AN ABSTRACT OF THE THESIS OF

Purbasha Chatterjee for the degree of Master of Science in Computer Science presented on December 11, 2018

Title: Answer Selection with Attentive Clustering.

Abstract approved: ______________________________

Prasad Tadepalli

Question answering forums like Reddit have been quite effective in improving social interaction and disseminating useful information. Community members ask a variety of questions related to a subject which are answered by other community members. The answers are given ratings by other members. In this thesis we study the problem of learning to recognize good answers using the community ratings as supervision. We design an attentive clustering neural network architecture to discriminate good answers from bad answers to a question. Taking advantage of the problem setting where there are usually many answers to the question that need to be scored, we also develop a collective classification model which clusters similar answers together, and biases the learner so that the answers in the same cluster have similar scores. The proposed solution uses a wide convolutional neural network to learn the text representation and computes a normalized score based on the relationship between the question and the answer and the similarity of the answer to other answers in the same cluster. Empirical results demonstrate that our collective classification model outperforms the baseline models and achieves the state-of-the-art performance in multiple benchmark domains.

Answer Selection with Attentive Clustering

by

Purbasha Chatterjee

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirement for the
degree of

Master of Science

Presented December 11, 2018

Commencement June 2019

Master of Science thesis of Purbasha Chatterjee presented on December 11, 2018

APPROVED:

Major Professor, representing Computer Science

Head of the School of Electrical Engineering & Computer Science

Dean of the Graduate School

Purbasha Chatterjee Author

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor Prof. Prasad Tadepalli for his continuous support during all the stages of this thesis. I would like to thank him for all the guidance and emotional support offered during these last two years. I would like to thank John Walker Orr for his constant support and advice.

I would like to express my appreciation and thanks to my friends and family members for their invaluable support in all my struggles and extending their support to continue my journey at OSU.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Question answering encompasses a broad research area in the field of natural language processing. Some of the proposed frameworks for question answering range from information-retrieval techniques to find the most relevant knowledge material or source, to methods for querying structured knowledge-bases or databases to generate an appropriate answer. One of the most popular challenges in this field is to select the most appropriate answers from the list of answers for a given question.

With the advent of advanced search engines, community question-answer forums have received wide popularity, allowing people from different backgrounds to ask and answer questions from different fields. Most of the forums are open to all, and due to the shortage of restrictions on access these forums are filled with either relevant or irrelevant answers- increasing the amount of time spent by the reader to find the best answers for the given question.

Apart from the above problem, chatbots are used widely as human assistants to ease the customer experience. These chatbots are expected to be well-trained to answer the customer's question with the most appropriate answer. A model with the ability to classify the answers for a given question as relevant and irrelevant can help the forums to list the answers from the most relevant answer to the most irrelevant one. Additionally, this model can be used in chatbots to rank the answer's relevancy for the question asked.

At the current time, the answer selection application is used widely in virtual assistants like Google Home and Amazon Alexa which query a large number of knowledge-bases and answer the human-posed question by selecting the most appropriate answers returned.

Answer Selection is one of the most challenging tasks because it requires the examination of a large set of answers before picking the best answer. Consider the following question and two possible answers in Table 1. Typically, a question and several answers are available and the task for the system is to classify them into good or bad. We view this as a supervised learning problem where the training examples consists of a set of questions and several answers for each question labeled as 'good' or 'bad'. The task is to learn to correctly distinguish good answers from bad answers on the test set.

Question	How does the Affordable Healthcare Act help me?
Good Answer	It helps you when you get cancer, and don't have to pay $100,000 for chemotherapy. Going into the relatively small amount of debt for that deductible is a much better place to be, financially. Further, before the ACA, insurance companies could deny coverage for preexisting conditions. So if you had no medical insurance, and then found out that you had cancer, you would not be able to get insurance to cover that cancer.
Bad Answer	Thanks. That actually makes sense. Is that why insurance prices have risen?

Table1: An example of Question- Good Answer and Bad Answer pair

These and similar techniques that weigh each word and analyze the appropriate similarity between both the pairs select the first one is the good answer. This has an obvious flaw in that the answers that do not add any new information might be scored highly as in Ans-1 above, while answers like Ans-2, which do not share any words with the question might be penalized.

