Automatic Short-Answer Grading using Corpus-based Semantic Similarity measurements

A thesis

Submitted in partial fulfillment of the requirement for the Degree of

Master of Engineering in Software Engineering

of

Jadavpur University

By

Bhuvnesh Chaturvedi
Registration Number: 140978 of 2017-2018
Exam Roll Number: M4SWE19023
Class Roll Number: 001711002022

Under the Guidance of

Ms. Rohini Basak
Assistant Professor
Dept. of Information Technology
Jadavpur University

May 2019

Certificate of Submission

This is to certify that the thesis entitled "**Automatic Short-Answer Grading using Corpus-based semantic similarity measurements**" has been carried out by **Bhuvnesh Chaturvedi** (University Registration Number: 140978 of 2017-2018, Examination Roll Number: M4SWE19023) under my guidance and supervision and be accepted in partial fulfillment of the requirement for the Degree of Master of Engineering in Software Engineering. The research results presented in the thesis have not been included in any other paper submitted for the award of any degree in any other University or Institute.

....

Ms. Rohini Basak (Thesis Supervisor)

Assistant Professor

Department of Information Technology

Jadavpur University, Kolkata-700032

Countersigned

....

Dr. Bhaskar Sardar

Associate Professor and Head

Department of Information Technology

Jadavpur University, Kolkata-700032

....

Prof. Chiranjib Bhattacharjee

Dean, Faculty of Engineering and Technology

Jadavpur University, Kolkata-700032

CERTIFICATE OF APPROVAL *

This is to certify that the thesis entitled "**Automatic Short-Answer Grading using Corpus-based semantic similarity measurements**" is a bonafide record of work carried out by **Bhuvnesh Chaturvedi** in partial fulfilment of the requirements for the award of the degree of Master of Engineering in Software Engineering in the Department of Information Technology, Jadavpur University during the period of June 2018 to May 2019. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, the opinion expressed or conclusion drawn therein but approve the thesis only for the purpose for which it has been submitted.

....

Signature of Examiner 1

Date:

....

Signature of Examiner 2

Date:

*Only in case the thesis is approved.

Declaration of Originality and Compliance of Academic Ethics

I hereby declare that this thesis entitled “**Automatic Short-Answer Grading using Corpus-based semantic similarity measurements**” contains literature survey and original research work by the undersigned candidate, as part of his Degree of Master of Engineering in Software Engineering.

All information has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name: BHUVNESH CHATURVEDI

Registration Number: 140978 of 2017-2018

Examination Roll Number: M4SWE19023

Class Roll Number: 001711002022

Thesis Title: Automatic Short-Answer Grading using Corpus-based semantic similarity measurements

....

Signature with Date

Acknowledgement

I take this opportunity to express my deepest gratitude and appreciation to all those who have helped me directly or indirectly towards the successful completion of this thesis.

Foremost, I would like to express my sincere gratitude to my respected guide and teacher **Ms. Rohini Basak** (Assistant Professor, Department of Information Technology, Jadavpur University) for her exemplary guidance, monitoring and constant encouragement throughout the course of this project. I feel deeply honored that I got the opportunity to work under her guidance.

I would also like to thank **Dr. Chiranjib Bhattacharjee** (Dean, Faculty of Engineering and Technology, Jadavpur University) and **Dr. Bhaskar Sardar** (Head of the Department of Information Technology, Jadavpur University) for providing me all the facilities and for their support to the activities of this project.

I would like to express my gratitude and indebtedness to my parents for their unbreakable belief, constant encouragement, moral support, and guidance.

Last but not the least, I would like to thank all my classmates of Master of Software Engineering batch of 2017-2019, for their co-operation and support. Their wealth of experience has been a source of strength for me throughout the duration of my work.

……………………………………………………………………

BHUVNESH CHATURVEDI

Registration Number: 140978 of 2017-2018

Examination Roll Number: M4SWE19023

Class Roll Number: 001711002022

Department of Information Technology

Jadavpur University, Kolkata-700032

Table of Contents

List of Figures

List of Tables

ABSTRACT

In this thesis, some unsupervised techniques for the task of Automatic Short Answer Grading (ASAG) have been explored. The task of ASAG involves scoring a student answer (SA) based on a given reference answer (RA). The RA and SA are provided as input to the method, after which the method computes the similarity between the two and then awards a similarity score to the SA based on how similar the SA is with RA. The corresponding question (Q) is also provided as input to the method but it is not necessary that it will be used when the SA is being evaluated.

Corpus-based semantic similarity measures along with a set of triple matching rules are used here to find out the similarities between the RA and SA. The scoring of the SA is based on the criteria that if the SA is found more similar to the given RA, then a higher score will be awarded to it.

Four models are used for the task in hand, with each model having their own advantages as well as disadvantages. The first model tags the words in RA and SA with their corresponding Parts-of-Speech tags (for example NN for noun, VB for verb and so on) using (POS) tagging and performs matching between the words having equivalent tags (i.e. nouns are matched with nouns; verbs are matched with verbs and so on). The second model parses the RA and SA into a set of dependency triples and performs matching between the triples. The third model also generates dependency triples by parsing which is followed by matching the triples, with the difference being that the RA-SA triples are matched based on some predefined rules. The fourth and final model uses the concept of word to word matching between the RA and SA, with one word of RA being matched with one and only one word of SA. All four models have used gensim's pre-trained Word2vec model on Google-news corpus and fastText model on English Wikipedia corpus.

The proposed method has shown quite promising correlation values between scores generated by the method and those awarded by human scorers on a standard computer science dataset, which consists of 12 assignments where each assignment has different number of questions along with a single RA and multiple SA for each question. The dataset has been described in details in subsequent chapters. The correlation reached the peak value at 0.646 for a single assignment (assignment 4 with Word2vec model) and an overall correlation reaching the highest value at 0.805.

Keywords: Automatic short answer grading, semantic similarity, dependency parser, POS tagger.

Chapter 1

Introduction

The task of Automatic Short Answer Grading (ASAG) involves scoring a student answer (SA) based on a given reference (model) answer (RA or MA). Optionally, scoring schemes may also be provided to indicate relative importance of different parts of the model answer. This is a complex natural language understanding task due to linguistic variations (same answer could be written in different ways), subjective nature of assessment (multiple possible correct answers or no correct answer) and lack of consistency in human rating (non-binary scoring on an ordinal scale within a range). Much prior ASAG work requires supervisors to grade key concepts and their variations using concept mapping or to grade a fraction of student answers to train supervised learning algorithms.

Much work has already been done in the field of ASAG in recent years. These works can be mainly divided into two categories. The first one is supervised approach, in which the graders first grade a few students' answers which in turn are used as training data for the classification algorithm. The second is unsupervised approach, in which text similarity approaches are used to compare model answer and students' answers to assign suitable grades to the student answers.

In this thesis, we present an unsupervised approach, which consists of four models that score the students' responses individually. One thing to note is that the similarity score is calculated for the RA keeping SA as reference thus reversing their roles. This is done when the SA is more elaborate while the RA is compact. If the SA is compact while the RA is more elaborate, the similarity score of SA is calculated using RA as reference. This is done for all four models and the justification for this is provided later in details. The first model starts with Parts-Of-Speech (POS) tagging of RA as well as SA, after which one word having a certain POS tag from RA is matched with all the words of SA having equivalent POS tags (for example, a Noun from RA is matched with all Nouns from SA) and finally the maximum similarity score is taken. Such similarity score is obtained for every word of RA and finally an average of these scores is taken to compute the mark that is to be awarded to the student. The similarity scores are separately obtained using Word2vec model on Google-news corpus and fastText model on English Wikipedia corpus. Both mechanisms generate a 300-dimensional vector for a word that is fed into it. To compare two words, the words are fed into the model, the outputs of which are their corresponding vectors. Then the cosine value of the angle between the two vectors is calculated which gives a value between -1 and +1, which is their similarity score. These corpus-based similarity calculation models are used in all four models.

The second model parses the RA and SA into a number of dependency triples (a triple consists of three parts: the relation, the governor word and the dependent word). Then the governor and dependent words of each triple from RA are matched with the governor and dependent words respectively of every triple from SA and finally the maximum similarity score is taken. Such similarity score is obtained for every triple of RA and finally an average of these scores is taken to compute the mark that is to be awarded to the student. The governor and dependent words are matched using the same Word2vec and fastText models used in Model 1.

