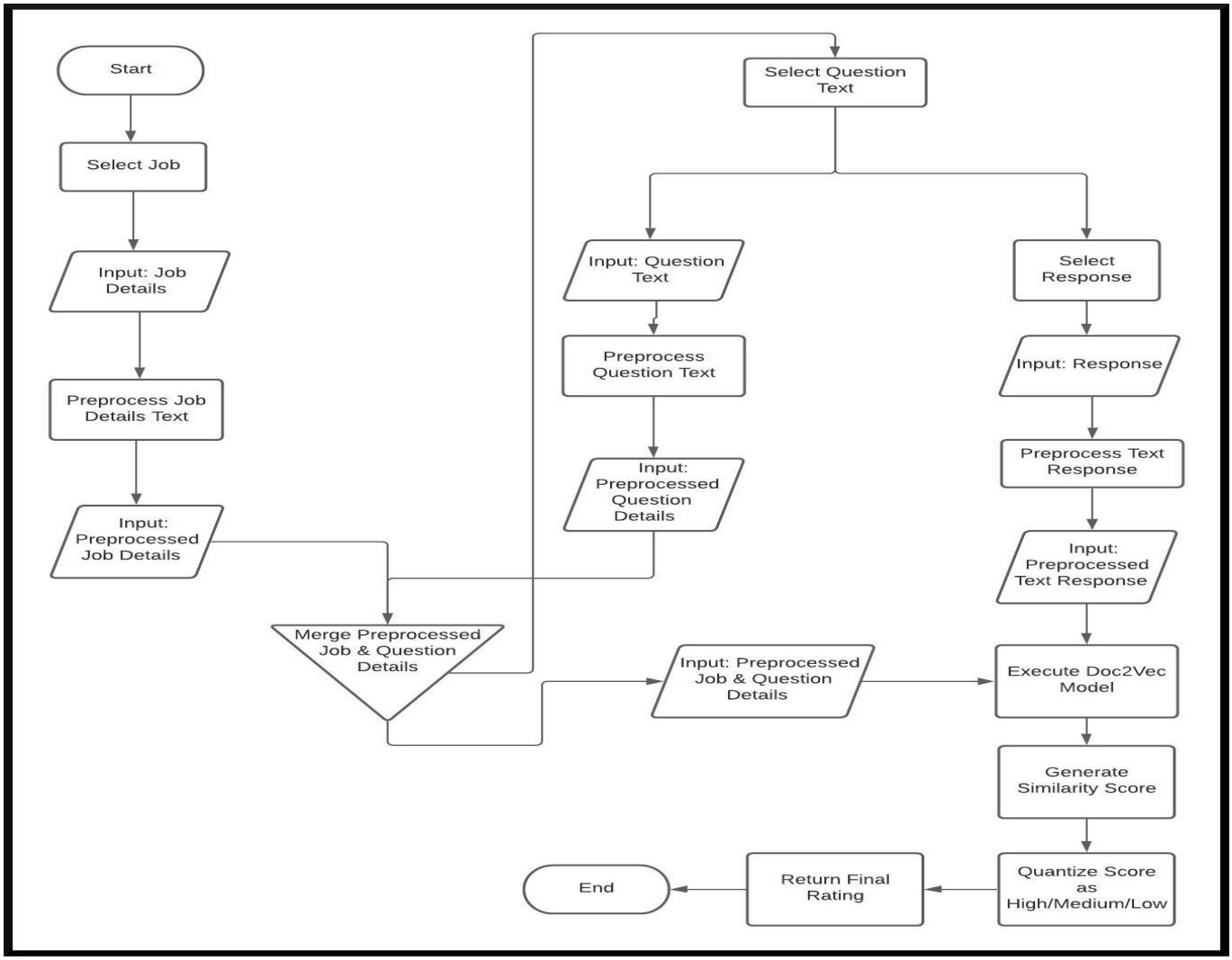

Fig. 3. Flowchart for the Relevance Checking Module

(4) **Job Must-Haves**

Job Description, **Job Summary** and **Job Card (Body) Text** describe the details about the company under which the job is enlisted and provide the details of the job position and other details such as roles and responsibilities, salary offered, working hours, etc.
Must-haves are the skills and requirements for that particular job. These are usually keywords that occur on the job pages displayed to the candidates so as to imply the skills which the employer is looking for to perform a certain job. These four components as a group will be further referred to as **Job Details**.
Unique **Job Details** are extracted and tagged separately, since several responses are provided for the same job description by different candidates. Since all the responses are unique, there is no need to perform any kind of extraction and can be assigned unique tags directly. The tagging procedure is performed for the Doc2Vec model that requires each unique document to be tagged or assigned with a unique identification number before the training process. An additional data set containing 92 annotated responses is used for the purpose of validation in the Relevance Checking Module and as a part of the training data in the Content Rating Module.

