Subjective Answer Evaluation using Machine Learning

Sitanshu Kushwaha[1] Nisha Ladhe[2] Snovia Gonsalves[3]
[1,2,3]Department of Computer Engineering
[1,2,3]St. Francis Institute of Technology, Mumbai, India

Abstract— The subjective answer evaluation using machine learning is not a new idea and has been in consideration for quite some time now. In this paper, by critiquing various methodologies used for this task in the past we are identifying the challenges of the problem at hand. Thus, we would like to propose a solution that uses Prediction based Word Embedding and try various similarity algorithms to evaluate the answer based on its semantic rather than just keywords.

Keywords: Machine Learning, NLP, Word Embeddings, Text Similarity

I. Introduction

Over the last few years, we have seen a significant rise in online training and also shifting of classroom-based training to online platforms. To check students' understanding of the course there is a need for online evaluation. But usually, the number of students enrolling in a particular course is much higher than they can be handled by a single instructor or a small group of instructors. So, the primary choice for evaluation is MCQs or objective questions, which can be automatically checked by online systems without the explicit need for human supervision. However, there is a limit to which these objective questions can evaluate one's understanding of the subject. Hence, there is a need for a subjective examination.

When human beings evaluate anything, the quality of evaluation may vary along with the emotions of the person and could be biased towards a few students. Evaluation of Subjective Answers for technical subjects involves a lot of time and effort of the evaluator. Evaluating using computer intelligent techniques ensures uniformity in marking as the same inference mechanism is used for all the students. Subjective answers have various parameters upon which they can be evaluated such as the question specific content and writing style.

II. Literature survey

While working with text data some of the most common practices in machine learning are Tokenization, Normalization, Stemming, Lemmatization, removing stop words, part of speech tagging, etc. These could be a good or a bad preprocessing step as per our application needs. For grading students' answers, a naive approach would be to extract keywords from the teacher's answer and map it with a student's answer[1] using cosine similarity[1], Jaccard Similarity[2], etc. to improve the accuracy of such naive evaluator, grammar could be checked for well-formed sentences[1]. However, the downside to such an approach is that students often tend to write answers in their own words and not necessarily use the same keywords as the teacher but might use synonyms of the keywords. So in practice, such an evaluator would fail to grade answers accurately.

A. *Text Similarity*

For evaluating a subjective answer concerning an ideal sample answer, the most important concept is text similarity. Most of the algorithms like K-means[3], Cosine similarity[4], Jensen-Shannon distance, Siamese Manhattan LSTM, etc. that could be used for text similarity works with numerical values and not directly text. There are multiple ways to represent words in the text as vectors this is known as Vectorization. One-Hot encoding, Distributed Representation Of Words, SVD, TF-IDF vector, Continuous Bag of Word, Skip-gram Model, etc. are a few of the widely used vectorization and word embedding techniques. Vector size in One-Hot encoding grows with vocabulary size. hence, making it computationally infeasible for a large corpus.

B. *Word Embeddings*

Machine Learning algorithms and almost all deep learning architectures are incapable of processing strings and hence rely on numbers. There are 2 main categories of word embeddings which are frequency-based embedding and prediction-based embedding.

1) *Frequency-based Word Embeddings*

Frequency-based embedding is further classified into count vector, TF-IDF vector, Co-Occurrence vector.

Firstly, a count vector matrix is formed by considering unique words and forming a matrix. Hence it helps in finding the most frequent word which must be present in both the documents. However, this approach falls short when we have millions of words in a document as the execution time will be higher.

Secondly, in TF-IDF it takes into consideration the entire corpus. It also considers words other than unique words that are ignored in the count vector method. TF-IDF lowers the weight of common words such as "the", "is", "a" etc and the frequency is higher for other unique words. However, TF-IDF gives importance to the rareness of a term but ignores the usefulness of the term.[5]

Furthermore, the method which is the co-occurrence matrix finds if similar words that occur together have similar contexts. It considers a window size and finds the occurrence of words throughout the document. The values are filled in a matrix. Using various decomposition techniques such as PCA, SVD etc new matrix is formed. However, this approach requires a huge memory and factorizing of a matrix is difficult[6]

2) *Prediction based vectors*

These methods were a prediction based on the sense that they provided probabilities to the words and proved to be state of the art for tasks like word analogies and word similarities. Word2vec model uses NLP technology to generate word vectors. Word2vec is a combination of 2 algorithms which are CBOW(continuous bag of words)[7] and the Skip-gram model[8].