The third model is similar to the second model except only one difference. In this model, the triples are matched based on a set of pre-defined matching rules. Thus, unlike in previous model in this model, the relation part of the triple plays a significant role. This model also first parses the RA and SA into a set of dependency triples. Then the triples of RA are matched with the triples of SA following the conditions of Rules 1 to 5 (details of them are provided later). In other words, first it is checked whether the RA and SA triples satisfy rule 1 and if they do, then a similarity score is awarded to the corresponding RA triple as per rule 1. If they do not satisfy rule 1, then it is checked whether they satisfy rule 2 and if they do, then a similarity score is awarded to the corresponding RA triple as per rule 2, and so on up to rule 5. If an RA triple matches with more than one SA triple for any rule, then similarity score is calculated for every pair of RA triple and matching SA triple, and the highest score among them is assigned to the RA triple. Once every RA triple is scored, the average similarity score is computed by taking the average of the similarity scores of all RA triples over the number of RA triples, which is then used to calculate the marks that is to be awarded to the student. Stanford POS tagger and dependency parser are used for POS tagging and triple extraction respectively in all the three models.

The fourth and final model arranges the words of RA and SA in a two-dimensional matrix, with the words in RA (or SA; whichever has less number of words) in rows and words of SA (or RA) in columns. Once done, similarity score of a pair of words (one taken from a row and the other from a column) is calculated. The words are matched using the same Word2vec and fastText models used in Model 1. Once all these scores are obtained, the highest score from the entire matrix is taken, after which the row and column to which the score belongs is struck off from being considered. Then the next highest score is considered, and the process is repeated until a similarity score from each of the rows is considered. Finally an average of these scores is taken over the number of rows to compute the mark that is to be awarded to the student answer.

For evaluation, the correlation value between the obtained scores from the models with those awarded by human judges is calculated. Higher the correlation value between the obtained score and the scores given by human judges, the more accurate is the method to assign grades to the student answers closely to the human judges.

Chapter 2

Literature Survey

Many approaches have already been proposed for the task of ASAG. A few of them have been discussed below.

Gomma et al. [1] presented an unsupervised approach which uses the concept of text to text similarity. Different string-based and corpus-based similarity measures were tested separately and the results were then combined to obtain a final score.

Adams et al. [2] evaluated a variety of representative distributional semantics based approaches for the task of unsupervised grading and have proposed an asymmetric method based on aligning word vectors that exploits properties of grading tasks. Aligning word vectors means moving the word vectors of one document in the vector space so that they can become word vectors of another document. This method allows words to move to multiple other words when a mismatch in document size occurs.

Alotaibi et al. [3] proposed a combination of different techniques to produce an approach for automatic short answer marking. In particular an integrated method was presented which uses Information Extraction (IE) and Machine Learning (ML) techniques. The result of the whole approach was to assign a mark depending on the percentage of correctness in the student's response not just classifying the resultant mark as correct or incorrect.

Roy et al. [4] proposed a technique which was based on the intuition that student answers to a question, as a collection, are expected to share more commonalities than any random collection of text snippets. If these commonalities can be identified from student answers then the same could be used to score them.

Sultan et al. [5] presented a fast and simple supervised method. From the given RA and SA, a number of text similarity features were extracted and then combined with key grading-specific constructs.

Pérez et al. [6] presented the working of a Computer-Assisted Assessment (CAA) system called Atenea. The evaluation process starts with tokenizing both SA and RA. After that individual scores are calculated by using Evaluating Responses with BLEU (ERB) and Latent Semantic Analysis (LSA) algorithms independently which are then combined to obtain a final score.

Bachman et al. [7] presented the working of a language assessment system called WebLAS (web-based language assessment system). It takes the input (RA, elements, alternatives, score assignments) to create a scoring key, which then employs regular expressions (regexes) for pattern matching. The task of scoring is simply the process of matching the regexes with the elements in the student response to score each such element.

Leacock et al. [8] presented the working of an automated short-answer marking engine called C-rater. If a set consisting of all the possible correct student responses is considered, then the c-rater scoring engine operates as a paraphrase recognizer that identifies members of this set, i.e. it identifies whether the student response is a paraphrase of the correct concept or not.

Basak et al. [11] presented a method that casts the ASAG task to an RTE task (Recognizing Textual Entailment), which determines whether the meaning of SA, or RA, is logically entailed by RA, or SA. The better the entailment between the RA-SA pair, the higher the score to be assigned to the SA. To determine the entailment, they have used the rules from one of their other works [12]. The matching rules were associated with numerical scores and the total score to be awarded was calculated in a well-specified manner.

Chapter 3

The proposed method

The proposed method consists of four models. Before going into the detailed description of those models, a brief description of the tools and libraries that have been used in the work are presented.

3.1 Tools

The tools used are described in the following subsection:

3.1.1 Stanford Parser

Parsing is the process of analyzing a string of symbols, either in natural language, computer language or data structures, conforming to the rules of a formal grammar. Parsing helps in analyzing the input text by associating additional grammar context to it. It also helps in identifying the subject, objects of interest, parts of speech information, etc. Applications can be built on it using this information to improve the results of their specific use case. Some of the applications where parsing is used are language prediction, translation, text similarity measurement, etc.

In machine translation and natural language processing systems, written text in human languages are parsed by computer programs, known as parsers. One such well known parser is Stanford Parser. It is a natural language parser which works out on the grammatical structure of sentences, for example, which group of words go together (as “phrases”) and which words is the subject or object of a verb. Probabilistic parsers use knowledge of language gained from hand-parsed sentences to try to produce the most likely analysis of new sentences. These statistical parsers still make some mistakes, but commonly work rather well.

The package[1] which was used here as part of the task is a Java implementation of probabilistic natural language parsers, both highly optimized PCFG (Probabilistic Context-Free Grammar) and lexicalized dependency parsers, and a lexicalized PCFG parser. The lexicalized probabilistic parser implements a factored product model, with separate PCFG phrase structure and lexical dependency experts, whose preferences are combined by efficient exact inference, using an A* algorithm. The software can also be used simply as an accurate unlexicalized stochastic context-free grammar parser. Either of these yields a good performance statistical parsing system.

[1] **https://nlp.stanford.edu/software/lex-parser.shtml#Download**

3.1.2 Gensim's Word2Vec

Word2Vec is a group of related models that are used to produce word embedding. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2Vec takes as its input a large corpus of text and produces a vector space which are usually of several hundred dimensions such that each unique word is assigned a unique vector in the vector space. Word vectors are positioned in the vector space in such a way that words with common context in the corpus are located in close proximity to one another.

Given enough data, usage and contexts, Word2Vec can make highly accurate guess about a word's meaning based on past appearances. Those guesses can be used to establish a word's association with other words (e.g. "Queen" is to "woman" then "King" is to what?), or cluster documents and classify them by topic. In ASAG systems, this can be used to cluster out the correct responses from the incorrect ones.

Once the vector representations of the words are obtained, different algorithms to measure the similarity between the words can be applied. One of the most commonly used methods is "cosine similarity" where the cosine value of the angle between the vectors corresponding to the two words is measured. A value of 0 signifies an angle of 90 degrees which implies that the words are not at all similar to each other. A value of 1 signifies an angle of 0 degrees which implies that the words are exactly same. A value between 0 and 1 represents the similarity between words with a value closer to 0 implying less similarity while a value closer to 1 represents more similarity. A Word2Vec comparison between *Sweden* and *Sweden* produces a value of 1 while between *Sweden* and *Norway* produce a value of 0.770617.

Word2Vec works like an auto-encoder in the sense that it is encoding each word to a vector and it does so by training words against other words that neighbour them in the input corpus. This can be done in two ways, either using context words to predict a target word (a method known as Continuous Bag-of-Words or CBOW), or using a word to predict a target context, which is called skip-gram. CBOW is faster than skip-gram but skip-gram performs better when we are dealing with infrequent words. However skip-gram produces more accurate results on large datasets.

The use of different model parameters and different corpus sizes can greatly affect the quality of a word2vec model. Accuracy can be improved in a number of ways, including the choice of model architecture (CBOW or Skip-Gram), increasing the size of the training data set, increasing the number of vector dimensions, and increasing the window size of words considered by the algorithm. Each of these improvements comes with the cost of increased computational complexity and therefore increased model generation time.

In models using large corpora and a high number of dimensions, the skip-gram model yields the highest overall accuracy, and consistently produces the highest accuracy on semantic relationships, as well as yielding the highest syntactic accuracy in most cases. However, the CBOW is less computationally expensive and yields similar accuracy results.

