Automatic Short Answer Scoring based on Paragraph Embeddings

Sarah Hassan[1], Aly A. Fahmy[2], Mohammad El-Ramly[3]
Computer Science Department
Faculty of Computers and Information, Cairo University
Cairo, Egypt

***Abstract*—Automatic scoring systems for students' short answers can eliminate from instructors the burden of grading large number of test questions and facilitate performing even more assessments during lectures especially when number of students is large. This paper presents a supervised learning approach for short answer automatic scoring based on paragraph embeddings. We review significant deep learning based models for generating paragraph embeddings and present a detailed empirical study of how the choice of paragraph embedding model influences accuracy in the task of automatic scoring.**

Keywords—Automatic scoring; short answer; Pearson correlation coefficient; RMSE; deep learning

I. Introduction

Improving the quality of education is always a desired goal in educational institutions. In higher education institutions, many courses are given in large classrooms where number of attending students is large. Such large learning environments present special challenges on instructors and one of these challenges is students' assessments. One technology solution for this problem is automatic scoring of students' answers. Automatic short answer scoring is the task of "assessing short natural language responses to objective questions using computational methods" [1]. This eliminates from instructors the burden of grading large number of test questions and facilitates performing even more assessments during lectures. Different types of questions are used in assessments such as multiple choice questions, true/false questions, numeric answer questions, short answers questions, and essay questions. Short answer and essay answer questions require more complicated work related to text processing and analysis; while automatic scoring of other types can be easy and direct task. This paper is concerned with automatic scoring of short answers. An answer is considered short answer if its length approximately ranges from one phrase to one paragraph [1].

The proposed model employs deep learning method named paragraph embedding on students' answers and reference answers to generate vector representation of answers. Cosine similarity measure between vectors of students' answers and reference answer is used as a feature vector to train regression classifier for predicting students' scores.

Paragraph embedding - also referred to as Paragraph vectors [2], sentence encoders [3] - is the method of generating numeric fixed-length vector representations for variable length pieces of texts [2].

There are two general types for paragraph embedding models. The first one is based on applying mathematical operations (e.g. sum, average) on retrieved word vectors for all words in a given paragraph in order to generate the paragraph vector. The second approach is training a model to infer the paragraph vector for a given sentence. We compared different state-of-the-art techniques from the two types by employing them to solve automatic short answer scoring problem. The objective of this study is to apply a comprehensive evaluation of multiple state-of-the-art paragraph embedding models by applying them to the task of short answer automatic scoring. And consequently, fill the present gap in the literature for this regard.

This paper is organized as follows:

- Section II presents related work of automatic short answer scoring algorithms.
- Section III briefly explains the proposed methods used for text modeling and text similarity.
- Section IV presents the used Dataset.
- Section V describes the models used in this research.
- Section VI shows experiments results and discussion.
- Section VII presents conclusion and future work.

II. Related Work

Different approaches are used to solve automatic short answering problem. The first approach is based on using unsupervised techniques that combine text-text similarity measures to compare student answer and reference answer and predict score based on similarity value.

Survey [4] presents 3 types of text-text similarity measures. The first type is string based similarity measures which work on string sequences (term-based similarity) or character composition (character-based similarity). The second type is corpus based similarity measures which are semantic measures that find similarity between words according to information gained from large corpora. The third type is knowledge based similarity measures which are also semantic similarity measures that are based on semantic networks.

Reference [5] compared number of knowledge-based and corpus-based measures of semantic similarity. These measures were applied on different domain specific corpuses with different sizes to examine the effect of domain and corpus size.

The researchers also introduced a method to feedback model with students' answers to solve the problem of correct student answers that are not similar to the reference answer. The best achieved correlation coefficient value was obtained using LSA corpus based similarity applied on domain-specific corpus built on Wikipedia along with feedback from students answers. Correlation coefficient value is 0.5099 measured by comparing grade assigned to every student answer with actual grade; and 0.6735 measured by comparing total student scores with actual totals of grades per student.