Good Answer-1	It helps you when you get cancer, and don't have to pay $100,000 for chemotherapy. Going into the relatively small amount of debt for that deductible is a much better place to be, financially. Further, before the ACA, insurance companies could deny coverage for preexisting conditions. So if you had no medical insurance, and then found out that you had cancer, you would not be able to get insurance to cover that cancer.
Good Answer-2	And 28 years old had access to healthcare insurance prior to the ACA... more affordably and with more options, especially for "hit by bus" catastrophic insurance. After the ACA they are paying more for their healthcare insurance because the system is set up to overcharge young adults to offset the cost for the elderly. The protection from pre-existing conditions with higher insurance costs. Whether it is a positive or a negative for him is his determination. As a young adult who has always had insurance (even paying for personal plans when not provided by my employer) I did not benefit from the ACA, it hurt me... the same story is replicated across much of the young adult community in America.

Table2: An example of two Good Answers

In traditional approaches in information retrieval, the answers are evaluated using TF-IDF-like (term-frequency X inverse document frequency) scores which highly score answers which share many common words with the questions. In the above example we can observe that two good answers are similar to each other in the set of words used although they express different sentiments about the topic at hand. Most of the previous research work has focused mainly over the similarity of the question and answer. This ignores the fact that a good answer should provide some new information not in the original question. In the extreme case, relying simply on word similarities gives the highest scores for answers that merely copy the question word by word, which should not be the case.

Consider the example below:

Question:	**Is there any hiking trail in Corvallis?**
Answer 1:	Even I was wondering where is the best hiking place in Corvallis
Answer 2:	Yes, you can visit Bald Hill
Answer 3:	You can also visit Chip Ross Park which is known for scenic views

Table3: An example of Answers where the bad answer has the most common words with the question.

In the above scenario, previous model tends to rank Answer 1 higher than Answer 2 because Answer 1 has the most common words with the question even though Answer 2 answers it best. Answer 2 and Answer 3 which answer the question appropriately do not have common words with the question but do have common words among them.

Ranking and classification tends to be one of the most important fields of machine learning. These ranking problems are generally sub-categorized to pointwise ranking, pairwise ranking and listwise ranking. Point-wise approach (Severyn et al, 2015) is the simplest and is an effective method to build a ranking model. While training, point-wise ranking considers the instances of question, answer and the score, say in the form of (q_i, a_i, y_i). This data is used to train a binary classifier: $f(w, g(q_i, a_{ij})) \rightarrow y_{ij}$, where $g(.)$ maps question-answer pair to a feature vector and w is a vector of model weights. The decision function $f(\cdot)$ computes a dot product between the weight w and a text representation of a pair produced by $g(\cdot)$. While testing, the learned model is used to classify the unknown pairs (q_i, a_{ij}), producing a raw score which is then used to rank the answer for the given question against the set of all answers.

The second approach is the pairwise-ranking (Santos dos et al, 2016) method which is more advanced than the pointwise method. In this method, the model is trained such that it can rank the good answers higher with a good margin than a bad answer in the following manner:

$$f(w, g(q_i, a_{ij})) \geq f(w, g(q_i, a_{ik})) + \in,$$

where a_{ij} is a good answer and a_{ik} is a bad answer. On the other hand, pointwise approach is given as:

$$f(w, g(q_i, a_{ij})) \rightarrow y_{ij}$$

This calculates a raw score y_{ij} which is then used to rank the answers for a given question. This approach does not utilize a margin by which a good answer should be ranked higher than a bad one. So, if a bad answer score is high then it would be ranked higher than a good answer. Hence, pairwise approach works better practically because predicting relative order is closer to ranking nature than predicting a relevance score.

The most advanced approach is the listwise ranking (Cao et al, 2007). This method considers a list of answers as a single instance in learning for a given question.

In the recent years, many deep learning models have been proposed for answer selection method (Yu et al, 2014; Feng at al, 2015; Severyn et al, 2015) which represents the text using convolutional neural network or recurrent neural network. These text representations are then used to give a matching score by utilizing a similarity function for the two texts (Yu et al, 2014; Hu et al, 2014; dos Santos et al, 2015; Wang et al D. N., 2015; Severyn et al, 2015; Tan et al, 2015). Furthermore, models using attention and interaction (Zhang et al, 2017) were introduced which could capture the detailed information using pair-wise ranking method.