3.1.3 fastText

fastText is a library for learning of word embeddings and text classification. It is created by Facebook's AI Research (FAIR) lab, and is written in C++ with support of multiprocessing during training. The model allows us to create a supervised or unsupervised learning algorithm for obtaining vector representation of words. Like Word2Vec, fastText also supports training Continuous Bag-of-Words (CBOW) or Skip-gram models using negative sampling, softmax or hierarchical softmax loss functions.

fastText is able to achieve really good performance in case of word representations and sentence classification, especially in the case of rare words by making use of character level information. It is achieved by representing each word as a bag of character n-grams in addition to the word itself. For example, if the word is "matter" with n=3 (a tri-gram), the fastText representations for the character n-grams is <ma, mat, att, tte, ter and er>. The characters < and > are added as boundary symbols to distinguish the n-grams of a word from a word itself. For example, if the word "mat" is part of the vocabulary, it is represented as <mat>. This helps in preserving the meaning of shorter words that might show up as n-grams of other words (like in our example, *mat* showed up as an n-gram for the word *matter*).

The length of n-grams to be used can be controlled by a set of min (for minimum number of characters to be used) and max (for maximum number of characters to be used) flags. These two flags control the range of values to get n-grams. The model is considered to be a bag of words model because apart from the sliding window of n-gram selection, there is no internal structure of a word that is taken into account. In other words, as long as the characters fall under the sliding window, the order of the character n-grams does not matter. Even the n-gram embeddings can be completely turned off by setting the value of both flags to zero. This is useful in cases when the words in the model are not words for a particular language and character level n-grams would not make sense.

Now, while fastText supports multi-threading while training, reading the data on which training is to be performed is done by a single-thread. This will create a bottleneck if the input data is huge. The parsing and tokenization is done when the input data is read. fastText only supports reading input data from a file and not from standard input (stdin) like keyboard. During the reading phase, for each unique word, four values are stored in a struct. These four values are *word* which is the string representation of the unique word, *count* which is the total count of the respective word in the input line, *entry_type* which is either word or label (used only for the supervised case), and *subwords* which is a vector of all the word n-grams of a particular word. These are also created when the input data is read, and passed on the training step.

Once the input and vectors are initialized, multiple training threads are started, the count of which can be set by using a thread argument. All these threads read from the input file, while updating the model with each input line that is read. An input line is truncated if a newline character is found or if the count of words that has been read reaches the maximum allowed line size, the default value being 1024.

The target vector for the loss function is computed by a normalized sum of all the input vectors. The input vectors are the vector representation of the original word, as well as all the n-grams of that word. The loss is computed which sets the weights for the forward pass, which propagate their way all the way back to the vectors for the input layer in the back propagation pass. This tuning of the input vector weights that happens during the back propagation pass is what allows the model to learn representations that maximize co occurrence similarity. The learning rate affects how much each particular instance affects the weights.

Once the whole process is complete, the model input weights, hidden layer weights along with arguments passed in are saved in a file with .bin format. Another flag is also provided which controls whether a .vec file is also created which contains vectors for the hidden layer in the word2vec file format. This .vec file can be used if we are not interested in building and training our own model and wants to use a pre-trained one.

Now that overviews of the tools that have been used are provided, the proposed methods with the detailed description of the models are introduced.

3.2 The Method

The method consists of four different models, each model scoring the students' responses individually. The pre-processing steps for all four models are same, and are explained in the following subsection.

3.2.1 Pre-processing Steps

Before invoking the actual process of ASAG, all RA and SA are subjected to the pre-processing steps first, which include splitting words joined by a slash or hyphen (eg. "function/method" becomes “function method” and “run-time” becomes “run time”) and removing unnecessary symbols (the dataset contains certain HTML tags which need to be removed). Certain punctuation symbols like “,”, “;”, ”.” were removed as their absence did not impact the scores generated by the method, but their presence causes generation of unnecessary dependency triples in Models 2 and 3. After that both RA and SA are converted to lowercases. The reason is that both Word2vec as well as fastText produce different similarity score when the case is changed and compared to when the case is same.

Once pre-processing is done, the actual method of ASAG is invoked. The four models are explained in the subsequent subsections. One important thing to note is that in the proposed models, we have shown that the SA is evaluated with reference to the RA in cases where SA was precise and concrete while RA was elaborate. However for some Q, it was found that the RA was concrete and stated as a gist, while the SA was elaborate containing extra piece of information. For such cases, the roles of RA and SA were reversed, i.e., the RA was evaluated with reference to the SA. This is a part of the process in each of the four models and not a separate model by itself, i.e., before the actual process begins; the length of RA and SA is checked and the decision is made as to whether SA will be scored based on RA or RA will be scored based on SA. This is applicable for all four models.

3.2.2 Model 1

The RA and SA are individually fed into Stanford POS-tagger [9] for Part-of-Speech (POS) tagging. As output from this phase the RA and SA with each word tagged with their corresponding POS are generated (eg. An input of “at the main function” will give an output of “at/IN the/DT main/JJ function/NN”, where NN signifies singular noun, JJ signifies adjective, IN signifies preposition or subordinating conjunction and DT signifies determiner).

Once the POS tagging is done, the response matching and scoring mechanism process gets started. Each word belonging to a particular POS tag in RA is compared with every word in SA having an equivalent POS tag (eg. a word in RA with POS tag NN will be matched with every word in SA with equivalent POS tags of NN, NNS, NNP or NNPS) and the maximum similarity score is considered. The similarity score between a given pair of words is determined by gensim’s Word2vec model and fastText model separately, where a 300 dimension vector of the words are generated in both the models and the similarity score is obtained by measuring the cosine value of the angle between the two word vectors.

Once the maximum score is obtained for each word in RA or SA as required and justified previously, the overall similarity score is calculated by taking the average of all the maximum scores over the number of words in the RA. This overall similarity score is then multiplied by the full marks of the question to obtain the marks that is to be awarded to the SA.

The above method can be stated in the form of an algorithm below:-

Step 1: Perform necessary pre-processing steps.

Step 2: Feed the RA and SA obtained after applying Step 1 into Stanford POS tagger individually to obtain RA and SA with each word tagged with their corresponding POS.

Step 3: Take each word from RA with a POS tag and match it with every word from SA having equivalent POS tags. The similarity scores thus obtained are grouped together.

Step 4: Once such groups are obtained for every word in RA, maximum scores from each group is considered.

Step 5: Once the maximum scores are obtained, they are summed up and then averaged over the number of groups to obtain the overall similarity score.

Step 6: The similarity score obtained in the above step is then multiplied with the full marks of the corresponding question to obtain the grade that is to be awarded to the SA.

The above steps are repeated for every SA of every question in the whole assignment, where both RA and SA consist of more than one word. If RA and/or SA consists of only one word, the similarity score and the grade to be awarded is calculated in a special way, the details of which are provided later.

3.2.3 Model 2

The RA and SA are fed into Stanford Parser [9] one by one and are parsed into a number of dependency triples. Once the dependency triples are obtained, each triple of RA is compared with every triple of the SA, except for the triple with relation as *root*, and the maximum matching score is considered to be taken. The reason for excluding the *root* triple is that *root* is an extra relation generated just for the purpose of presenting the word on which the parse tree is rooted from and has no other significance. Its absence does not impact the score but its presence is found to lower the overall similarity score, as the governor part of the root relation, which is ROOT, produces a very low score when compared with the governor nodes of other triples, thus lowering the similarity score, which in turn lowers the overall score.

The matching score is obtained by comparing the governor word pairs of the RA and SA triples and the dependent word pairs of the RA and SA triples and then finally averaging the sum of the two scores. The same gensim's Word2vec model and fastText model used in model 1 are also used here to determine the similarity score between the corresponding pairs of words of the RA-SA triples.

Once the maximum score is obtained for each triple of RA, the overall similarity score is calculated by taking the average of all the maximum scores over the number of triples of the RA. This overall similarity score is then multiplied by the full marks of the question to obtain the marks that is to be awarded to the SA.

The above method is described in the form of an algorithm below:-

Step 1: Perform necessary pre-processing steps.

Step 2: Feed the RA and SA obtained after applying Step 1 into Stanford Parser to convert them into a set of triples of the form R(G,D) from both RA and SA. Here R signifies the Relation part, G signifies the Governor part and D signifies the Dependent part.

Step 3: Take each triple from RA and match it with each triple of SA by matching their corresponding governor (G) parts and dependent (D) parts, irrespective of the relations (R) parts. The similarity score obtained are grouped together, i.e. the similarity score obtained when each triple from SA is matched with one triple from RA are put in one group from which the highest similarity score will be considered**.** For example, if one triple from RA is matched with three triples from SA, three similarity scores will be obtained, one for each triple in SA. These three scores are put in one group and the highest of them all is taken into consideration when the average similarity score is being calculated. Such groups are generated for every triple in RA.

Step 4: Once such groups are obtained for every triple of RA, maximum scores from each group is considered.

Step 5: Once the maximum scores are obtained, they are summed up and then averaged over the number of triples in RA to obtain the overall similarity score.