Reference [6] tested different text-text similarity measures and combined measures that gave best results to get overall similarity. The different measures were tested on student answers dataset provided at [7], the highest correlation coefficient achieved when comparing students' answers with reference answers was 0.504. This result was achieved by combining N-gram character based similarity with DISCO [8] first order corpus based similarity applied using Wikipedia data packets.

A second approach is training machine learning algorithms to predict scores given set of calculated features.

Reference [9] presents two supervised learning models. The first model is a regression model trained to predict students' scores. The second model is a multi-class classifier trained to predict the labels of student' answers (e.g. correct, incorrect, or contradictory). The models were trained on eight calculated features. The first three features are based on text similarity between student answer and reference answer calculated in three different methods. The second three features are the same three text similarity measures but calculated after removing question text words from both reference answer and student answer. The seventh feature is calculated after applying term weighting based on variant of tf-idf. The final feature is the ratio of number of words in student answer to that in reference answer. The first model is trained on student answers dataset provided at [7] and the second model is trained on SemEval-2013 task [10]. The regression model achieved correlation coefficient of 0.592 and RMSE of 0.887 when tested on out-of-domain data. The model achieved 0.63 correlation coefficient value and .85 RMSE when tested on in-domain data. The classification model achieved F1 score of 0.550.

A third approach is employing deep learning architectures that enable multi-level automatic feature representation learning. These architectures are increasingly used in the past years as they showed superior results in various NLP tasks [11].

Several paragraph embedding models are built based on deep learning architectures and now became the state-of-the-art methods for NLP problems. Reference [11] provides a review of different significant deep learning models applied in NLP tasks. For the task of paragraph embedding, multiple models were presented. Convolution Neural Networks (CNN) which proved their effectiveness in many computer vision tasks had become the natural choice in NLP tasks with the need for models that can extract high level features form sequences of words. Recurrent Neural Networks (RNN) is another type of deep learning models that become widely used in NLP tasks. Its advantage comes from its nature of memorizing previous computations and using them in current processing. In the context of word sequences, this allows it to capture the inherent sequential nature present in language. A third type of deep learning models is Recursive Neural Networks that is based on the argument that language exhibits a natural recursive structure. I.e. words and sub-phrases are combined into phrases in a tree structure manner. A forth type is Deep Reinforcement Learning models which are applied in NLP problems related to language generation. A fifth type that gained some interest recently is based on merging neural networks with a form of memory that the model can interact with. This type is called Memory Augmented Networks. Word Embeddings trained on large unlabeled corpora provide distributional vector representation of words. These representations have the advantage that they capture semantic meanings of words. The semantic similarities between words can then be measured with simple methods such as cosine similarity. Word embeddings are often used as the input layer for deep learning models. [11]. In this paper, some state-of-the-art deep learning based models trained for NLP tasks are presented and evaluated in the context automatic short answer scoring problem.

III. Vector Representation and Text Similarity Measures

In this paper, the focus is on two approaches that can be used to generate vector representations of short answers, i.e. paragraph embeddings. The first approach is generating the paragraph vector of an answer by calculating the sum of word vectors for words in the answer. The second approach is training a deep learning model to directly infer the paragraph vector of a given answer.

A. Words Embeddings

Word embeddings are techniques for learning vector representations for words. Word embedding models are usually trained on large unlabeled corpora in order to exploit their benefits [12]. Multiple models for word embeddings trained on large corpus of data are publically available. These pre-trained models have the advantage of storing semantic relationship between words. In this paper, we tested 4 different pre-trained word vector models for generating paragraph vectors and applied results in task of automatic short answering.

The method is: given a short answer (student answer or reference answer); the objective is to generate the paragraph vector of the answer. First, the answer is tokenized to get a list of its words and any necessary text processing is applied (e.g. removing punctuation marks). Second, given word vector model, the corresponding word vector for each word is retrieved. Finally, to get the paragraph vector, sum operation (which is commonly used in all reviewed literature) is applied on all word vectors to get a single vector representing the paragraph vector. Fig.1 demonstrates this method.

Below we introduce word embedding methods chosen for this study.

