Question Answering Chatbot using Deep Learning with NLP

Devanshi Singh
Department of Computer Science and Engineering
National Institute of Technology, Tiruchirappalli
Tamil Nadu, India
devanshisingh911999@gmail.com

K. Rebecca Suraksha
Department of Computer Science and Engineering
National Institute of Technology, Tiruchirappalli
Tamil Nadu, India
krebesuraksha1523@gmail.com

S. Jaya Nirmala
Department of Computer Science and Engineering
National Institute of Technology, Tiruchirappalli
Tamil Nadu, India
sjaya@nitt.edu

Abstract— **In spite of the number of techniques, models and datasets, Question Answering is still an exacting problem because of the issues in understanding the question and extracting the correct answer. It refers to creating platforms that when given a question in a natural language by humans, can automatically answer it. While many information retrieval chatbots achieve the task, recently, deep learning has earned a lot of attention to question answering due to its capability to learn optimal representation for the given task. This paper aims to build a closed domain, factoid Question Answering system. We recruit NLP methods of pattern matching and information retrieval to create an answer candidate pool. Before scoring similarities between the question and answers, we map them into some feature space. Our approach solves this task through distributional representations of the words and sentences wherein encodings store their lexical, semantic, and syntactic aspects. We use a convolutional neural network architecture to rank these candidate answers. Our model learns an optimal representation for the input question and answer sentences and a matching function to relate each such pair in a supervised manner from training data. Our model does not require any manual feature engineering or language sensitive data; hence can be extended to various domains. Training and testing on TREC QA, a Question Answering dataset, showed very promising metrics for our model.**

Keywords—Question Answering, Deep Learning, Information Retrieval, BM25, Stanford CorefAnnotator, word embeddings, Convolutional Neural Networks, TREC QA.

I. Introduction

Question Answering (QA) chatbots have become robust systems for answering the questions asked in a natural language used by humans. These systems can be divided into two groups. One group extracts the answers from a pre-structured database after transforming the question into a database query and the other group focuses on closed/open-domain QA wherein information retrieval (IR) is employed to find answers from a group of natural language documents.

Closed domain question answering refers to answering questions that are relevant under a specific domain, e.g., medicine, insurance, etc. This paper proposes a QA chatbot for the education domain to help students with their queries about subjects and courses and put unnecessary load off teachers. Such chatbots are revolutionary when it comes to student interaction, collaboration, and how they absorb information. Closed domain QA requires multiple intermediate steps. The first is to create a knowledge base that exploits the specific domain's specialty, followed by collecting the most probable answer documents and selecting the best one. The last step requires reranking the selected documents, which is possible with the advent of Deep Learning models.

The proposed question answering chatbot allows users to upload relevant files that serve as the context for our QA system. The file has to undergo several NLP transformations before it gets transformed to a knowledge base and acts as context. Upon receiving a query from the user, it employs information retrieval methods to find the most probable answer sentences. Finally, with the help of a convolutional neural networks (CNNs) based ranking model, potential answers are ranked, and the top answer is chosen. The re-ranker encodes input QA pairs using their lexical, semantic, and syntactic features and then calculates the similarity between their encodings. However, the choice of features and the representations of these encodings is an experimental process. Semantic text matching is a tedious task requiring a lot of parsers, lexicons, and knowledge bases. This problem is addressed using distributional word matching, for which lexical and semantic resources are used for matching question and possible answers.

Deep Learning methods not only perform distributional word similarity check between sentences but also learn the optimal encodings for them. In particular, CNNs have proved to efficiently map input sentences into low dimensional vectors while keeping their syntactic and semantic information intact. Our sentence model is based on a CNN architecture that has shown state-of-the-art results on many NLP sentence classification tasks [1, 2]. But we use this only for generating an intermediate representation of the input sentences. To calculate the similarity score, we use a method used by the model by Yu et al. [3] using deep learning, that has showed state-of-the-art results on answer sentence selection. Our model provides a means of considering the interaction factor between the intermediate representations and the similarity score to create richer representations. This paper, to some extent, addresses the issues of linguistic gaps, context awareness, and ambiguity that occur in user-provided knowledge base systems.