Step 6: The similarity score obtained in the above step is then multiplied with the full marks of the corresponding question to obtain the grade that is to be awarded to the SA.

The above steps are repeated for every SA of every question in the whole assignment, where both RA and SA consist of more than one word. If RA and/or SA consists of only one word, the similarity score and the grade to be awarded is calculated in a special way, the details of which are provided later.

3.2.4 Model 3

Similar to Model 2, this model also parses the RA and SA into a set of dependency triples by subjecting each of them individually into Stanford dependency parser.

Once the dependency parsing phase is over, the actual process of matching and scoring starts. However unlike in Model 2, here the matching operations between the RA-SA triples are performed based on a set of predefined rules [12]. For simplified notation, let us consider that the triples generated from RA are of the form $R_R(G_R, D_R)$, where R_R signifies the relation, G_R is the governor and D_R is the dependent. Similarly triples generated from SA are of the form $R_S(G_S, D_S)$, where R_S signifies the relation, G_S is the governor and D_S is the dependent. Here too the root triple is ignored, the reason being same as that given in Model 2.

The rules are described here in details and are as follows:

3.2.4.1. Rule 1 If the SA triple $R_S(G_S, D_S)$ completely matches with the RA triple $R_R(G_R, D_R)$, i.e., R_S matches with R_R, G_S matches with G_R, and D_S matches with D_R, then a complete matching score of 1 is assigned to the RA triple. Fig. 1 below illustrates this matching rule.

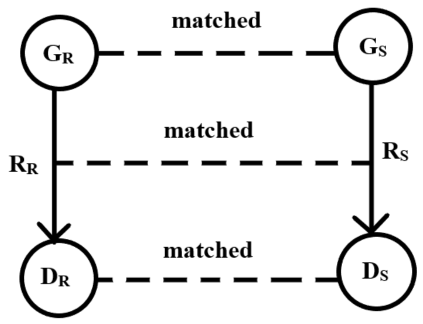


Fig. 1. Matching rule 1

3.2.4.2. Rule 2 If the nodes D_S and G_S of SA triple completely match with the nodes G_R and D_R respectively of a RA triple but the relations R_S and R_R do not match, then if the relation R_R belongs to the set {amod, rcmod, ccomp, advcl}, while the relation R_S belongs to the set which matches the conditions given in Table 1, then a full score of 1 is assigned to the RA triple. Fig. 2 below illustrates this matching rule.

Table 1 (based on rules defined in [12])

R_R (Relation from RA triple)	Equivalent R_S (Relation from SA triple)
amod	dobj, nsubjpass, nsubj, nn, amod
rcmod	nsubj, agent, nsubjpass
ccomp	nsubj
advcl	nsubjpass

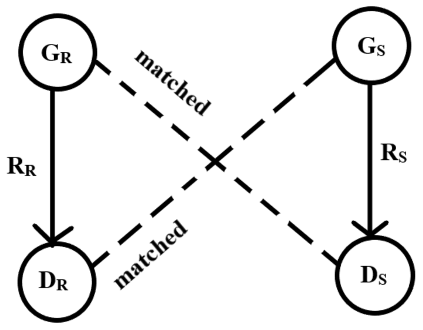


Fig. 2. Matching rule 2

3.2.4.3. Rule 3 If the nodes G_S and D_S of SA triple match with the nodes G_R and D_R respectively of the RA triple but the relations R_S and R_R do not match, then for the conditions given in Table 2, a full score of 1 is assigned to the RA triple. Fig. 3 below illustrates this matching rule.

Table 2 (based on rules defined in [12])

R_R (Relation from RA triple)	Equivalent R_S (Relation from SA triple)
Appos	nsubj, dobj, nn, dep, tmod, rcmod, conj_and
Rcmod	infmod, appos, partmod
Prep	nsubjpass
Xsubj	dobj

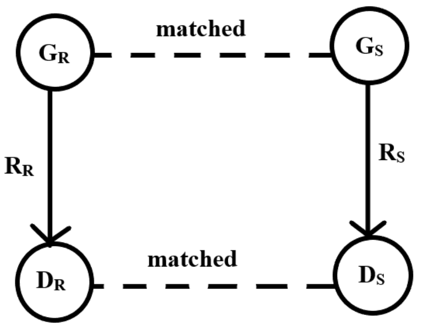
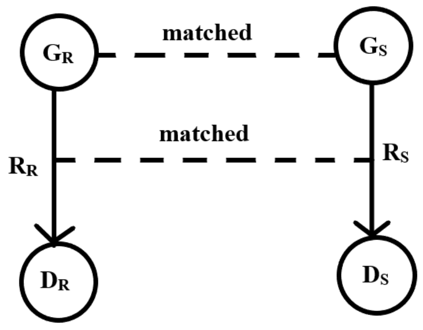
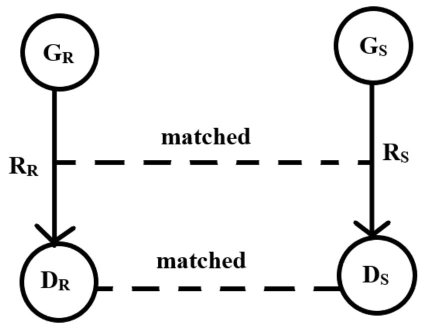


Fig. 3. Matching rule 3

3.2.4.4. Rule 4 If the relation R_S exactly matches with the relation R_R and either nodes G_S and G_R matches exactly or nodes D_S and D_R matches exactly then the nodes G_S and G_R are compared with each other to obtain their similarity score. Similarly the nodes D_S and D_R are compared to obtain the similarity score between them. Then the average of these two scores is taken, which gives the score that is to be assigned to this RA triple. Fig. 4(a) and Fig. 4(b) below illustrate both parts of this matching rule.

 G_R matched G_S matched R_R R_S D_R D_S Fig. 4(a). Matching rule 4 (Governor matches)	 G_R G_S matched R_R R_S matched D_R D_S Fig. 4(b). Matching rule 4 (Dependency matches)

3.2.4.5. Rule 5 There are some relations that, although seem to be less important and insignificant, cannot be ruled out and need to be considered. These relations are as follows

{aux, auxpass, cop, det, expl, mark, nn, prt, predet, cc}

If these relations are ignored, it results in generating lower score. Therefore they should also be considered. The words G_S and G_R are compared with the words D_S and D_R respectively to obtain their similarity score. The average of these two scores is taken which gives the score that is to be assigned to this RA triple.

This only happens if there is at least one triple in both RA and SA which have the relation belonging to the above set. In other words, if there is one triple in RA and one triple in SA with any of the above relation, then their governors and dependents are matched and averaged to obtain the similarity score to be assigned to RA. If there are two or more triples in RA with any of the above relation but only one triple in SA with any of the above relation, then each of the triples in RA is matched with the one triple in SA and the obtained similarity score is assigned to the RA triple. If there is one triple in RA with any of the above relation and two or more relations in SA with any of the above relation, then the one triple in RA is matched with all such triples in SA and the maximum similarity score is taken from them and assigned to the RA triple.

If it so happens that either only RA or SA contains such triples with the relation part belonging to the above relation set, then the triples are ignored and not considered for calculation of the similarity score. The reason is that there is no triple in the SA (considering that RA contains the triple with any of the above relations) with a relation equivalent to the ones in the above set. For example, RA contains a triple of the form det(method-2,a-1) while SA does not contains any triples with any of the above relations, but say a triple of the form conj:and(push-1, pop-3) then the two triples will not be compared as the relations “det” and “conj:and” are not equivalent.

If an RA triple matches with multiple SA triples, then the maximum of all the maximum scores is considered and assigned to this triple. Once a score is assigned to each RA triple, the overall similarity score is calculated by taking the average of all the maximum scores over the number of triples in the RA. This overall similarity score is then multiplied by the full marks of the question to obtain the marks that is to be awarded to the SA.

The above method is described in the form of an algorithm below:-

Step 1: Perform necessary pre-processing steps.

Step 2: Feed the RA and SA obtained after applying Step 1 into Stanford Parser one by one which converts them into a set of triplets of the form R(G,D) from both RA and SA. Here R signifies the Relation part, G signifies the Governor part and D signifies the Dependent part.

Step 3: Take each triple from RA and SA and perform the sub-steps 3.(i) to 3.(v) given below:

Step 3(i): If the RA and SA triples satisfy Rule 1, then assign a similarity score of 1 to the RA triple.

Step 3(ii): If the RA and SA triples satisfy Rule 2, then assign a similarity score of 1 to the RA triple.

Step 3(iii): If the RA and SA triples satisfy Rule 3, then assign a similarity score of 1 to the RA triple.