3.4.3 Evaluating Relevance using Doc2Vec .

Doc2Vec [13] aims to create a numeric representation of a document, regardless of its length. But unlike words, documents do not come in logical structures such as words, hence, the need to use another method arises. The solution implemented was to use the basic Word2Vec [14] model and an additional Paragraph ID Vector that remains unique for each document. The Distributed Memory version of Paragraph Vector (PV-DM) has been used to perform this task. It acts as a temporary memory that remembers missing fetures from the current context, or in brief, serves as the topic for the given text. While the word vectors are used to represent the concept of a word, the document vector is used to represent the concept of a document.
A Doc2Vec model is trained on these preprocessed **Job Details** and responses. Once the model training is complete, similarity scores are generated by calculating the cosine similarity between each

response and its corresponding job detail. Further, the Box Cox transformation is applied to standardize the similarity scores generated. These standardized similarity scores are mapped into bins (low, medium and high) and the boundary scores used to define each bin are recorded.

3.5 Content Rating Module

A Deep Learning based approach has been developed in this module to classify responses as *low*, *medium* or *high*. The techniques used for this purpose include Recurrent Neural Networks combined with pre-trained word embeddings such as GLoVe[15] (Global Vectors for Word Representation).

3.5.1 Data Set

The data set provided for the Kaggle competition Hewlett Foundation: Automated Essay Scoring [16] that comprises of 12,976 essays rated with a score between 1 to 100 is used to classify responses. Shortlist's own annnotated data set comprising of 92 responses was appended to this data, increasing the total to 13,068 tuples. The data set of 65,926 responses introduced previously in the Relevance Checking Module is used as our Test data set. Pretrained GLoVe Embeddings (840 Billion Words, 300 Dimensions) are used as the word embeddings for this task.

3.5.2 Data Preprocessing

Data preprocessing involved in this module is slightly diferent than that of the Relevance Checking Module. In this case, one major difference involves the inclusion of punctuation. Other changes include the expansion of contractions and the correction of wrongly spelled words. The free text responses are not subjected to stemming and lemmatization as the data set used in this module is not domain-specific [17]. The steps followed for data preprocessing in the prescribed order include:

(1) Replacement of UTF-8 encoded characters with their corresponding punctuation marks or miscellaneous characters.
(2) Removal of any UTF-8 encoded characters that cannot be replaced.
(3) Expansion of contractions.(for eg. *hasn't* expanded as *has not*)
(4) Replacement of multiple instances of same punctuation mark with only a single one of the same type.
(5) Insertion of single whitespace between words and certain types of punctuation.
(6) Removal of extra whitespaces.
(7) Tokenization and removal of any garbage characters from the list of tokens
(8) Replacement of typically misspelled words with their correct spellings by comparing with the GLoVe word vocabulary.
(9) Replacement of 2-digit or greater numerical values with '#'.
(10) Tokenization over whitespaces.
(11) Conversion of each list of tokens into sequence of integers generated by the Tokenizer.
(12) Post-padding of sequences.

The steps involving the insertion of whitespaces between words and punctuation, the expansion of contractions, the replacement of typically misspelled words with their correct spellings and the replacement of numbers with '#' contribute towards improving the coverage of GLoVe word embeddings for the data set. The step involving the insertion of whitespaces allows the Tokenizer to treat punctuation as separate tokens as certain punctuation marks are included in the list of tokens covered within the GLoVe Embeddings. The Tokenizer generates a set of key value pairs with the word as the key and the assigned integer as its value. No word is assigned the index 0 as it is used for the post-padding step to distinguish the padded elements from the sequence boundary. Also, for the post padding step, a maximum sequence length of 171 elements was chosen, i.e. any sequence having length less than 171 will be padded with zeros to bring its length to 171 whereas any sequence that has length greater than 171 will be truncated to 171 elements in length.
The target scores present in the training and testing data set were continuous numeric values. So the Box Cox Transformation [18] was applied to the scores in order to get a distribution that better resembles the Standard Normal Distribution. Finally these scores are mapped into *low*, *medium* and *high* bins by setting percentile boundaries at 50 percentile between low and medium and 70 percentile between medium and high classes. This simplified the problem from regression to classification. The classes (*low*, *medium* and *high*) represent the three target classes.