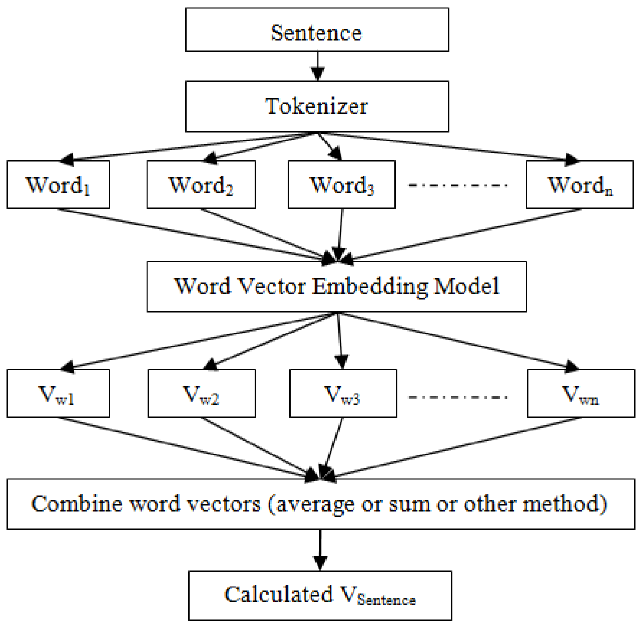
Sentence
Tokenizer
Word$_1$
Word$_2$
Word$_3$
Word$_n$
Word Vector Embedding Model
V$_{w1}$
V$_{w2}$
V$_{w3}$
V$_{wn}$
Combine word vectors (average or sum or other method)
Calculated V$_{Sentence}$

Fig. 1. Generating Paragraph Vectors from Pre-Trained Word Vector Models.

1) Word2Vec: The word2vec model [2] is based on neural network trained to predict a word given the context of surrounding words. One of the features of this model is that after training, words with similar meanings are mapped to similar positions in vector space. Also, some semantic relationships between words can be inferred by applying simple mathematical calculations. For example: "King" - "man" +"woman" = "Queen". Google [1] provides 300-dimensional pre-trained word2vec model. The model was trained on part of Google News dataset (approximately 100 billion words). The model provides word vectors for 3 million words and phrases.

2) GloVe [2]: GloVe (Global Vectors) [13] is a word embedding model that combines features from two major models for generating word vectors: global matrix factorization and local context window (word2vec is an example of local context window model). This combination of features from the two models' types allows taking advantages from both models as well as overcoming some of their drawbacks. We used 300-dimensional pre-trained word vectors trained on a combination of Gigaword5 and 2014 dump of English Wikipedia. The model provides word vectors for 400,000 tokens.

3) Fasttext[3]: Fasttext model [12] is based on skip-gram model where each word is represented as a bag of its character n-grams. A vector representation is associated to each character n-gram; and word vector is computed as the sum of the n-gram vector representations. We used 300-dimensional pre-trained word vectors model that provides 2 million word vectors trained on Common Crawl.

4) Elmo[4]: Elmo (Embeddings from Language Models) [14] is a deep contextualized word representation that models (1) word syntax and semantics and (2) word uses across different linguistic contexts (polysemy). The learned word vectors are function of the internal state of deep bidirectional language model (biLM) rather than just using that top LSTM layer. We used 1024-dimensional word vectors pre-trained on 1 billion word benchmark5.

B. Paragraph Embeddings

In paragraph embedding models, deep learning model is trained on sequences of text to directly learn and infer vector representation of variable-size sequence. Fig.2 demonstrates this method.

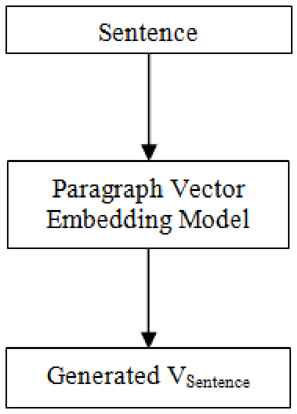

Fig. 2. Generating Paragraph Vectors from Paragraph Embedding Models.

Below we introduce paragraph embedding methods chosen for this study.