This paper is divided into five sections, described as follows. Section 2 explores some related works in the field. Section 3 provides comprehensive discussions of the information flow and the proposed deep learning model for the chatbot. Section 4 compares our model's performance against existing models and explores the effect of certain functionalities on our model. Section 5 concludes our work. Section 6 discusses the future scope of the work.

II. Related Work

A. *Chatbots in Education*

Many chatbots [4] have been developed in the past decade that have helped both students and teachers. Percy [5] answers queries from Piazza, an online forum using linear SVM, CSIEC [6] allows students to learn English by conversing with them in natural language. Many schools have developed their own chatbots that carry and pass over information related to their courses, events, etc. Michelle, of the University of People, Pounce [7], of Georgia State University are amongst many.

B. *Using distributional representations in Natural Language Processing*

Compared to string or logical forms of words, distributed representations better capture latent semantic information and are better for checking similarity. They are showed to be fruitful in applications like relation extraction [8] and word sense disambiguation [9]**.** In many cases, distributed representations for sentences and phrases are needed instead of words. Relevant ideas for this involve using category theory [10], parse trees with recursive auto-encoders [11,12], and CNNs [13, 1]. The CNNs approach will be the focus of this paper.

C. *Applying Deep Learning in Question Answering(QA)*

Due to their versatility, researchers have started to use neural networks for QA. Yih et al. [14] built a single relation question answering which has a knowledge base of triplets, Bordes et al. [15, 16] constructed a Siamese network to plot query and answer pairs into a joint space. Xiong et al. [17] used memory and attention mechanisms to simulate reasoning capabilities and tested them on Insurance data. Tan et al. [18] built embeddings for input text using bidirectional long short-term memory (biLSTM) and measured relatedness using cosine similarity. Abacha et al. [19] proposed a novel approach of mapping new queries to previously answered questions that are "similar" based on Recognizing Question Entailment (RQE) and tested it on medical data.

III. Proposed Work

The question answering chatbot works in three phases: creating a knowledge base, creating an answer candidate pool, and selecting the best answer sentence.

A. *Creating the knowledge base*

The chatbot allows users to upload relevant documents in PDF format. The text from PDF is extracted. This text is raw and undergoes cleaning, which includes removing extra spaces and tabs, irrelevant phrases like URLs, irrelevant words such as "Chapter," "Table," "Figure," etc., that are generally present in a textbook. Stanford's CoreNLP functionality CorefAnnotator [20] provides coreference mapping between words in adjacent sentences to implement context-awareness among sentences. Next, this text undergoes several NLP transformations such as tokenization, lemmatization, singularization, punctuation removal, etc., to reduce the vocabulary size in the final knowledge base. Finally, this text is saved as a knowledge base that is essentially a collection of sentences from the textbook.

B. *Creating the answer candidate pool*

This phase takes user query as input, which undergoes text cleaning and preprocessing steps similar to the PDF file. The chatbot employs the following information retrieval methods to check the similarity between the question and sentences in the knowledge base.

- *Inverse Document Frequency (IDF) creation:* Every sentence in the knowledge base is considered a document, hence the words 'document' and 'answer sentence' will be used interchangeably. We compute the IDF score for each word in the documents to signify the importance of a word in the document and reduce the importance of common words such as 'the', 'is', 'a' etc., using Eq. 1,

$$idf[word] = \log\left(\frac{N - freq[word] + 0.5}{freq[word] + 0.5}\right) \tag{1}$$

  where *N* is number of documents in the knowledge base, *freq[word]* is the number of documents in which 'word' appear.

- *BM25 Similarity Score calculation:* BM25 score is calculated using Eq. 2. between the query and each sentence in the knowledge base. The top 10 sentences based on the BM25 score are saved as the answer candidate pool for the given question,

$$score+ = \left(\frac{idf[word] * freq[word] * (k+1)}{freq[word] + k * \left(1 - b + b * \frac{docLen}{avg.docLen}\right)}\right) \tag{2}$$

where *idf[word]* is the computed IDF score for each word in the current document, *freq[word]* is the frequency of the word in that document, *docLen* and *avg.docLen* are length and average length in terms of number of words in the current answer sentence. *k* and *b* are parameters having value 1.2 and 0.75, respectively. Score is the BM25, calculated for each sentence, iterated over every word in it.