Despite these innovations and performance improvements, all these methods focus on the similarity of question to the answer and ignore the similarities between different answers. Our hypothesis in this work is that similarities between the answer words is just as important for selecting good answers if not more so.

Our research proposes a novel approach by combining the instances of pairwise and listwise ranking approaches Our main contribution is a new “Attentive Clustering” model, which first uses hard clustering to group similar answers together and uses soft clustering to modulate the scores of question-answer pairs. Together, they take into account the answer similarities to collectively classify the candidate answers. The intuition behind clustering is that similar answers tend to have similar scores. Hence, if a given question has two or more relevant answers then they will have similar scores. On the other hand, irrelevant answers tend to vary a lot and thus will tend to have different scores. So, considering the similarity measure between the answers will increase the score between good answers and penalize the bad ones.

Additionally, we also introduce a new dataset collected from the subreddit ”Explain like i’m five” containing natural questions and answers from its community with average answer length of more than 400 compared to other community question answering data of average answer length less than 100. It contains a large number of relatively long answers for each question. We show that Attentive Clustering outperforms all state-of-the-art results on this dataset sometimes with significant margins.

Chapter 2

Related Work

Some of the previously proposed techniques for answer selection range from information retrieval methods to machine learning methods which involve tailoring features manually by using semantic, syntactic or lexical similarity. These methods tend to calculate similarities between the question-answer pair by using bag-of-words (BOW), longest common substring or by word overlap ratio.

Recently, many deep learning methods achieved significant success in the field of natural language processing including question answering. These methods use deep neural network for distributed representation and then calculate the similarity between the pair at literal level. Current state-of-the-art methods incorporate attention mechanisms in a neural network architecture. These models learn to pay attention to relevant sections of the answers.

2.1 Information Retrieval Approaches

Initially, answer selection model was based on finding the semantic similarity and developed features manually using techniques such as:

- cosine similarity: captures the similarity between two words vectors by calculating cosine angle between them
- jaccard similarity: captures the common words between two sentences) between two documents
- GESD (Geometric mean of Euclidean and Sigmoid Dot product): It combines L2-norm and inner product where L2-norm is the forward-line

semantic distance between a pair and the inner product computes the angle between two sentences vector

- BM25: ranks the documents based on the common terms of the query matching with each document
- tf-idf: rank the terms higher that are frequent in the document but lowers the rank of the terms that are common in most of the documents; e.g lowers the rank of the articles, prepositions.

These models generally utilized lexical databases such as WordNet which were language dependent and prohibited the model to learn the required relevancy.

Some of the recent attempts at answer selection scores the relevancy of an answer to the question by mapping the question and answer pair into an n-dimensional vector space, and then calculating the cosine similarity or GESD (Geometric mean of Euclidean and Sigmoid Dot product) similarity between them. This intends to score the answer high if it is relevant and low if the answer is irrelevant. But these methods fail to capture the examples like below:

Question:	Is there any hiking trail in Corvallis?
Answers:	Yes, you can visit Bald Hill

Table5: An example of Question- Good Answer pair with no words in common

In the above answer, although the answer has been answered appropriately, due to the lack of word-level similarity, there is a high chance to rank it as bad by the above model.

2.2 Deep Learning Approach

(Feng at al, 2015) proposed vectorizing questions and answers using convolutional neural network and then use them for calculating GESD similarity between them. Some other authors for example (Yu et al, 2014; Severyn et al, 2015) proposed the method of scoring QA pair using neural network representations. Similarly, (Tan et al, 2015) suggested vectorizing questions and answers using recurrent neural network). (Qiu, 2015) introduced an approach of learning similarity between question and answer using convolutional neural network which encodes the sentence in the semantic space followed with interaction modeled at tensor (geometric object describing relation between vectors) layer. In (Wang et al D. N., 2015) proposed a model of recurrent neural network which would convert the model to learning to rank problem after a joint feature vector is determined from a joint long short-term memory (LSTM) which links to question and answer.

(Severyn et al, 2015; Wang et al Z. M., 2016; Tay, 2017) have proposed an approach to score the question answer pair using pointwise ranking (described in chapter 1). It learns the lexical and semantic interaction among question and answer vectors and scores them. But these models fail to capture the in-depth context of the question and answer vector interaction. For example; the same two words could have completely different context in question and answer. The context could be properly derived by giving appropriate amount of weight to the surrounding words in the sentence. The deep neural network utilizes the embedded words for the text representation.