Step 3(iv): If the RA and SA triples satisfy Rule 4, then the average of the similarity scores of the Governor parts and the Dependent parts is assigned to the RA triple.

Step 3(v): If the RA and SA triples satisfy Rule 5, then the average of the similarity scores of the Governor parts and the Dependent parts is assigned to the RA triple.

The similarity score obtained in the above sub-steps are grouped together on the basis of each RA triple.

Step 4: Once such groups are obtained for every triple in RA, maximum scores from each group is obtained.

Step 5: Once the maximum scores are obtained, they are summed up and then averaged over the number of groups to obtain the overall similarity score.

Step 6: The similarity score obtained in the above step is then multiplied with the full marks of the corresponding question to obtain the grade that is to be awarded to the SA.

The above steps are repeated for every SA of every question in the whole assignment, where both RA and SA consist of more than one word. If RA and/or SA consists of only one word, the similarity score and the grade to be awarded is calculated in a special way, the details of which are provided later.

3.2.5 Model 4

The words of the RA and SA are arranged in the form of a two-dimensional matrix. Which of the RA or SA words will be arranged in the rows or columns depends on the length or the number of words in the answer. If the RA contains lesser number of words than the SA, its words will be arranged row-wise while those of SA will be arranged column-wise, and vice versa.

After the arrangement of the words, each word from a row is matched with every word in the columns to obtain a similarity score. The same gensim's Word2vec model and fastText model used in model 1 are also used here to obtain the similarity score.

Next the scoring mechanism starts by taking the highest similarity score from the overall matrix, and then striking out the corresponding row and column. For example, a maximum score of say 0.98 is obtained at row 2 and column 3 of the matrix. Then this score is taken into consideration and it is made sure that none of the other scores belonging to row 2 and column 3 will be considered in the next iteration. This is done to make sure that one word from RA is matched to one and only one word of SA. The next maximum score from the rest of the matrix barring the ones contained in the stricken out row and column is obtained and the corresponding row and column are struck off. The entire process is repeated until the maximum scores for every row is taken into account.

Finally an average of these scores is taken over the number of rows to obtain an overall similarity score, which is then multiplied by the full marks carried by the question to obtain the marks that is to be awarded to the SA. The idea of considering the maximum similarity score is based on n-queen's problem of greedy approach, where the target is to maximize the score that is to be awarded to the SA.

The above method is described in the form of an algorithm below:-

Step 1: Perform necessary pre-processing steps.

Step 2: Arrange the words from RA (or SA depending on which has fewer words) in rows. Similarly arrange the words from SA (or RA) in columns. We will obtain a two-dimensional matrix.

Step 3: Obtain similarity scores between each pair of words, one taken from a row and other from a column and store it in the two-dimensional matrix.

Step 4: Repeat the sub-steps 4.(i) and 4.(ii) until maximum score for every row is obtained:-

Step 4(i): Obtain the maximum score from the overall matrix.

Step 4(ii): Once obtained, strike off the row and column to which this score belongs. Once struck off, any score in that row and column will no more be considered while finding the next highest score.

Step 5: Once the maximum scores are obtained for every row, they are summed up and then averaged over the number of rows to obtain the overall similarity score.

Step 6: The similarity score obtained in the above step is then multiplied with the full marks of the corresponding question to obtain the grade that is to be awarded to the SA.

The above steps are repeated for every SA of every question in the whole assignment, where both RA and SA consist of more than one word. If RA and/or SA consists of only one word, the similarity score and the grade to be awarded is calculated in a special way, the details of which are provided later.

Apart from the four models, a separate mechanism was included for scoring in special cases in which RA or SA consists of only a single word. For example, for certain questions, the RA is simply "push" whereas the SA contains the word "push" or one of its synonyms along with some extra information. For such cases, this single word in RA (or SA if the role of SA and RA are reversed) is compared with every word in SA, or RA, and the maximum similarity score is taken and used to calculate the score that is to be awarded to the SA in consideration. The point to be noted here is that this is not a separate model; rather this is part of all the four models and is used only for the special case mentioned here.

The above method is described in the form of an algorithm below:-

Step 1: Perform necessary pre-processing steps.

Step 2: The single word in RA (or SA) is matched with every word in SA (or RA) and the maximum similarity score is taken.

Step 3: The similarity score obtained in the above step is then multiplied with the full marks of the corresponding question to obtain the grade that is to be awarded to the SA.

The above steps are repeated for every SA of every question in the whole assignment, where the RA or SA or both consists of only one word.

Chapter 4

Simulation

Consider the following example of Q-RA-SA triple taken from the dataset*:

Q: *What are the two main functions defined by a stack?*

RA: *push and pop*

SA: *pop and push*

The working is only shown with Word2vec. The approach is same when using fastText and so its working is not shown.

4.1 Model 1

Individual words along with their corresponding POS tags for both RA and SA are shown in Table 3.

Table 3 (words with POS tags)

Words in RA	**POS tags**	**Words in SA**	**POS tags**
push	NN	pop	NN
and	CC	and	CC
pop	NN	push	NN

The scores obtained when the model is run with Word2vec are shown in Table 4.

Table 4 (scores obtained with Word2vec)

	pop(NN)	and(CC)	push(NN)
push(NN)	0.185	-	1.0
and(CC)	-	1.0	-
pop(NN)	1.0	-	0.185

It is to be noted that since the length of SA is equal to RA, their roles have been kept as it is and SA is evaluated with respect to RA.

*** web.eecs.umich.edu/~mihalcea/downloads/ShortAnswerGrading_v2.0.zip**

Now “and” from SA only matches one word “and” in RA with similarity score of 1.0, so this score is considered. For “push” from SA, there are two matching words in RA with similarity scores of 1.0 and 0.185 among which 1.0 is maximum of the three, so this score is considered. For “pop” from SA also, there are two matching words in RA with similarity scores of 1.0 and 0.185 among which 1.0 is maximum of the three, so this score is considered.

Now that all similarity scores are obtained, the average similarity score is calculated as:

Score(Word2vec) = (1.0+1.0+1.0)/3=1.0

Since the full mark of this Q is 5, the grade that is to be awarded to this SA is calculated as:

Grade(Word2vec)=1.0*5=5.0

The two human graders have awarded scores of 5 and 5 respectively to this response, and the obtained score of 5.0, is same as the scores awarded by the human judges.

4.2 Model 2

The RA and SA dependency triples of the above Q-RA-SA triple are provided in Table 5.

Table 5 (RA and SA dependency triples)

RA triples	**SA triples**
1: root(ROOT-0, push-1) 2: cc(push-1, and-2) 3: conj:and(push-1, pop-3) 1: root(ROOT-0, pop-3)	1: root(ROOT-0, pop-1) 2: cc(pop-1, and-2) 3: conj:and(pop-1, push-3) 1: root(ROOT-0, push-3)

The matched token pairs, their similarity scores and the average similarity scores are provided in Table 6.

Table 6 (Assignment of scores)

Tokens being matched	Similarity Score	Average Score (score to be assigned to the triple)
push→pop	0.185	0.593
and→and	1.0	
push→pop	0.185	0.110
and→push	0.034	
----	----	----
push→pop	0.185	0.116
pop→and	0.047	
push→pop	0.185	0.185
pop→push	0.185	

In Table 6, the first four rows are put in one group (group 1) while the next four rows are put in another group (group 2). The reason is that the first group consists of similarity scores obtained when triple 2 of SA is matched with all two triples of RA, and the next group consists of similarity scores obtained when triple 3 of SA is matched with all two triples of RA. The "root" triples are not considered for the reasons described previously. The maximum score from group 1 is 0.593 while the maximum score from group 2 is 0.185.

Once the grouping is done, the overall average similarity score is calculated as

Score(Word2vec) = (0.593+0.185)/2 = 0.389,

and the grade to be awarded to the SA is

Grade(Word2vec)=0.389*5=1.9

The two human graders have awarded scores of 5 and 5 respectively to this response, and the obtained score of 1.9, is far from the scores awarded by the human judges. The reason is that this model is simply comparing the governors and dependents of the RA and SA triples without considering any other things.

4.2 Model 3

The RA and SA dependency triples of the Q-RA-SA triple are provided in Table 7, while the matched token pairs, the rule based on which they are matched, their similarity scores and the average similarity scores are provided in Table 8.