1) Doc2vec: With this model, a neural network classifier with stochastic gradient descent algorithm is trained on a fixed-width sliding window over words of paragraphs. Paragraph vectors are learned along with word vectors so the trained model can be used to infer paragraph vectors or word vectors [2] [15].

Gensim Doc2vec [16] is a python library that provides an implementation of paragraph vectors model. It is designed to model word sequences ranging from n-gram sentence, paragraph, or document. We trained 300-dimensional doc2vec model on our benchmark dataset used for addressing automatic short answering problem.

2) InferSent[6]: InferSent [3]is a sentence encoder model for learning universal representation of sentences. The encoder model is based on bidirectional LSTM architecture with max pooling trained on the supervised data of the Stanford Natural Language Inference datasets (SNLI). We used two 4096-dimensional pre-trained models, one uses fasttext pre-trained word vectors for representing words in sentences and the second uses GloVe pre-trained word vectors.

[1] https://code.google.com/archive/p/word2vec/
[2] https://nlp.stanford.edu/projects/glove/
[3] https://github.com/facebookresearch/fastText

[4] https://allennlp.org/elmo
[5] http://www.statmt.org/lm-benchmark/
[6] https://github.com/facebookresearch/InferSent

3) Skip-Thoughts [7] : Skip-thoughts model is an unsupervised learning model for sentence encoding, i.e. mapping a sentence composed of words to sentence vector. The model is trained to generate sentence vectors using an approach similar to skip-gram model [17] but works on sentence level instead of word level. So given a training corpus of contiguous text, the model is trained to predict surrounding context sentences of a given sentence. The model also provides a vocabulary expansion method to encode new words that were not seen in the training phase [18]. We used 2400-dimensional pre-trained skip-thought model where sentence vectors were trained on BookCorpus dataset [19].

C. Text Similarity Measure

For similarity between paragraph vectors, cosine similarity method is used. Cosine similarity of vectors is the cosine of the angle between vectors in inner-product space [20]. This value has the property of being 1.0 for identical vectors and 0.0 for orthogonal vectors. Cosine similarity can also be calculated by applying the inner-product between vectors of unit length.

IV. The Dataset

The benchmark dataset used is short answer grading dataset V2.0[8]. This dataset consists of ten assignments between four to seven questions each and two exams with ten questions each. Total number of questions is 87; six of them are not short answer questions so they were excluded from the authors' published work. The experiments in this paper also were applied only on the 81 short answer questions. The number of students' answers per question ranges from 24 to 31 with average of 28 answers and the total number of short answers is 2273.

These assignments/exams were provided at an introductory computer science class at the University of North Texas. Elements of the dataset are questions' texts, the reference answer for each question, and students' answers. The answers were graded by two different graders; both grades of grader1 and grader 2 along with the average grade of the two graders are provided for each answer. All three types of grades are in range 0 to 5 [7]. This research works on average of grades following other researchers [7, 9, 6].

TABLE I. Sample Question, Reference Answer and Students' Answers

Question	What is typically included in a class definition?	
Reference Answer	Data members (attributes) and member functions	
Students Answers And Average Grades		
Student 1 Answer	Data members and member functions	5
Student 2 Answer	the keyword class followed by they class name, on the inside you declare public and private declarations of your class	3.5
Student 2 Answer	Class name, two curly prenthesis, public and private	2

[7] https://github.com/ryankiros/skip-thoughts
[8] http://lit.csci.unt.edu/index.php/Downloads

Table I shows sample question with its reference answer and 3 samples of students' answers with their average grades. Prior to feeding students answers and reference answers for the different paragraph embedding models explained in section III, primitive sentence pre-processing techniques were applied to extract tokens after removing punctuation marks and stop words.

V. Methods

In order to compare the accuracy of the various models listed in section III; these models are tested for the task of automatic short answer scoring. Steps are as follow (applied separately for each paragraph embedding model):

A. Generate Paragraph Vector

For each answer (student answer and reference answer), the paragraph vector is retrieved using each of the models listed in section III. In the case of using sum of pre-trained word embeddings, the first step is to load a pre-trained file containing dictionary of words with their word vectors into memory. Then, words of the answer are matched with words in dictionary to get list of word vectors for found words. This is the case with GloVe, Google word2vec, and fasttext embeddings. For Elmo embeddings, a tensorflow hub[9] is used to load online trained model.