One of the most commonly used embedding technique is word2vec (Mikolov, 2013) which is a neural embedding model. The process of calculating the probability distribution to get the vector representation of a word given a surrounding word is known as Continuous Bag of Words(CBOW). Precisely, it tries to predict the target word by considering the context of the surrounding word. In CBOW we can even consider multiple words in the context to calculate the multinomial distribution of the target word. An alternative to CBOW with multiple words in context is known as Skip-Gram model where one predicts *n* number of words given a single target word as input.

Another most widely used approach inspired from word2vec is Global Vectors for word representation (Glove). Glove (Pennington, 2014) helps to locate the words with similar context by mapping the word vector representation in a *n*-dimensional word space. Most of the unsupervised algorithms are based on word frequency and co-occurrence counts of the words. It produces a word vector with a meaningful structure by preserving the similarities of the words in the vector distance. It also minimizes the loss of the representation in lower dimensions that explains most of the variance of high dimensional data.

2.3 Attention Mechanism

Some of the promising attention-based mechanisms have been shown for NLP tasks, such as caption generation (Xu et al, 2015), machine translation (Bahdanau et al, 2015; Sutskever et al, 2014) and factoid question answering (Hermann et al, 2015). Later (Tan et al, 2015) proposed attention mechanism incorporated into the model using bi-directional long short-term memory, focusing on particular parts of the answer depending on question embedding. (Santos dos et al, 2016) introduced an approach of attentive pooling over convolutional and recurrent neural architecture which was a two-way attention mechanism for discriminative model training. (Zhang et al, 2017) proposed a combined approach of attention and interaction approach which learned the interaction of each pair of candidates and applied attention mechanism to measure the importance of each candidate answer. (Su et al, 2017) introduced enhanced embedding with word overlap along with attentive pooling and (Bachrach et al, 2017) introduced an approach of attention mechanism using a combined global and local view. This made the attention mechanism dependent on global embedding of the answer which was attained susing a separate model.

Other than attention mechanism, feature engineering of heuristics coupled with deep learning architectures has proven to be an efficient approach for ranking question and answer (Chen et al, 2017; Mohtarami, 2016). Some of the feature engineering introduced semantic and lexical relationships. These relationships like expressive rules can support sentence matching to capture the essential aspects of the information which are not captured by the underlying learning approach (Mihaylov et al, 2016; Belinkov, 2015; Tay, 2017) .

Chapter 3

Answer Selection by Semantic Analysis

In this chapter we look at the formal definition of answer selection, community question answering and different types of similarity analysis. We describe our method to solve the answer selection problem and provide an extensive evaluation of the proposed method.

3.1 Problem Statement

Answer selection defines a problem where a given question has a set of candidate answers and each of this answer is either relevant or irrelevant to the question. We assume a set of questions from the subreddit which is a community question-answering (CQA) forum. When a user posts a question, many answers are posted by other users. These answers from the CQA is utilized in our model and ranked according to their relevance to the question considering their context similarity.

An answer could be similar to the question either in terms of semantics or syntax. When the two sentences share a common meaning irrespective of their structure, then the two sentences are said to be semantically similar. Some of the common semantic similarity approaches are cosine similarity, jaccard similarity, tf-idf, etc.

Syntactic similarity is the degree to which the set of words of both the sentences are similar. A common approach is to look at the dependency parses and then group the sentences according to the parse nodes. Another approach is to POS tag the sentence and then compare with the POS tag of the other sentence for similarity. Part-of-speech (POS) tag assigns a label to each word indicating its part of speech.

3.2 Approach

To deal with the text data, initially each word of the sentence is embedded into an n-dimensional space. Embedding the word to n-dimension space helps to learn a representation of the text where words with similar meaning are embedded close to each other (using Euclidean distance metric to measure the closeness). The embedded words are then mapped to a neural architecture followed by a pooling method to get a proper text representation.