3.5.3 Methodology for Content Rating

Initially, A Word2Vec model was trained over the preprocessed text corpora in the Training and Testing data sets. A longer window size of 11 was chosen in order to generate embeddings that provide better topical relation between words [19]. The embeddings generated by the Word2Vec Model as well as the GLoVe embeddings are loaded into the memory at the same time to create two separate embedding matrices that are used in the Embedding layers of the Recurrent Neural Networks trained for the task of classification in this module. Two RNNs [20] that utilize the GLoVe and Custom Word2Vec embeddings respectively are trained for 10 epochs. The data set is split into training and validation sets comprising of 75% (9,801 responses) and 25% (3,267 responses) of the data respectively.
The RNN model architecture comprises of an Embedding layer followed by a 1-dimensional Spatial Dropout Layer. It was followed by a single LSTM layer. The output of layer was normalized batchwise and excessive parameters were subjected once again to a Dropout layer. A Dense layer with three output modes to generate the probabilities of three classes acted as the output layer. Categorical Crossentropy was used as the loss function and the Adam optimizer was used to optimize the loss value at each epoch.

3.6 Positivity Factor Evaluation

Detecting negation within a free text response is a complicated task due to the presence of negation within the text in various forms such as negative words and contractions. Another issue that arises indirectly in the detection of contractions that suggest negation such as can't, hasn't, etc. is that when the candidates use a variety of electronic devices such as cellphones, desktops, laptops, tablets, etc. there is a very high chance of the apostrophe(''') not being entered due to human error that occurs while typing and if there is no autocorrect functionality available in the device to correct the same. The algorithm proposed to tackle this issue performs the detection of negative words and original negative contractions as well as their corresponding erroneous forms of negative contractions with the missing apostrophe(?). The positivity factor is calculated using the following formula:

$$Positivity\ Factor = 2^{-Number\ of\ Penalties} \quad (3)$$

3.6.1 Data Preprocessing

The steps followed for text preprocessing are:

(1) Text Conversion to lowercase
(2) Removal of punctuation marks such as '(', ')', ',' (comma) and '.' (period).
(3) Removal of '-' from text.

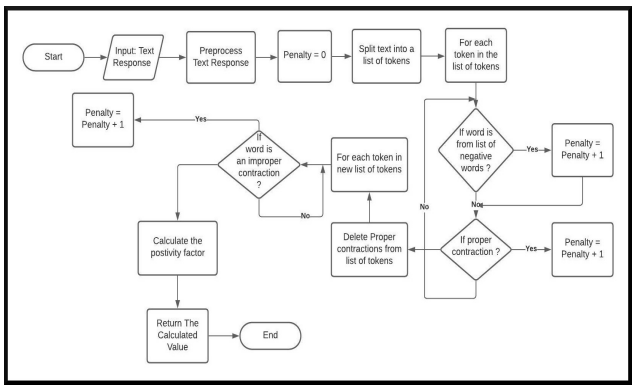

Fig. 4. Flowchart for Positivity Factor Module

4. RESULTS AND EVALUATION

4.1 Candidate Mood Analysis

The tone of the candidate was evaluated over each of the responses as well as over the aggregate of all the responses provided by a particular candidate. It was observed that a majority of candidates' responses displayed a **Joyous** tone and **Analytical**, **Tentative** language styles. Few responses that clearly displayed negation, for eg. responses stating that the candidate did not have any experience in some field were tagged with a **Sadness** emotion.