CBOW (Continuous bag of words) can be considered as learning word embeddings by training a model to predict a word given its context. Skip-gram is the opposite, learning word embeddings by training a model to predict context given a word. Both models are built using 3 layers neural network with an input layer, hidden layer, and output layer. Skip-gram: works well with a small amount of the training data, represents well even rare words or phrases. CBOW: several times faster to train than the skip-gram, slightly better accuracy for the frequent words.

Count based methods calculate the co-occurrence matrix for all words, hence they tend to consume a lot of memory compared to the predictive models. Then the dimensionality reduction is applied to the large matrix which lowers the dimensions, which in the process makes the model more robust because it captures the most significant information while losing less significant information or noise. However at the expense of higher memory. With a prediction-based model, one of the advantages is that they consume less memory since they don't have to compute large co-occurrence matrices.

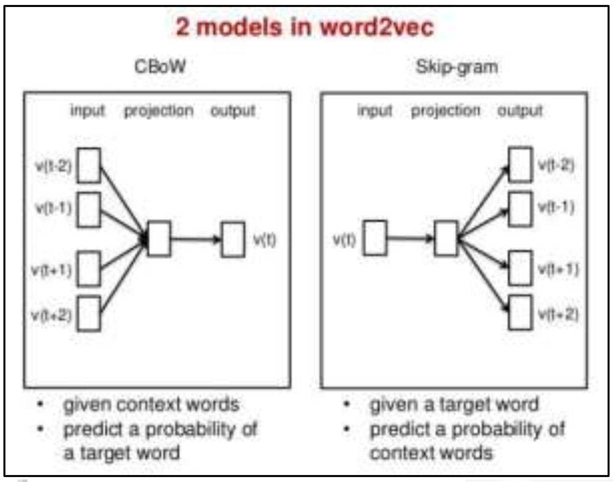

Fig. 1: CBoW and Skip-gram

C. Word Embeddings + Siamese BLSTM model

Siamese networks have two or more identical sub-networks in them[10]. It performs well on similarity tasks and is used in tasks like sentence semantic similarity recognizing forged signature and many more. There are many BLSTM models available. But the main focus is on character level BLSTM. The reason for character level BLSTM is 1) deals with spelling and unknown words.

1) *Morphology, compound words.*

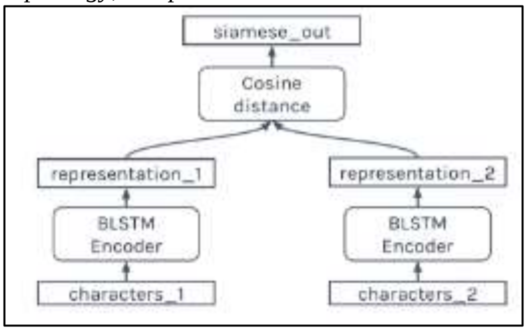

Fig. 2: Siamese BLSTM Model

The main idea of the Siamese model is that we take that representation on parse string and then we took another layer on top which is like a view layer that calculates the cosine distance. If the sample is positive then we get minimum cosine distance ie more similar to the words are.

D. Word Embeddings + LDA + Jensen-Shannon distance

1) Latent Dirichlet Allocation(LDA)

LDA is an unsupervised generative model that assigns a topic distribution to documents[9]. The model assumes that every document consists of several topics and hence, there is a topic overlap within the document. The number of topics should be specified in advance to the model. The document is clustered into the specified number of clusters and word distribution for each topic is obtained without explicitly conveying it to the model. Hence the model generates latent variables 1) A distribution of topics over a document. 2) Distribution over words for each topic.

2) Jensen-Shannon distance

Once we have a topic distribution for a new unseen document. The goal is to find the similarity between the teacher and student documents. Jensen-Shannon distance can be used to assess the similarity of these two documents as it is used to find the similarity between two probability distributions.

III. CHALLENGES IDENTIFIED

The challenges identified are as follows:

A. Keywords matching

After keywords extraction from the answers, matching the keywords between teachers' and students' answers efficiently is one of the main challenges of the project.

B. Vectorization

Choosing an appropriate model for Vectorization can be tricky as each of those works best for a particular setup of the system.

C. Curse of dimensionality

Vectorization for a large corpus can lead to the very high dimensionality of the vectors. this could lead to a very slow system with high space requirements.

IV. PROBLEM DEFINITION

In the existing systems, the occurrence and frequency of keywords are given the highest priority for the evaluation and grading of students' answers. The answer can be grammatically correct and can have all the necessary keywords present, but that does not mean it has the same meaning as the teacher's model answer. That is why we should check how similar is the meaning of the student's answer to that of the teacher's.