3.2.1 Word Embedding

In natural language processing, word embedding is a process of embedding a set of words into a vector of real numbers. It is a feature learning technique where words of similar meaning are mapped close to each other. The logic behind word embedding is to capture the semantic or contextual information of the text. One of the simplest methods to achieve this is one-hot encoding. In this method a set of documents or sentences is collected and then the occurrence of each word is counted. The output matrix consists of document as the row and word as the column.

Another approach is tf-idf where each term in the document is weighted. If a term occurs frequently in one document that term is given a higher weight but when terms occur frequently in different documents, then the weight of that term is reduced considering that they do not contain any useful information. Basically, term frequency (tf) is the ratio of number of times term *t* appears in a document by the total number of terms in the document and inverse document frequency (idf) is calculated as logarithmic of ratio of total number of documents by the number of documents with term *t* in it.

The first layer in representing the QA text after passing through embedding layer converts each input word w into a real-valued word vector embedding $w \in R^d$ where d is the dimension of the embedding. The embedded words are represented as a column vector in an embedding matrix $W^i \in R^{d \times |v|}$, where v is the vocabulary size. For a given input pair of question-answer (q, a), where the question q contains N tokens/ words and the given answer contains M tokens, the output of the word embedding layer is given as:

$$q_e = [e_q^{w1}\ e_q^{w2}\ e_q^{w3}\ e_q^{w4} \ldots\ldots e_q^{wN}] \text{ and } a_e = [e_a^{w1}\ e_a^{w2}\ e_a^{w3}\ e_a^{w4} \ldots\ldots e_a^{wM}]$$

3.2.2 Convolutional Neural Network- Wide & Narrow:

Convolutional networks use the feature extractor as the initial layer which convolves the input. It performs convolution (*) between the input matrix and the filter. Considering $t \in R^{|j|}$ is the input where t^i denotes to single feature value of the i-th word in the sentence and j is the length of the input. This 1-D convolution performs dot product with the filter vector $f \in R^{|k|}$, outputting: $c_i = t^T\ t_{(i-k+1:i)}.f$, where c is convolutional output and f is the filter vector of size k. According to the narrow convolution, it restricts the width of the filter $\leq$ j. The narrow convolution output is the subset of the output of the wide convolution. The wide convolution ensures to yield valid values even when $j \leq k$ and handles the words at the boundaries with equal weight. In general, wide convolution can be computed by padding the sentence with j-1 zeros to control the variable input length. It also ensures that it always generates a non-empty and a valid output c.

3.2.3 Text Representation:

The given input is parsed through the word embedding to map each word of the sentence to a distributional vector. Here, we have used Glove embedding with 50-dimensional space. The embedded sentence is then fed to the wide convolutional neural network with 128 filters with sizes of [1,2,3,5]. The convoluted embedded matrix for question is computed as: $Q = W^1X^q + b^1$ where $Q \in R^{f \times N}$. Similarly, for answers the convoluted embedded matrix would be computed as: $A = W^1X^a + b^1$ where $A \in R^{f \times M}$; where W^1 and b^1 are parameters to be learned by the model and $X^q = [x_1\ x_2\ x_3 \ldots\ldots x_N]$ containing $x_k \in R^{di}$ where k-th word is centralized and surrounded by i sequence of word embeddings.

3.3 Attentive Clustering

The baseline model exhibits the property of attentive pooling. In general, we tend to use either max-pooling, which extracts the maximum value of the input area or average pooling which extracts the mean of the input area. Pooling does not affect the convolutional depth rather it helps to reduce the spatial dimension of the input. Instead of using a simple max pooling we extended the attentive pooling (Santos dos et al, 2016) method along with clustering. The attentive pooling computes the attention vector by utilizing the similarity score between the projected sequences of the input pair. This bilinear similarity measure is followed by a non-linear activation. For a given input *(q,a)* we compute the matrix $Q \in R^{f \times N}$ and $A \in R^{f \times M}$. Then we perform element-wise activation as follows: $Z=\sigma(Q^T P_1 A)$ where $P_1 \in R^{f \times f}$ is the parameter to be learned by the model and σ is the activation function. This is followed by a row-wise and column-wise max pooling. This generates the vector $[z^q]_j = \max_{1<n<N} [z_{j,n}]$ and $[z^a]_j = \max_{1<m<M} [z_{m,j}]$ where each element j represents the important score of the vector z^a for the context surrounding the jth word. This is followed by a softmax function which generates the attention vectors σ^a and σ^q as follows: $[\sigma_j^a] = \frac{e^{[z^a]_j}}{\sum_{(1<m<M)} e^{[z^a]_m}}$ and $[\sigma_j^q] = \frac{e^{[z^q]_j}}{\sum_{(1<n<N)} e^{[z^q]_n}}$. The final representation is calculated by considering the dot product between convolved output and the attention vector: $x^q=Q\sigma^q$ and $x^a=A\sigma^a$. Ultimately, cosine similarity between x^q and x^a is computed: $s_{qa} = \frac{x^q \cdot x^a}{\|x^q\|\|x^a\|}$