Table 1. Tone Analysis of Responses for a single Job Detail

Sr. No.	Question Text ID	Response ID	Document Tones	Sentence Tones
1	1	1	Joy - 0.550785, Analytical - 0.546148	Empty - 0
2	1	2	Analytical - 0.694442	Analytical - 0.73677
3	2	1	Analytical - 0.639296	Analytical - 0.687768, Analytical - 0.560098, Tentative - 0.5538, Analytical - 0.636458
4	2	2	Analytical - 0.699597, Confident - 0.774455	Analytical - 0.802152, Confident - 0.874372

4.2 Grammatical Correctness & Sentence Structure

It was observed that a majority of the responses had correctly spelled words and were devoid of major grammatical mistakes. The range of gramatical accuracy of the responses varied from 90% to 100%. Spelling accuracy had a slightly better overall range, varying from 92% to 100% per response with a select few responses having accuracy as low as 85%.

Table 2. Grammatical Correctness & Sentence Structure Analysis of Responses for a single Job Detail

Sr. No.	Question Text ID	Response ID	Grammatical Accuracy	Spelling Accuracy
1	1	1	98.13%	87.88%
2	1	2	100%	92.6%
3	2	1	95.42%	95.24%
4	2	2	94.88%	95.21%

4.3 Relevance Checking using Doc2Vec

Cosine Similarity was chosen as the initial metric to evaluate the relevance of the response w.r.t the job details and question text. It was observed that the resulting distribution was negatively skewed and thus, converted to a form resembling the standard normal distribution.

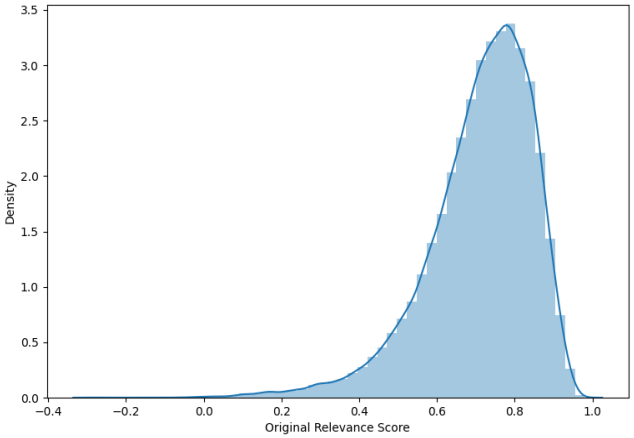

Fig. 5. Distribution of Relevance Scores

The Box Cox Transformation was applied in the next step to this distribution in order to standardize the scores.

$$x_{\lambda}^{'} = \frac{x^{\lambda} - 1}{\lambda} \tag{4}$$

Upon applying the Box Cox transformation on the raw distribution, it was observed to resemble the standard normal distribution more closely. At this point, the distribution scores are mapped into three bins/classes namely *low*, *medium* and *high* by selecting certain threshold values.
A similar procedure was then performed on the validation data set comprising of 92 responses to obtain the classes for those responses. The effectiveness of the module was judged on the basis of