For the case of using direct training of paragraph embedding model, settings vary based on the model. We trained doc2vec directly on tokenized students answers and reference answers on order to learn vector representation of them. InferSent provides a pre-trained sentence encoder model that can be used directly to infer paragraph vector of a given sentence. As the training phase requires word vectors to be used as input layer of the deep learning model, the pre-trained model comes with two versions, one trained with Glove word vectors and the second trained with fasttext word vectors. Skip thoughts also provides pre-trained model sentence encoder. The pre-trained model comes with multiple files that can be loaded and used directly to encode sentences to paragraph vectors.

Tables II and III shows the different file sizes of pre-trained embedding models used from the two types.

TABLE II. Disk consumption of pre-trained Word Vector Models

Word Embedding Model	Pre-trained Word Vectors File size
GloVe	989 MB
Google word2vec	3.39 GB
Fasttext	4.20 GB

TABLE III. Disk consumption of pre-trained Paragraph Embedding Models

Paragraph Embedding Model	Pre-trained Model File size
Doc2vec	Didn't use saved models
InferSent	146 MB
Skip thoughts	Multiple files with total size of 5.25 GB

[9] https://tfhub.dev/google/elmo/2

B. *Calculate Cosine Similarity*

For each student answer-reference answer pair, we calculate cosine similarity of their corresponding paragraph vectors.

C. *Train Regression Classifier*

We used cosine similarity measure as a feature vector to train Ridge regression classifier model for predicting students' scores. A train/test split of 85% for training data and 25% for testing data is used.

D. *Measure Accuracy*

Calculate Pearson Correlation Coefficient and RMSE for predicted scores and actual grades.

VI. Experiments and Discussion

Table IV shows the results of applied methods explained in section V. The table shows that the best correlation coefficient result is 0.569 and the best RMSE value is 0.797. Both values were achieved by training doc2vec model only on sentences from the dataset to generate their paragraph vector. But this raises the question: will this model produce same results when tested on new unseen data from same domain of questions? Out of the other models, fasttext achieved best correlation coefficient value (0.519) and Google word2vec achieved best RMSE (0.821) value which requires further investigation as trained paragraph embedding models claim to achieve best state-of-the-art results. From memory consumption perspective, GloVe based paragraph embedding model, InfeSent model, and Doc2vec model provided best results with least amount of memory needed to load pre-trained models. Elmo model consumed the largest amount of memory and time for running the model and yet didn't achieve the best results.

TABLE IV. Results of Regression Classifier Trained on Similarity Between Paragraph Vector of Student Answers And Reference Answers

Paragraph Embedding Method	Model	Dim	Pearson Correlation Coefficient	RMSE
Sum of pre-trained word vector model	GloVe	300	0.507	0.838
	Google word2vec	300	0.532	0.821
	FastText	300	0.519	0.831
	Elmo	1024	0.390	0.896
Training of paragraph vectors model	doc2vec	300	0.569	0.797
	InferSent with Glove word vectors	4096	0.506	0.843
	InferSent with fasttext word vectors	4096	0.4597	0.862
	Skip thoughts	2400	0.468	0.861

Researches [6] and [9] tested their models on same dataset and provided the same accuracy measures as ours. Research [6] which apply direct text-text similarity between student answer and reference answer to predict score achieved correlation coefficient value of 0.504 compared to 0.569 reported in table IV. We couldn't compare with the RMSE value because it was not included in the mentioned paper.

Research [9] presented result of 0.63 correlation coefficient value and 0.85 RMSE by training a regression classification model. All models tested in this paper shows comparable results for RMSE but fewer results in the correlation coefficient. We emphasize on that classification model in reference [9] uses multiple feature vectors for classification task including cosine similarity of off-the-shelf word embeddings. Authors didn't provide test measurements for the effect of each feature vector separately .The classification task in our model uses only one feature vector which is cosine similarity of paragraph vectors.