Table 7 (RA and SA dependency triples)

RA triples	SA triples
1: root(ROOT-0, push-1) 2: cc(push-1, and-2) 3: conj:and(push-1, pop-3) 1: root(ROOT-0, pop-3)	1: root(ROOT-0, pop-1) 2: cc(pop-1, and-2) 3: conj:and(pop-1, push-3) 1: root(ROOT-0, push-3)

Table 8 (Assignment of scores)

Tokens being matched	MR	SS	AS
push→pop	5	0.185	0.593
and→and		1.0	
----		----	----
push→push	2	1.0	1.0
pop→pop		1.0	

Here, MR signifies the matching rule number, SS signifies the similarity score of the tokens being matched and AS signifies the average of the similarity scores between the corresponding governors and dependents of RA and SA, which is the score that to be assigned to the triple.

Here every triple of RA is getting matched to only one triple of SA, so obtaining maximum score for each triple of RA is not required since there is only one.

The overall average similarity score is calculated as

Score(Word2vec) = (0.593+1.0)/2 = 0.797,

and the grade to be awarded is

Grade(Word2vec)=0.797*5=3.985=4.0(rounded off to one decimal place)

The two human graders have awarded scores of 5 and 5 respectively to this response, and the obtained score of 4.0, is significantly closer to the scores awarded by the human judges.

It can be observed that score significantly improved once the rules are applied during the matching process. This proves the effectiveness of Model 3 as compared to Model 2, even though both models are using the concept of triple matching.

4.4 Model 4

At first the RA-SA words are arranged in the form of a 2 dimensional matrix. If the RA contains less number of words than SA, then RA words are arranged in the rows while the SA words are arranged in the columns. Similarly if the SA contains less number of words than RA, then SA words are arranged in the rows while the RA words are arranged in the columns. Once this arrangement is done, the similarity scores are calculated and arranged in a similar 2 dimensional matrix.

The scores obtained when the model is run with Word2vec are arranged in the form of a two-dimensional matrix and is shown in Table 9. The approach is same when using fastText with the difference being in the similarity scores and the overall score and so its working is not shown here.

Table 9 (scores obtained with Word2vec)

	pop	and	push
push	0.185	0.034	1.0
and	0.047	1.0	0.034
pop	1.0	0.047	0.185

The algorithm to find the highest similarity score starts at row 0 and column 2. Since it contains the overall highest score with value 1.0, it is taken and the row 0 and column 2 are struck off. The updated matrix is shown in Table 10.

Table 10 (updated after first iteration)

	pop	and	push
push	-	-	-
and	0.047	1.0	-
pop	1.0	0.047	-

In the next iteration the next highest score is taken, which is again 1.0 at row 1 and column 1, so it is taken and the corresponding row and column are struck off.

In the next and final iteration the next highest score from the remaining matrix is taken, which is again 1.0 at row 2 and column 0, so it is taken and the corresponding row and column are struck off.

Since all rows have been considered the algorithm stops and the average similarity score is calculated as

Score(Word2vec) = (1.0+1.0+1.0)/3=1.0

and the grade to be awarded is,

Grade(Word2vec)=1.0*5=5.0

The two human graders have awarded scores of 5 and 5 respectively to this response, and the obtained score of 5.0, is same as the scores awarded by the human judges.

Chapter 5

Experiments and Results

Experiments were carried out for all four of the models on a standard computer science dataset consisting of questions, reference answers and student responses taken from an undergraduate computer science course [10]. Table 11 presents a summary of the dataset used here for the experimental purpose. The dataset consists of a total of 12 assignments containing different number (9 assignments have 7 questions, 1 has 4 and 2 have 10) of questions. Each question (Q) is provided with a reference answer (RA) as well as a set of student answers (SA). The number of questions as well as the number of student responses for each question and for each assignment is provided in the Table 5. Each student response is evaluated by two human scorers and the average of their scores is also provided.

Table 11 (Statistics of the dataset being used)

Assignment No.	**No. Of Questions (Q)**	**No. of SA per Q**	**Total no. of SA**
1	7	29	203
2	7	30	210
3	7	31	217
4	7	30	210
5	4	28	112
6	7	26	182
7	7	26	182
8	7	27	189
9	7	27	189
10	7	24	168
11	10	30	300
12	10	28	280
Total	87	-	2442

The performances of all four models were tested on the 12 assignments and the scores produced by the method are compared with the scores of both the examiners as well as their average to come up with the correlation values which are provided in the Table 12 and Table 13. Table 13 presents the scores obtained using Word2vec trained on Google-News corpus[2], while Table 14 presents the scores obtained using fastText trained on Wikipedia articles. Each of the tables shows the scores using all the four models. *Ex1* and *Ex2* indicate the correlation values between the scores generated by the method and the scores awarded by Examiner 1 and Examiner 2 respectively. Avg indicates the correlation values between the scores generated by the method and the average scores of the two examiners.

[2] **https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit**

As observed from the tables, Model 4 exhibits the best performance, with a correlation value of 0.646, for Assignment 4 with Word2vec on Google-News corpus. On the other hand, Models 1, 2 and 3 all exhibit best performances, with correlation values of 0.625, 0.612 and 0.601, for Assignments 1, 6 and 6 respectively with fastText on English Wikipedia corpus[3]. The overall correlation score over the entire assignment was also computed and it reaches the highest value of 0.805 for Model 4 with Word2vec on Google-News corpus.

Table 12 (Scores with Word2Vec on Google-News Corpus)

Assign-ment	**Model 1**			**Model 2**			**Model 3**			**Model 4**		
	Ex 1	**Ex 2**	**Avg**	**Ex 1**	**Ex 2**	**Avg**	**Ex 1**	**Ex 2**	**Avg**	**Ex 1**	**Ex 2**	**Avg**
1	0.557	0.565	0.608	0.486	0.483	0.525	0.325	0.373	0.377	0.555	0.554	0.602
2	0.218	0.393	0.319	0.208	0.371	0.302	0.041	0.188	0.113	0.224	0.396	0.324
3	0.416	0.426	0.475	0.171	0.137	0.179	0.045	0.045	0.051	0.297	0.211	0.299
4	0.468	0.576	0.546	0.246	0.149	0.227	0.239	0.149	0.223	0.534	**0.646**	0.619
5	0.548	0.357	0.539	0.025	0.022	0.027	0.032	0.017	0.03	0.532	0.415	0.552
6	0.426	0.426	0.579	0.252	0.41	0.374	0.241	0.405	0.365	0.323	0.475	0.453
7	0.375	0.497	0.468	0.064	-0.082	0.002	0.072	-0.075	0.009	0.353	0.437	0.427
8	0.419	0.425	0.463	0.289	0.356	0.341	0.312	0.376	0.365	0.32	0.259	0.331
9	0.421	0.334	0.427	0.408	0.379	0.432	0.411	0.38	0.434	0.521	0.424	0.531
10	0.423	0.383	0.438	0.135	-0.008	0.1	0.139	-0.008	0.103	0.294	0.252	0.3
11	0.374	0.434	0.425	0.224	0.274	0.26	0.219	0.265	0.252	0.358	0.486	0.429
12	0.386	0.438	0.425	0.136	0.168	0.146	0.156	0.187	0.169	0.239	0.269	0.251
Overall	0.652	0.785	0.448	0.358	0.41	0.204	0.395	0.466	0.221	0.639	0.805	0.4

[3] https://dl.fbaipublicfiles.com/fasttext/vectors-wiki/wiki.en.vec

Table 13 (Scores with fastText on Wikipedia Corpus)

Assign-ment	**Model 1**			**Model 2**			**Model 3**			**Model 4**		
	Ex 1	**Ex 2**	**Avg**	**Ex 1**	**Ex 2**	**Avg**	**Ex 1**	**Ex 2**	**Avg**	**Ex 1**	**Ex 2**	**Avg**
1	**0.625**	0.582	0.656	0.487	0.483	0.527	0.483	0.477	0.521	0.632	0.594	0.666
2	0.253	0.371	0.33	0.238	0.412	0.34	0.222	0.397	0.324	0.351	0.478	0.442
3	0.434	0.416	0.484	0.409	0.313	0.422	0.395	0.315	0.413	0.429	0.347	0.451
4	0.524	0.472	0.545	0.327	0.279	0.334	0.319	0.269	0.325	0.535	0.496	0.561
5	0.398	0.346	0.427	0.111	0.087	0.116	0.109	0.068	0.106	0.471	0.444	0.519
6	0.356	0.593	0.536	0.331	**0.612**	0.529	0.325	**0.601**	0.52	0.368	0.624	0.559
7	0.421	0.539	0.518	0.323	0.358	0.37	0.309	0.357	0.361	0.386	0.503	0.478
8	0.443	0.429	0.482	0.295	0.234	0.303	0.329	0.281	0.346	0.364	0.342	0.393
9	0.244	0.167	0.239	0.155	0.113	0.154	0.154	0.112	0.153	0.223	0.162	0.221
10	0.46	0.402	0.472	0.409	0.34	0.414	0.408	0.339	0.413	0.491	0.429	0.503
11	0.559	0.561	0.612	0.487	0.542	0.547	0.479	0.531	0.537	0.558	0.577	0.615
12	0.456	0.459	0.488	0.281	0.223	0.267	0.294	0.238	0.282	0.398	0.377	0.414
Overall	0.686	0.783	0.461	0.611	0.717	0.336	0.609	0.713	0.335	0.687	0.801	0.456