C. Selecting the best answer sentence

The answer pool and question are fed to our neural network. Fig 1. portrays the complete architecture of our model.

1) Problem Formulation: The task of selecting best asnwer sentence can be looked upon as binary classification. We can denote the given question as q and its associated set of candidate answer sentences as $A = \{a_1, a_2, ... a_n\}$. Then each sentence a_i can be associated with a judgement value y_i as given in Eq. 3. Hence a A is associated with a label set Y,

$$y_i = \begin{cases} 1, & if\ a_i\ is\ correct\ answer\ to\ q \\ 0, & if\ a_i\ wrong\ answer\ to\ q \end{cases} \tag{3}$$

$Y = \{y_1, y_2, ... y_n\}$. Hence, our goal is to train a classifier over the datapoints $\{q, a_i, y_i\}$ such that for any new question answer pair the label value can be predicted.

2) Sentence Model: The model for mapping sentences to vector representation is influenced by architectures of [1, 21], used for text classification tasks. However, we use it only for learning sentence embeddings which are then used for matching. The sentence is handled as a sequence of words. Hence each sentence is expressed as a matrix S where each column is the word vector w_i for that word, looked up from pre-trained word vectors such as Glove [22]. Next, a convolution layer extracts patterns from this matrix using numerous filters. Wide convolution is used to extract various features and create numerous feature maps and padding is used to give equal attention to all the words in the sentence. The pros and cons of wide and narrow convolution can be seen in [1]. Each filter will generate a unique feature map c_i using Eq. 4.

$$c_i = (s * F) = \sum_{k=i}^{i+m-1} s_k f_k \tag{4}$$

where * is the convolution operation between the two vectors $\boldsymbol{s}$ and F. s is the part of sentence matrix under the filter and F is the filter of width m. This set of feature maps is sent to nonlinearity and max-pooling layers to create the sentence's final embedding. These layers act as a non-linear feature extractor.

3) Word Overlap Features: There are two issues with this model. First is that we assign random vectors to the words not found in Glove and this can result in suboptimal performance when the number of QA pairs is small, and vectors can't be learned from the model. Second is that the distributed representations donot work well with proper nouns and numbers which are an essential part of factoid question answering. Hence to handle these issues, we follow [23] and introduce additional features x_{q_feat} and x_{a_feat} to establish relatedness between question and answer pairs. We calculate these extra features as word overlap measures for input question and answer sentences, including and excluding stop-words. We combine them with x_q and x_a before sending them to CNN. This calculation is straightforward and does not require any external resources.

4) Matching Layer: The outputs from the CNN layer denoted by x_q and x_a are used to compute question-answer similarity score. We use Eq.5 described in [5],