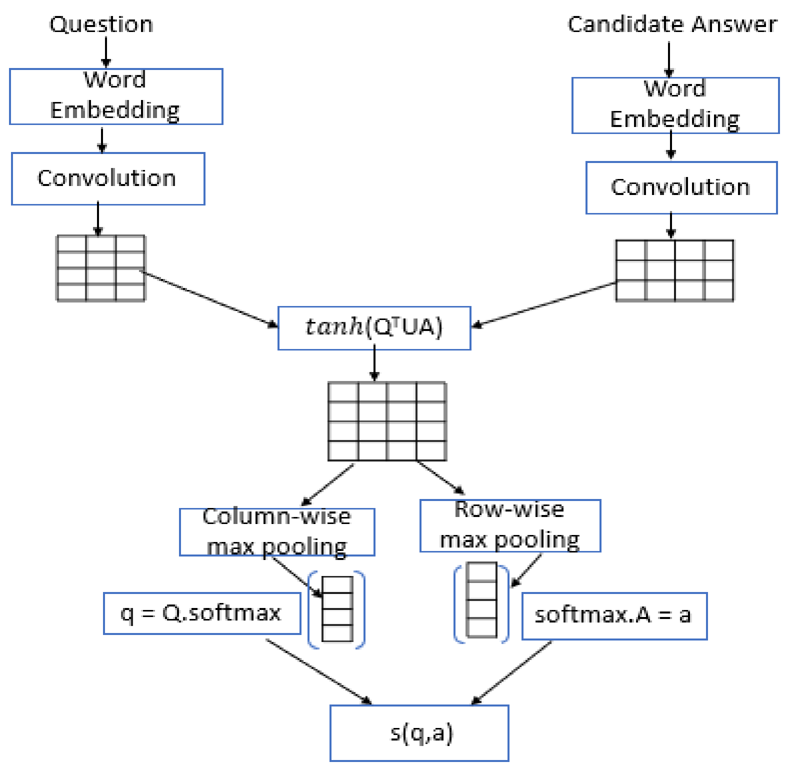

Fig1: Attentive Pooling

The above attentive pooling model has been extended by clustering the answers for a given question. For a given question, all its corresponding answers are clustered by using K-means clustering. Here, k is treated as a hyperparameter and computed as $k = \begin{cases} \frac{na}{10} + 1; if\ na \,\%\, 10 < 5 \\ \frac{na}{10} + 2;\ otherwise \end{cases}$; where na is the number of total answers each question contains. While training, for each good answer we pick the cluster the candidate answer belongs to. Then for each answer in the cluster, we perform the attentive pooling to get the representation pair of question and the answer. Finally, this pair is used to compute the cosine similarity between the pair: $s_{qc} = \frac{x^q \cdot x^c}{\|x^q\| \|x^c\|}$

Apart from capturing the similarity metric only between question and answer, we also capture the similarity between answers. This enables the model to rank answers with similar contexts similarly. Thus, we also apply the attentive pooling on the candidate answer and each answer in cluster as a pair. While applying attentive pooling, we use different parameters for element-wise activation Z=$\sigma(A^T P_2 C)$; where A is the candidate answer, C is the answer from the cluster and P_2 is the parameter to be learned. Following the same attentive pooling approach, we compute the cosine similarity between the pair given as: $s_{ac} = \frac{x^a \cdot x^c}{\|x^a\|\|x^c\|}$ .The three scores computed from the network are then used to calculate the final score of the candidate answer with respect to question.