V. PROPOSED SYSTEM METHODOLOGY

A. Algorithm

Training the model to be used for Text Similarity:
Step 1: Vectorization
Step 2: Word Embedding
Text Similarity:
Step 1: Tokenize sentences from the answers.
Step 2: Vectorize words in the Sentences.
Step 3: Used Trained word Embeddings to find similarities between teacher and student answers.

Step 4: Grade based on the percent of similarity obtained.

B. Block diagram

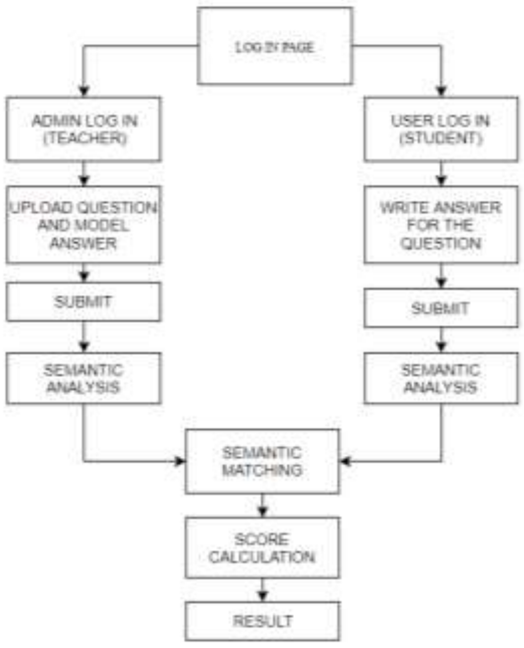

Fig. 3: Block Diagram for the working of System.

VI. PERFORMANCE EVALUATION PARAMETER

Comparing the performance of our system with other algorithms to show grades obtained using different approaches.

VII. EXPERIMENTAL SETUP

A. Hardware Requirements:

NVIDIA GPU card with CUDA Compute Capability 3.5 or higher.

B. Software Requirements:

1) Tensorflow
2) Python 3.6
3) Django

C. Dataset Required:

Set of questions, their model answers, and students' answers.

D. Output Required:

The output will be the grade that is obtained after the evaluation of the student's answer by comparing its meaning to that of the teachers. The range of the grade would be from 0 to 100.

REFERENCES

[1] Piyush Patil, Vaibhav Miniyar, Sachin Patil and Amol Bandal, "Subjective Answer Evaluation using Machine Learning" International Journal of Pure and applied mathematics, DOI: 05.23.2018.

[2] Suphakit Niwattanakul, Jatsada Singthongchai, Ekkachai Naenudorn, and Supachanun Wanapu, "Using of Jaccard Coefficient for Keywords Similarity" Proceedings of the International MultiConference of Engineers and Computer Scientists 2013 Vol I, IMECS 2013, March 13 - 15, 2013, Hong Kong

[3] Usino, Wendi & Satria, Anton & Hamed, Khalid & Bramantoro, Arif & A, Hasniaty & Amaldi, Wahyu "Document Similarity Detection using K-Means and Cosine Distance. International Journal of Advanced Computer Science and Applications". 10. 10.14569/IJACSA.2019.0100222.

[4] Bhattacharjee, Saprativa & Das, Anirban & Bhattacharya, Ujjwal & Parui, Swapan & Roy, Sudipta. "Sentiment analysis using cosine similarity measure". 27-32. 10.1109/ReTIS.2015.7232847.

[5] Albitar, Shereen & Fournier, Sébastien & Espinasse, Bernard. "An Effective TF/IDF-based Text-to-Text Semantic Similarity Measure for Text Classification". 10.1007/978-3-319-11749-2_8.

[6] Ahmad Pesaranghader and Saravanan Muthaiyah "Semantic Similarity Using First and Second-Order Co-occurrence Matrices and Information Content Vectors ".E-ISSN: 2224-2872 Issue 3, Volume 12, March 2013.

[7] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, "Efficient Estimation of Word Representations in Vector Space". arXiv:1301.3781v3 [cs.CL] 7 Sep 2013

[8] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, Ilya Sutskever, "Distributed Representations of Words and Phrases and their Compositionality",

[9] Diana Inkpen and Amir H. Razavi, "Topic Classification using Latent Dirichlet Allocation at Multiple Levels" Ijcla Vol. 5, No. 1, Jan-jun 2014, Pp. 43–55 Received 06/01/14 Accepted 06/02/14 Final 17/06/14

[10] JonasMueller, Adithya Thayagarajan, "Siamese Recurrent Architectures for Learning Sentence Similarity" Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16).