$$sim(x_q, x_a) = x_q^T M x_a \tag{5}$$

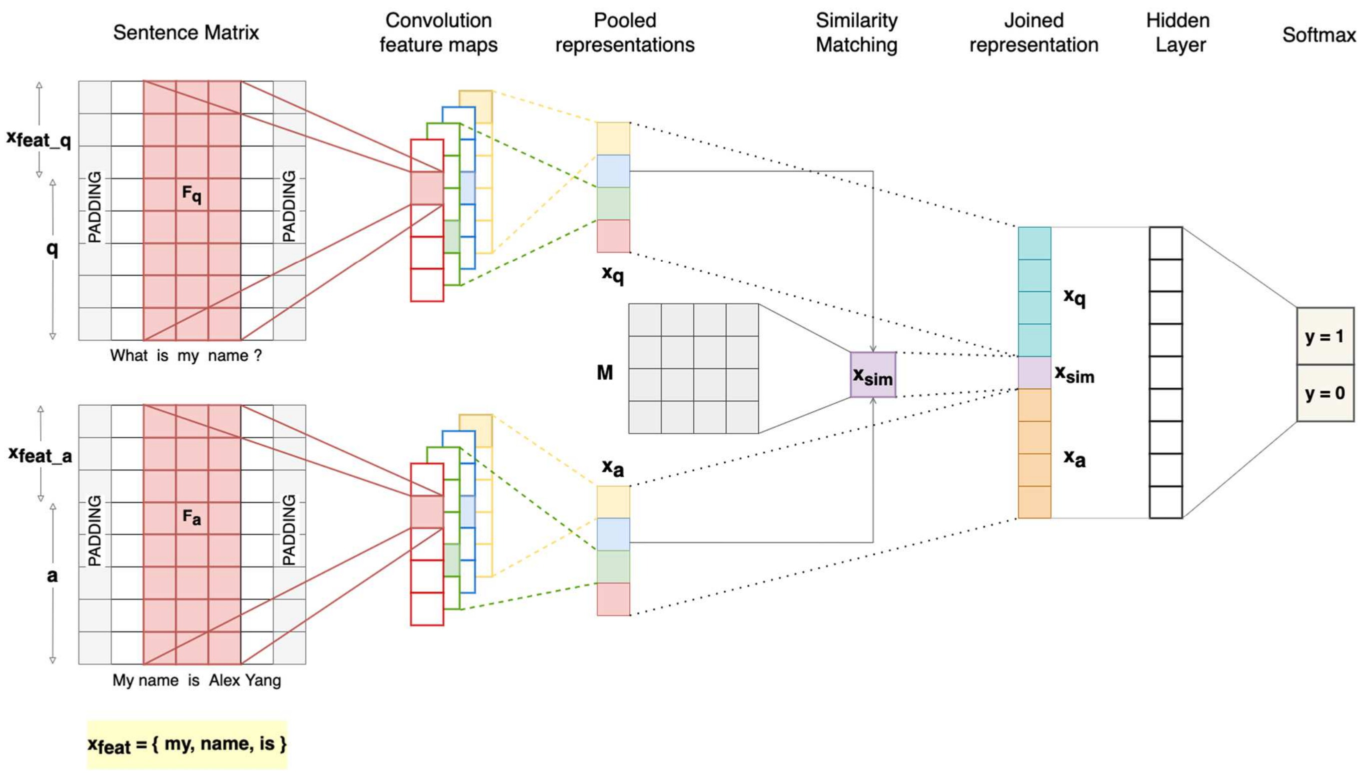

Fig. 1. Our Deep Learning architecture for reranking the sentences in the answer candidate pool for a given question.

where M is a similarity matrix. Eq. 5 is an example of a widely used noisy channel approach. We try to find a candidate answer $x_a' = M.x_a$ that is nearest to the given query x_q. M is a model parameter and is optimized while learning.

5) Classifier: Finally, all the outputs from previous layers, x_q, x_a *and* x_{sim} are sent as an input to a hidden layer that does the following transformation,

$$\alpha(W_h * x + b) \tag{6}$$

where W_h is the hidden layer's weight vector and α is the activation function. This output is flattened into a vector x which is sent as input to the softmax layer. It calculates the softmax probability for a class j using Eq. 7.

$$p(y = j|x) = \frac{e^{x^T\theta_j}}{\sum_{k=1}^{K} e^{x^T\theta_k}} \tag{7}$$

where θ is set of model parameters. Note that we have two classes here, label 0 and label 1. Finally, the candidate answers are sorted based on the probability value $p(y = \text{'Label 1'}|x)$, and the top ranker is the most relevant answer.

6) Regularization: We also experiment with the dropout regularization technique to overcome the issue of overfitting and generalizing the predictions. We randomly mask nodes of the network and make them ineffective while training.

IV. Experiments and Evaluation

A. Training

Following hyperparameters are chosen for training:

- filter width: 5
- number of filters: 100
- activation function: tanh
- hidden layer size: sum of sizes of x_q, x_a, x_{sim}
- dropout rate: 0.5

Model parameters are W_q, b_q, W_a, b_a, M, W_h, b_h, $W_{softmax}$, $b_{softmax}$, namely weights and biases and similarity matrix M. The cost for training is computed using negative log likelihood (NLL) given by Eq. 8. The model trains to minimize this cost.

$$L(y) = -\log(P(y = j|x)) \tag{8}$$

The parameters are optimized using stochastic gradient descent (SGD) with backpropagation. To fasten up the convergence rate of SGD, the technique of Adadelta [24] is used. The model is optimized in batches for many epochs, but training is stopped if no further optimization is possible.