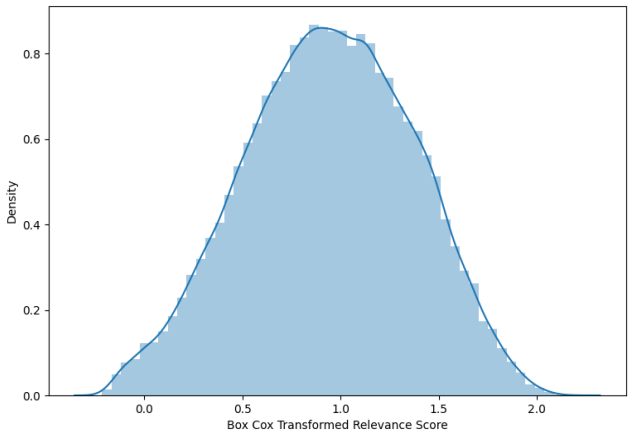

Fig. 6. Distribution of Relevance Scores after Box Cox Transformation

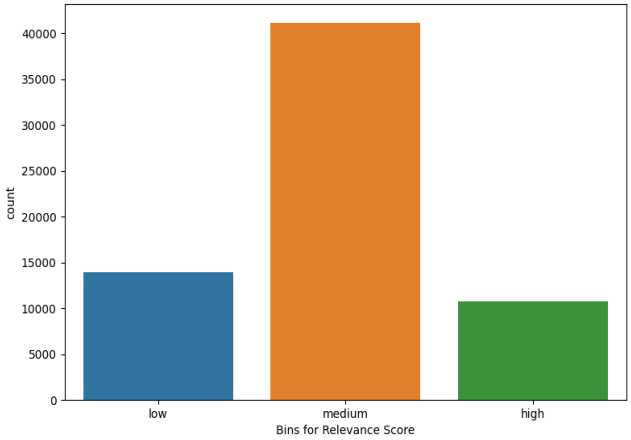

Fig. 7. Mapping of Transformed Scores into Bins

two metrics, the first of which was accuracy. An accuracy of **48%** was obtained upon comparing the classes. The second metric used was a custom metric called **Maximum Relevance Factor (MRF)** that checks if the class of the response predicted by the model is greater than or equal to the class of the annotated response. This was in accordance to the business case wherein the possibility of the inclusion of a **false positive**, the case in which a mediocre response being assigned a higher relevance rating by the model is accepted but the main aim of minimizing the cases involving **false negatives** that result in a highly relevant response being rejected is achieved.

$$MRF = \frac{Count\ of\ Responses\ Rated\ Equal\ or\ Higher}{Total\ Number\ of\ Responses} \quad (5)$$

An MRF of **0.75** is observed for the given responses which indicates a higher acceptance rate of the candidates' responses by the model.

4.4 Content Rating using Deep Learning

The RNN that utilises GLoVe Embeddings delivers an accuracy of 93.85% on the training data set and 91.26% on the validation data set. The other RNN that makes use of the embeddings generated by the Word2Vec model trained on our corpora results in an accuracy of 97.36% on the Training data set and 91.09% on the Validation data set. It was observed that the first RNN using GLoVe embeddings demonstrated higher accuracy when rating responses belonging to the *low* class whereas the one using the embeddings from the Word2Vec model rated responses belonging to the remaining classes (*high, medium*) more efficiently. Upon observing the responses, their original ratings and their corresponding ratings predicted by both the models, it was decided to take an ensemble of the ratings generated by both the models by selecting the highest possible class among the predictions made by both for each response, for eg. if one model predicts a *low* class and the other predicts *medium* for a given response, the *medium* rating is chosen as the final class for that particular response. This ensemble of outputs results in an increased accuracy of 92.39% on the Validation data, nearly a 1% increase in comparison to both the models.

Table 3. Model Accuracy Values

Sr. No.	Word Embedding	Training	Validation
1	GLoVe	93.85%	91.86%
2	Custom Word2Vec	97.36%	91.09%

4.5 Positivity Factor Generation

It was observed that a few well formed responses with high gramatical and spelling accuracy, but with the obvious presence of negative words or phrases were classifed to the *medium* class instead of the *low* class by the Content Rating Module. In order to prevent this from heavily influencing any future evaluation, the Positivity Factor Generation Module detects the presence of negative words, properly framed contractions and even improper or misspelled contractions and returns a positivity factor that can be used to gauge the overall quality of the content of the response.

Table 4. Relevance Checking, Content Rating and PF Evaluation

Sr. No.	Question Text ID	Response ID	OPS Rating	Relevance Class	Content Rating	PF
1	1	1	medium	medium	medium	1
2	1	2	medium	high	medium	0.07
3	2	1	high	medium	high	1
4	2	2	high	high	high	1

5. CONCLUSION & FUTURE SCOPE

In this paper, a distinct approach has been proposed to evaluate candidates for the purpose of recruitment on the basis of the free text answers provided by them for open-ended questions on the Shortlist Portal. A third party Tone Analyzer API was used to understand the candidate's mood while responding to the questions asked during the evaluation and can thus be used to develop an idea of the candidate's psych profile. Each of the candidates' responses was also evaluated on the basis of their grammatical accuracy and the structure of the framed sentences. The responses were also checked for their relevance w.r.t the job description that was associated with the question asked using Doc2Vec. Lastly, a Deep Learning model

in the form of a Recurrent Neural Network was used to classify the content of the responses to be of low, medium or high quality. The overall quality of evaluation by the Content Rating module was improved by introducing a Positivity Factor that penalizes well formed responses with presence of negation in their context. This system establishes a baseline towards what can be defined as a recommendation system that recommends candidates to companies based upon the parameters of assessment specified in this paper. This system can be improved by optimizing the spell checking algorithm used currently or implementing a spell checking and correction algorithm with better time complexity. The model architecture can also be redefined to include Convolution layers that may help to provide a better result by analysing the text sequence for bigrams and trigrams. The autonomous modules within this system can also be improved further by implementing multiple variations in the data preprocessing techniques used and choosing the best possible combinations among those that are tested.