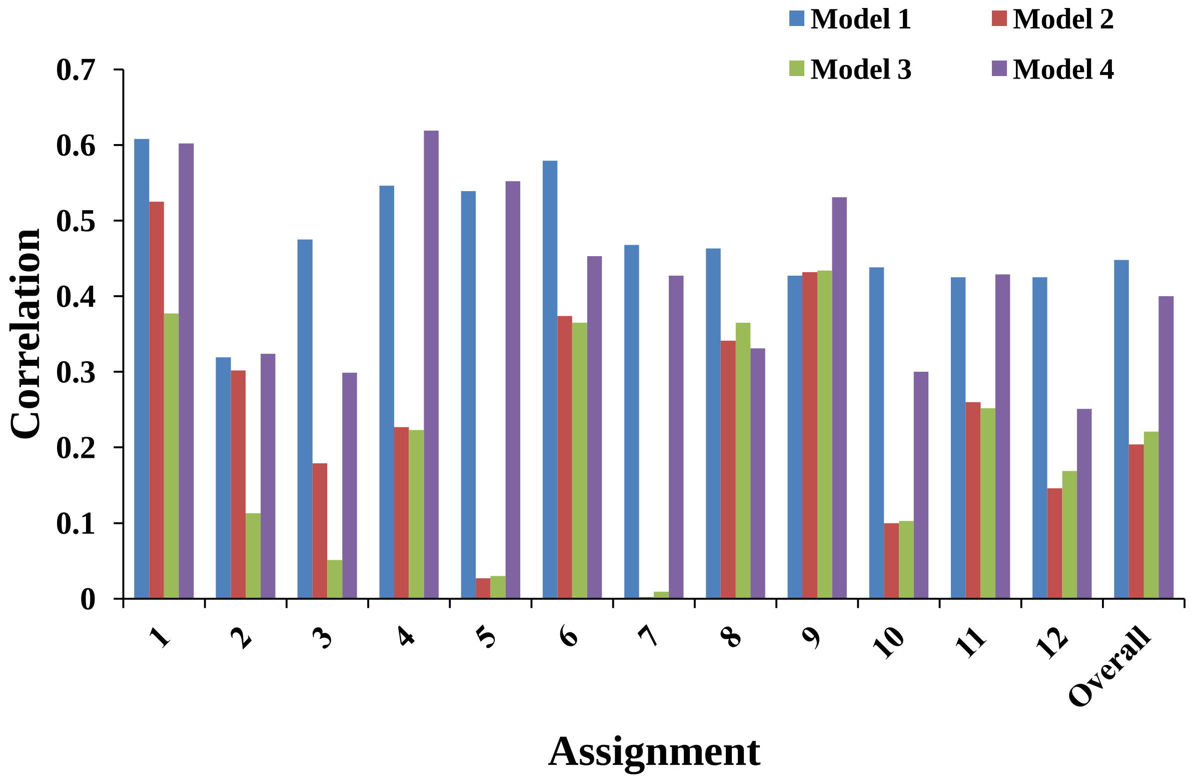


Fig. 5 (Comparison of Correlation using Word2Vec)

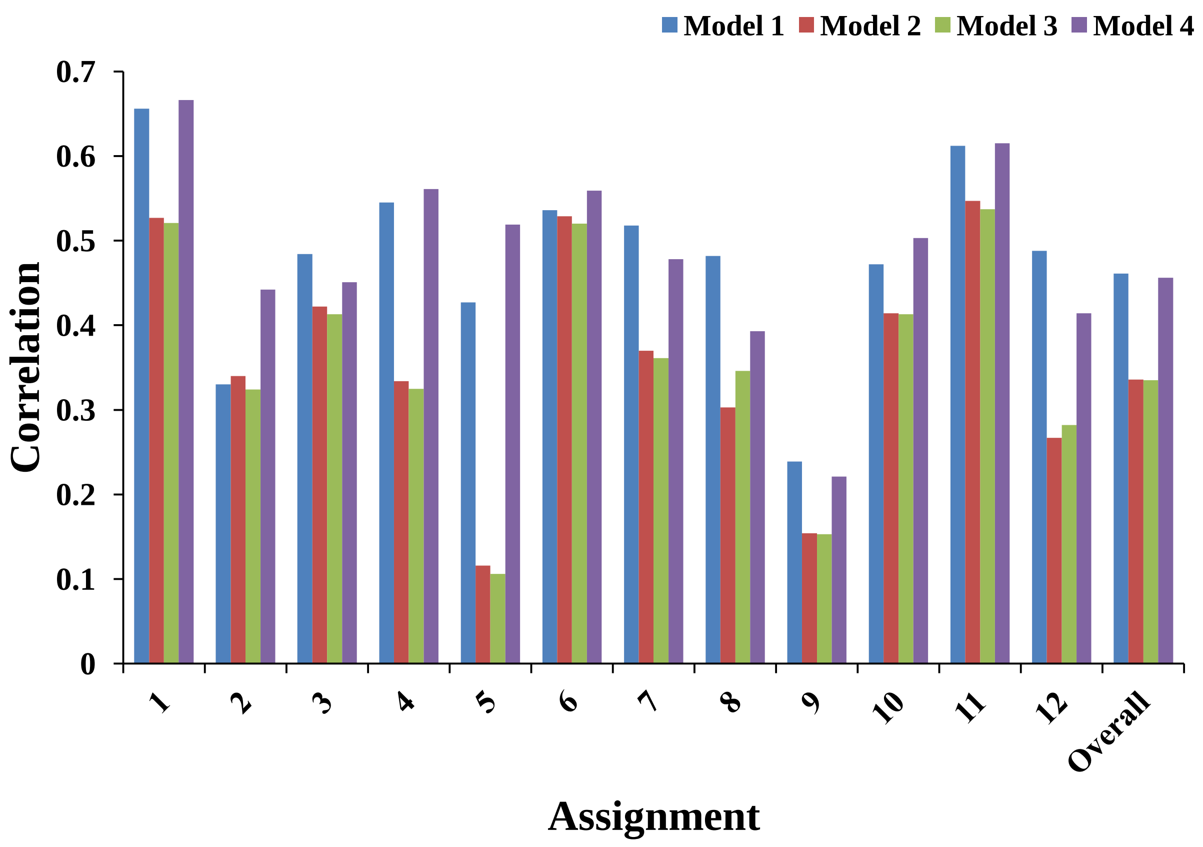


Fig. 6 (Comparison of Correlation using fastText)

Fig. 5 and Fig. 6 present the comparison of correlation values for all the four Models using Word2Vec and fastText based similarity measures respectively. Only the *Avg* of the scores awarded by the two Examiners *Ex1* and *Ex2* has been considered for graphical representation. From Fig. 5 and Fig. 6 it is observed that except for Assignment 9, all four models perform better when using fastText with Wikipedia instead of Google-News corpus. The reason is that fastText can even build vectors for words that are not in the corpus being used by using the vectors for the components that makes up the unknown word. It does so by breaking up the word into n-gram characters (the value of n is decided by the fastText algorithm and not by us), and then using the vector representations of these n-gram components to build the vector for the word. Hence presence of words such as *Enqueue*, which has significance in computer science terminology but not in plain English, will not negatively affect the score generated for the SA containing it.

Another point to be observed is that the correlation values for Assignments 5 and 7 are lower for Models 2 and 3 when Word2vec is used as compared to when fastText is used. The reason is the different working principles of Word2vec and fastText. Word2vec uses the whole word to generate its corresponding vector, while fastText breaks down the word into a group of n-grams, then generates vectors for these individual n-grams and then finally adds them up to come up with a vector for the original word. Therefore when a word that is not present in the corpus is supplied to Word2vec model, it generates an erroneous resultant vector which results in generation of a lower score when the word is compared with any other word.

This is not the case with fastText, as even if the input word is not present in its corpus, it still breaks down the word into a group of n-grams where the n-grams might be present for some other words in the corpus and thus can be used to generate a vector for this word. For example the pair of words (n, array) when checked with Word2vec generates a similarity score of 0.010 while for the same fastText generates a score of 0.231. If such word pairs occur frequently, it will lower the overall similarity score, which in turn will lower the grade to be awarded to the student answer. Such cases are found in much more quantity in Assignments 5 and 7 compared to the rest of the assignments of the underlying dataset.