TREC QA dataset is chosen for training which is structured as shown in Table I. It includes a set of factoid questions and a set of incorrect and correct answers for each. Wang et al. [25] created this dataset from Text REtrieval Conference (TREC) QA track (8-13) data, where the candidate answers were automatically selected from answer candidate pool for each question.

B. Performance Metrics

Two metrics, Mean Average Precision (MAP) as in Eq. 10 and Mean Reciprocal Rank (MRR) as in Eq. 9, are standard in IR and QA. Mean Average Precision computes the ranks of all the correct answers, and Mean Reciprocal Rank computes the rank of the first correct answer.

$$MRR = \frac{1}{|Q|}\sum_{q=1}^{|Q|}\frac{1}{rank(q)} \tag{9}$$

$$MAP = \frac{1}{|Q|}\sum_{q=1}^{Q} AveragePrecision(q) \tag{10}$$

where *rank(q)* is the position of the first correct answer in the candidate pool.

C. Results

The results of the training are reported under two settings: TRAIN-ALL, TRAIN. Under each setting, we analyse our model with and without regularization. In addition to this, we compare our work with similar works by Yu et al. [23] and Severyn & Moschitti [26]. Both these works have established state-of-the-art metrics under similar settings.

TABLE I. STRUCTURE OF THE TREC QA DATASET

Dataset	# Questions	# QA Pairs	% Correct
TRAIN - ALL	1229	53417	12%
TRAIN	94	4718	7.4%
DEV	82	1148	19.3%
TEST	100	1517	18.7%

Table II summarizes the performance when no extra overlap features were integrated and the model was trained using only the input question and answer pairs. As can be seen, our model outperforms the system in [23] and is comparable to [26]. Model in [23] also uses CNN to learn distributed representations, but it only operates on unigram and bigram, while our model uses higher filter width to capture farther dependencies. Along with this, our model combines the distributed representations of question and answer with the similarity score for final classification.

Table III compares the performance when word overlap features are included as x_{q_feat} and x_{a_feat}. Our model is slightly better than works of [23,26] in terms of both MAP and MRR for TRAIN. But for TRAIN-ALL there is slight decline in the values. But our model performs better compared to when no overlap features were incorporated.

TABLE II. PERFORMANCE ON TRAIN AND TRAIN-ALL (NO OVERLAP FEATURES)

Dataset	Model	MAP	MRR
TRAIN - ALL	Yu et al. [23] (unigram)	0.5470	0.6329
	Yu et al. [23](bigram)	0.5693	0.6613
	Severyn & Moschitti [26]	0.6709	0.7280
	Our Model	**0.6201**	**0.6676**
TRAIN	Yu et al. [23] (unigram)	0.5387	0.6284
	Yu et al. [23](bigram)	0.5476	0.6437
	Severyn & Moschitti [26]	0.6258	0.6591
	Our Model	**0.6345**	**0.6570**

TABLE III. PERFORMANCE ON TRAIN AND TRAIN-ALL (WITH OVERLAP FEATURES)

Dataset	Model	MAP	MRR
TRAIN - ALL	Yu et al. [23] (unigram)	0.6934	0.7677
	Yu et al. [23](bigram)	0.7113	0.7846
	Severyn & Moschitti [26]	0.7459	0.8078
	Our Model	**0.7742**	**0.8451**
TRAIN	Yu et al. [23] (unigram)	0.6889	0.7727
	Yu et al. [23](bigram)	0.7058	0.7800
	Severyn & Moschitti [26]	0.7329	0.7962
	Our Model	**0.7473**	**0.8211**

Table IV explores how regularization has affected the model. We masked the vectors x_q and x_a with a randomly generated mask vector before sending them to the matching layer. There is a slight reduction in metrics for both TRAIN and TRAIN-ALL. This may be attributed to the fact that dropout should be applied when the network is very deep with respect to the dataset. Our model is not very deep relative to TREC (having only one hidden layer). Hence using dropout only lowers the capacity of the network by switching off its nodes. Moreover, there is a chance that the number of training epochs is small and hence SGD is unable to reach convergence.

Wang et al. [25] reported that TRAIN-ALL being noisy had reduced the performance of their model. This is not true here, instead performance improved on TRAIN-ALL. Fig. 2 explores where our model stands compared to other works in the field.