6. REFERENCES

[1] Rodrigues, F., Arajo, L. (2012). Automatic Assessment of Short Free Text Answers. 4th International Conference on Computer Supported Education. 2.

[2] Noorbehbahani, F., Kardan, A. (2011). The automatic assessment of free text answers using a modified BLEU algorithm. Computers & Education. 56. 337-345. 10.1016/j.compedu.2010.07.013.

[3] Chakraborty, U., Das, S. (2015). Automatic Free Text Answer Evaluation using Knowledge Network. IJCA. 117. 10.5120/20532-2876.

[4] Perez, D., Gliozzo, A., Strapparava, C., Alfonseca, E., Rodriguez, P., Magnini, B. (2005). In automatic assessment of students? free-text answers underpinned by the combination of a BLEU-inspired algorithm and latent semantic analysis. In: Proceedings of the 18th international Florida artificial intelligence research society conference (FLAIRS?05) (pp. 358?362).

[5] Mohler, M., Mihalcea, R., (2009). Text-to-Text Semantic Similarity for Automatic Short Answer Grading. Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)

[6] Burstein, J., Leacock, C., Swartz, R. (2001) Automated Evaluation Of Essays And Short Answers. Fifth International Computer Assisted Assessment Conference Loughborough University 2nd and 3rd July 2001.

[7] Leacock, C., Chodorow, M. C-rater: Automated Scoring of Short-Answer Questions. Computers and the Humanities 37, 389?405 (2003). https://doi.org/10.1023/A:1025779619903

[8] Rudner, L.M., Liang, T. (2002). Automated essay scoring using Bayes? Theorem. The Journal of Technology, Learning and Assessment, 1(2), pp. 3-21.

[9] Larkey, L.S. (1998). Automatic essay grading using text categorization techniques. In: Proceedings of the 21st annual international ACM SIGIR conference on research and development in information retrieval (pp. 90?95).

[10] IBM Watson Platform, "https://www.ibm.com/watson/"

[11] GrammarBot: Grammar Check API, "https://www.grammarbot.io/"

[12] How to Write a Spelling Corrector by Peter Norvig, "https://norvig.com/spell-correct.html"

[13] Quoc, L., Mikolov, T. (2014). Distributed Representations of Sentences and Documents. 31st International Conference on Machine Learning, ICML 2014. 4.

[14] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. Advances in Neural Information Processing Systems. 26.

[15] Pennington, J., Socher, R., Manning, C. (2014). GloVe: Global Vectors for Word Representation.

[16] Hewlett Foundation: Automated Essay Scoring, "https://www.kaggle.com/c/asap-aes"

[17] Camacho-Collados, J., Pilevar, L.M. (2017). On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis.

[18] Bickel, P., Doksum, K. (1981). An Analysis of Transformation Revisited. Journal of the American Statistical Association. 76. 10.2307/2287831.

[19] Jurafsky, D, Martin, J.H., (2009), Speech and Language Processing (2nd Edition). Prentice-Hall, Inc., USA.

[20] Sherstinsky, Alex. (2020). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. Physica D: Nonlinear Phenomena. 404. 132306. 10.1016/j.physd.2019.132306.

[21] Glasgow, B., Mandell, A., Binney D., Lila, G., Fisher, D. (1997). MITA : An Information Extraction Approach to Analysis of Free-Form Text in Life Insurance Applications. Innovative Applications of Artificial Intelligence, Providence, RI, USA, July 27-31, 1997.

[22] Contreras, J.O., Hilles S., Abubakar, Z. B. (2018). Automated Essay Scoring with Ontology based on Text Mining and NLTK tools, International Conference on Smart Computing and Electronic Enterprise (ICSCEE), Shah Alam, pp. 1-6, doi: 10.1109/ICSCEE.2018.8538399.

[23] Shah, C., Pomerantz J., (2010). Evaluating and Predicting Answer Quality in Community QA. 411-418. 10.1145/1835449.1835518.

[24] Mitchell, T., Russell, P., Broomhead, Aldridge, N. (2002). Towards robust computerised marking of free-text responses.

[25] Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., Gao, J. (2020). Deep Learning Based Text Classification: A Comprehensive Review. ArXiv, abs/2004.03705.