Chapter 6

Error Analysis

Even though our Models generate scores quite close to that of the human scores, there are some exceptional cases where they generate low scores. One such case is given below, which is taken from Assignment 8:

Q: *What operations would you need to perform to find a given element on a stack?*

RA: *Pop all the elements and store them on another stack until the element is found, then push back all the elements on the original stack.*

SA: *StackPush() StackPop() StackIsEmpty()*

Here the student has answered the question by giving the operations that needs to be performed in terms of function names. Now a human scorer easily understands this, but the method requires a little bit of context to understand this concept. Such cases can be handled in two ways. The first way is to feed the context in which a word or expression is used. In the example provided, the context will be that "StackPush" signifies a function which performs the operation of pushing an element onto a stack. This context can then be used by the models to understand the actual meaning of the response and not treat it as random text with no meaning. The second way is for the models to understand the response by itself without the use of any context provided separately along with the response. This can be achieved by building a knowledge base (KB) of such cases so that the models can consult the KB when they encounter such responses. This will lead to the introduction of the concept of Machine Learning, which is not preferred as the KB will be huge and will only grow as and when such new cases are encountered.

The following simulation will shed some light on why such cases will produce a lower score. The RA and SA are fed into Model 1 and the working is shown below:

Individual words along with their corresponding POS tags for both RA and SA are shown in Table 14.

Table 14 (words with POS tags)

Words in RA	**POS tags**	**Words in SA**	**POS tags**
pop	NN	stackpush	JJ
all	PDT	stackpop	NN
the	DT	stackisempty	NN
elements	NNS	-	-
and	CC	-	-
store	VBP	-	-
them	PRP	-	-
on	IN	-	-
another	DT	-	-
stack	VBP	-	-
until	IN	-	-
element	NN	-	-
is	VBZ	-	-
found	VBN	-	-
then	RB	-	-
push	VB	-	-
back	RB	-	-
original	JJ		

The scores obtained when the model is run with Word2vec are shown in Table 15.

Table 15 (scores obtained with Word2vec)

Word in SA	**Word in RA with equivalent POS**	**Similarity Score**	**Maximum Score taken into consideration**
stackpush(JJ)	original(JJ)	-0.073	-0.073
stackpop(NN)	pop(NN)	0.047	0.047
stackpop(NN)	elements(NNS)	0.006	
stackpop(NN)	element(NN)	-0.073	
stackisempty(NN)	pop(NN)	0.047	0.047
stackisempty(NN)	elements(NNS)	0.006	
stackisempty(NN)	element(NN)	-0.073	

It is to be noted that since the length of SA is lesser than RA, their roles have been reversed and RA is evaluated with respect to SA. Also there are many words in RA having POS tags that are not equivalent to the POS tags of the words in SA, and so they are not shown in the above tables as they won’t be matched.

Now “stackpush” from SA only matches one word in RA with similarity score of -0.073, so this score is considered. For “stackpop” from SA, there are three matching words in RA with similarity scores of 0.047, 0.006 and -0.073 among which 0.047 is maximum of the three, so this score is considered. For “stackisempty” from SA also, there are three matching words in RA with similarity scores of 0.047, 0.006 and -0.073 among which 0.047 is maximum of the three, so this score is considered.

Now that all similarity scores are obtained, the average similarity score is calculated as:

Score(Word2vec) = (0.047+0.047-0.073)/3=0.007

Since the full mark of this Q is 5, the grade that is to be awarded to this SA is calculated as:

Grade(Word2vec)=0.007*5=0.035, which when rounded off to one decimal place gives a value of 0.

The two human graders have awarded scores of 2 and 4 respectively to this response, but one of the models has awarded it a score of 0. So the method failed to score this response correctly, and though it might be said that the obtained score is close to one awarded by first human scorer, it is nowhere close to the one awarded by the second human scorer.

Chapter 7

Conclusions and Future Work

As evident with the correlation values between the scores obtained by the proposed method and the scores awarded by the examiners, all four of our models have proven to be quite efficient in assigning scores to the student answers sufficiently close to the scores awarded by the human examiners. Given a reference answer (RA), our models assign a similarity score (in the range [0, 1]) to the student answer (SA), which is then scaled up with reference to the full marks assigned to the question (Q).

Experiments were carried out on all four of our models on a standard ASAG dataset and the correlation values between the obtained scores and those awarded by the human examiners reveals that the proposed models can competently award the students responses with scores sufficiently close to that of the human examiners.

In some cases, it was observed that the RA has conveyed the answer in very few words, i.e. in a gist, while the student has conveyed almost the same answer with more words by providing some extra information, which may or may not be related to the provided answer (RA). In such cases, if the SA is evaluated with respect to the corresponding RA, this extra piece of information may not get matched to that given in the RA, resulting in assignment of a lower score to the SA than it should have been awarded. Thus, for such cases reversing the role of RA and SA, i.e. evaluating the RA with respect to the SA, is found to be effective as the models generate fairly high scores nearer to those awarded by the human examiners. However, for such cases an assumption was made that the extra piece of information provided by the student is correct, and therefore the provision of penalising the student response for extra information which is incorrect or irrelevant is currently out of the scope of the proposed method.

In future, some advance mechanisms will be augmented to find out whether the extra piece of information is incorrect or not in relevance to the context of the answer, which will help us with the decision of penalising the student answer or not.

Apart from the rules that have been used in the models for the purpose of matching the RA and SA, the efficiency of the proposed models is upper bounded by the semantic similarity scores produced by the corpus-based similarity measures word2vec and fastText. The similarity scores returned by these similarity modules play an important role in assigning scores nearer to the human graders. Incorrect scores produced by them can adversely impact the performance of the models.

There are several RA and SA in the dataset having spelling errors. For a fair judgement, we have evaluated the performance of the models without correcting those errors in the dataset. In future, we will try to add spell checker to get rid of these errors, which should improve the performance of the method to some extent.

References

[1] W. H. Gomaa, A. A. Fahmy. Short Answer Grading Using String Similarity and Corpus-Based Similarity, International Journal of Advanced Computer Science and Applications, Vol. 3, No. 11, 2012.

[2] O. Adams, S. Roy, R. Krishnapuram. Distributed Vector Representations for Automatic Short Answer Grading, Proceedings of the 3rd Workshop on Natural Language Processing Techniques for Educational Applications, pp. 20–29, Osaka, Japan, December 12 2016.

[3] S. T. Alotaibi,and A. A. Mirza. (2012). HYBRID Approach For Automatic Short Answer Marking.

[4] S. Roy, S. Dandapat, A. Nagesh, and Y. Narahari. Wisdom of students: A consistent automatic short answer grading technique. In Proceedings of 13th ICON-2016, pp. 178, 2016.

[5]M. A. Sultan, C. Salazar, T. Sumner: Fast and easy short answer grading with high accuracy. In: NAACL: HLT, San Diego, California (2016), pp. 1070–1075

[6] D. Pérez, A. M. Gliozzo, C. Strapparava,E. Alfonseca,P. Rodríguez,B. Magnini. Automatic assessment of students' free-text answers underpinned by the combination of a BLEU-inspired algorithm and latent semantic analysis. In D. Cook, L. Holder, I. Russell, Z. Markov (Eds.), *Proceedings of the 18th international florida artificial intelligence research society conference* (pp. 358–363). Clearwater Beach: AAAI Press.

[7] L. Bachman, N. Carr, G. Kamei, M. Kim, M. Pan, C. Salvador & Y. Sawaki (2002). A Reliable Approach to Automatic Assessment of Short Answer Free Responses. In Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002). pp. 1–4.

[8] C. Leacock, and M. Chodorow. (2003) C-rater: Automated Scoring of Short-Answer Questions. *Computers and the Humanities*, 37(4), pp. 389-405.

[9] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55-60.

[10] M. Mohler, R. Bunescu, and R. Mihalcea, Learning to grade short answer questions using semantic similarity measures and dependency graph alignments, HLT: 49th Annual Meeting of the ACL: Human Language Technologies, vol, 1, pp. 752-762, 2011.

[11] R. Basak, S. Naskar,and A. Gelbukh. (2019). Short-answer grading using textual entailment. Journal of Intelligent & Fuzzy Systems, vol. 36, no. 5, pp. 4909-4919, 2019.

[12] R. Basak,S. Naskar,P. Pakray,and A. Gelbukh. (2015). Recognizing Textual Entailment by Soft Dependency Tree Matching. Computacion y Sistemas. 19. 685–700. 10.13053/CyS-19-4-2331.