TABLE IV. THE EFFECT OF USING DROPOUT AS REGULARIZATION

Dataset	Method	MAP	MRR
TRAIN - ALL	No Dropout	0.7742	0.8451
	Dropout	**0.7609**	**0.8253**
TRAIN	No Drpoout	0.7473	0.8211
	Dropout	**0.7101**	**0.7708**

V. CONCLUSION

The goal of this chatbot is to help students with their queries regarding their course material. They can upload their textbooks and ask questions based on them. Common issues that occur in user-provided data are lexical gaps, context awareness, ambiguity. Our chatbot has handled these well using good NLP transformations and coreference resolutions before creating the knowledge base.

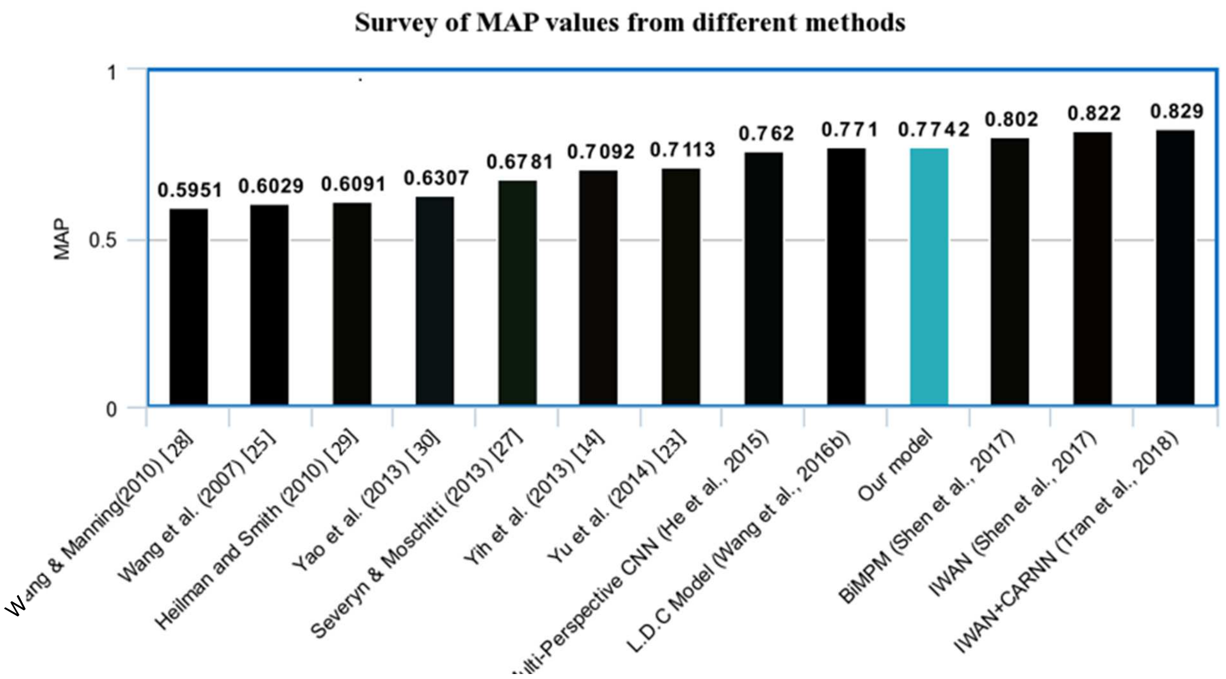

Fig. 2. Performance of our model compared to all other works in the field. Values are taken under TRAIN-ALL setting.

The quality and precision of question answering were improved using BM25 to create the candidate pool and CNN based ranking of those candidates. Underfitting was tackled using a deep neural network, while for overfitting, dropout was used. Using convolutional neural networks resulted in richer representations for questions and answers and emphasized the importance of learning high-quality sentence models. Hence, our model showed significant improvement on some previous state-of-the-art works done.

VI. Future Scope

We can implement the following to extend our work.

- *Augmenting with a web scraper* to collect text from any website and ask questions on that.
- *Adding a "contact teacher" option* for the cases when chatbot fails to give satisfactory answer.
- *Introducing a feedback system* so that the developers can improve upon the required area of the chatbot.
- *Allowing multiple file uploads* to expand the context for the questions and to increase the chances of finding the correct answer.
- *Adding a "contact teacher" option* for the cases when chat.