VII. Conclusion and Future Work

In this study, seven different models for embedding short answer text were evaluated. 4 are based on sum of pre-trained word vectors and 3 are based on trained deep learning model for inferring paragraph vectors. The models were evaluated in the context of the automatic short answer scoring task and the study reveals that using pre-rained models achieved comparable results for the task of automatic short answer scoring.

A Forthcoming paper aims to apply the same methods to other short answer scoring datasets to see if similar results will be achieved. Also, to investigate the impact of word vectors combination new operators (such as weighted sum) and considering the additional use of non-embedding features on the correlation and RMSE values.

References

[1] S. Burrows, I. Gurevych and B. Stein, "The eras and trends of automatic short answer grading," International Journal of Artificial Intelligence in Education, vol. 15, no. 1, pp. 60-117, 2015.

[2] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *In Proceedings of the 31st International Conference on Machine Learning (ICML 2014)*, Beijing, China, 2014.

[3] D. K. H. S. B. B. Alexis Conneau, "Supervised learning of universal sentence representations from natural language inference data," *arXiv preprint arXiv*, vol. 1705, no. 02364, 2017.

[4] A. A. F. Wael H Gomaa, "A Survey of text similarity approaches," *International Journal of Computer Applications68*, pp. 13-18, 2013.

[5] M. Mohlerl and R. Mihalcea, "Text-to-text semantic similarity for automatic short answer grading," in *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, Athens, Greece, 2009.

[6] W. H. Gomaa and A. A. Fahmy, "Short Answer Grading Using String Similarity And Corpus-Based Similarity," *International Journal of Advanced Computer Science and Applications(IJACSA)*, pp. 115-121, 2012.

[7] M. Mohler, R. Bunescu and R. Mihalcea, "Learning to Grade Short Answer Questions using Semantic Similarity Measures and Dependency Graph Alignments," in *Proceedings of the 49th Annual Meeting of the Association of Computational Linguistics - Human Language Technologies (ACL HLT 2011)*, Portland, Oregon, 2011.

[8] P. Kolb, "DISCO: A Multilingual Database of Distributionally Similar Words," *In Proceedings of KONVENS-2008*, 2008.

[9] S. M. Arafat, C. Salazar and T. Sumner, "Fast and easy short answer grading with high accuracy.," in Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies., San Diego, California, 2016.

[10] Dzikovska, M. O., R. D. Nielsen and C. Leacock, "The joint student response analysis and recognizing textual entailment challenge: making sense of student responses in educational applications.," *Language Resources and Evaluation,* vol. 50, no. 1, pp. 67-93, 2016.

[11] D. H. S. P. E. C. Tom Youngy, "Recent trends in deep learning based natural language processing," *arXiv preprint arXiv:,* vol. 1708, no. 02709, 2017.

[12] P. Bojanowski, E. Grave, A. Joulin and T. Mikolov, "Enriching word vectors with subword information," *arXiv preprint arXiv,* vol. 1607, no. 04606 , 2016.

[13] Pennington, Jeffrey, R. Socher and C. Manning, "Glove: Global vectors for word representation.," 2014.

[14] M. N. M. I. M. G. Matthew E. Petersy, "Deep contextualized word representations," *arXiv preprint arXiv,* vol. 1802, no. 05365, 2018.

[15] J. H. Lau and T. Baldwin, "An empirical evaluation of doc2vec with practical insights into document embedding generation," arXiv preprint arXiv:1607.05368, 2016.

[16] R. Rehurek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, Valletta, Malta, 2010.

[17] T. Mikolov, K. Chen, G. Corrado and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv,* vol. 1301, no. 3781, 2013.

[18] K. Ryan, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba and S. Fidler, "Skip-thought vectors," *Advances in neural information processing systems,* pp. 3294-3302, 2015.

[19] R. K. R. Z. R. S. R. U. A. T. S. F. Yukun Zhu, "Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books.," *arXiv preprint arXiv,* vol. 1506, no. 06724, 2015.

[20] A. Singhal, "Modern information retrieval: A brief overview.," *IEEE Data Eng. Bull.,* vol. 24(4), pp. 35-43, 2001.