References

[1] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modeling sentences. In Proceedings of ACL, 2014.

[2] Y. Kim. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751, Doha, Qatar, October 2014.

[3] L. Yu, K. M. Hermann, P. Blunsom, and S. Pulman. Deep learning for answer sentence selection. *CoRR*, 2014.

[4] Roos, S. (2018). Chatbots in education: A passing trend or a valuable pedagogical tool?.

[5] Chopra, S., Gianforte, R., Sholar, J. (2016). Meet Percy: CS 221 Teaching Assistant Chatbot. ACM Transactions on Graphics, 1(1).

[6] Jia, Jiyou. (2009). CSIEC: A computer-assisted English learning chatbot based on textual knowledge and reasoning. Knowledge-Based Systems.

[7] https://www.admithub.com/case-study/how-georgia-state-university-supports-every-student-with-personalized-text-messaging/

[8] Gregory Grefenstette.Explorations in Automatic Thesaurus Discovery. Kluwer Academic Publishers, Norwell, MA, USA, 1994.

[9] Diana McCarthy, Rob Koeling, Julie Weeds, and John A. Carroll. Finding predominant word senses in untagged text. In Proceedings of ACL, 2004.

[10] Stephen Clark, Bob Coecke, and Mehrnoosh Sadrzadeh. A compositional distributional model of meaning.Proceedings of the Second Symposium on Quantum Interaction, 2008.

[11] Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. Semantic compositionality through recursive matrix-vector spaces. In Proceedings of EMNLP-CoNLL, 2012.

[12] Karl Moritz Hermann and Phil Blunsom. The role of syntax in vector space models of compositional semantics. In Proceedings of ACL, 2013.

[13] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. In Proceedings of ICML, 2008.

[14] Wen-tau Yih, Xiaodong He, and Christopher Meek. Semantic parsing for single-relation question answering. In Proceedings of ACL, 2014

[15] Antoine Bordes, Jason Weston, and Nicolas Usunier. Open question answering with weakly supervised embedding models. In Proceedings of ECML, 2014.

[16] Antoine Bordes, Sumit Chopra, and Jason Weston. Question answering with subgraph embeddings. In Proceedings of EMNLP, 2014.

[17] Caiming Xiong, Stephen Merity, Richard Socher. Dynamic Memory Networks for Visual and Textual Question Answering, MetaMind, Palo Alto, CA USA, 2016.

[18] Ming Tan, Cicero dos Santos, Bing Xiang & Bowen Zhou. LSTM based deep learning models for non-factoid answer selection, IBM Watson Core Technologies. 2016

[19] Asma Ben Abacha, Dina Demner - Fushman, A Question Entailment approach to Question Answering, Lister Hill Center, U.S. National Library of Medicine, U.S. National Institutes of Health, Bethesda, MD, 2019.

[20] Recasens, M., de Marneffe, M. C., Potts, C. (2013, June). The life and death of discourse entities: Identifying singleton mentions. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (pp. 627-633).

[21] Y. Kim. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751, Doha, Qatar, October 2014.

[22] Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). doi:10.3115/v1/d14-1162

[23] L. Yu, K. M. Hermann, P. Blunsom, and S. Pulman. Deep learning for answer sentence selection. *CoRR*, 2014.

[24] M. D. Zeiler. Adadelta: An adaptive learning rate method. *CoRR*, 2012.

[25] Mengqiu Wang, Noah A Smith, and Teruko Mitamura. What is the jeopardy model? a quasi synchronous grammar for Q.A. In EMNLP-CoNLL, 2007.

[26] Aliaksei Severyn, Alessandro Moschitti. Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks, 2015

[27] A. Severyn and A. Moschitti. Automatic feature engineering for answer selection and extraction. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 458–467, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[28] M. Wang and C. D. Manning. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In ACL, 2010.

[29] M. Heilman and N. A. Smith. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In NAACL, 2010.

[30] P. C. Xuchen Yao, Benjamin Van Durme and C. Callison-Burch. Answer extraction as sequence tagging with tree edit distance. In NAACL, 2013