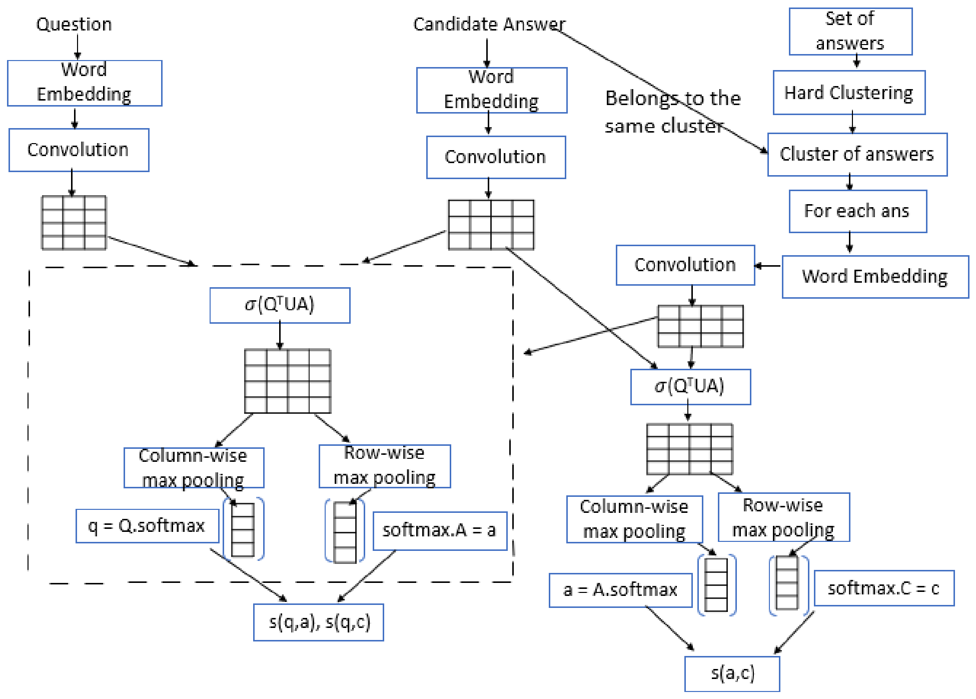

Fig2: Attentive Clustering

3.3.1 Scoring procedure and training:

The given input is passed through the network and scores s(q,a), s(q,c) and s(a,c) are obtained. All these scores are combined to calculate the final score between the pair of question and the candidate answer. The final score is computed as:

$$s = \left[\alpha\left(\frac{\sum(s(q,c)\ .\ s(a,c))}{\Sigma s(a,c)}\right)\right] + (1-\alpha)\ .\ s(q,a)$$

where α is the hyperparameter denoting the amount of weight given to the similarity measure between question-answer and answer-answer pair.

The network is trained on a set of training samples by minimizing a pairwise ranking loss function. For each sample, two pairs of input are considered that is (q,a$^+$) and (q,a$^-$) where a$^+$ denotes a relevant answer with good score and a$^-$ denotes an irrelevant answer for the question. The objective function of the network is the hinge loss which is defined as:

$$L = max\{0,m-(s^+ - s^-)\} + |\lambda|^2$$

where m is the margin and s^+ denotes the score between (q,a$^+$) and s^- denotes score between (q,a$^-$). While training for each sample we select a pair of relevant answer as the candidate answer and the question to obtain the score s^+. Simultaneously, we randomly sample an irrelevant answer and calculate its score s^- with respect to the question. These scores are used for calculating the above pairwise ranking loss. λ is the regularization term. Here we have used l2-norm.

It might appear that there is a relation between pairwise ranking loss and the SVM loss function. The SVM loss function for binary problem is given as: $\xi i = max\ (0, 1 - y_i f(x_i))$ where $\xi i \geq 0$. It is an approximation of hinge loss. The multiclass loss function is given as $\xi i = \sum_{j \neq yi} max(0, s_j - s_{yi} + \Delta)$. These are two approaches for maximizing the margin in the loss function (Tsochantaridis, 2005).

First, the loss function can be generalized by re-scaling the slack variables. When a margin constraint is violated with a high loss then it is penalized by multiplying the margin violation by the loss or by scaling the slack variable with inverse loss. This yields $\min_{w \in R^d} ||w||^2 + \frac{C}{N} \sum_i^N max(0,1 - y_i f(x_i))$ where $||w||^2$ is the regularization parameter. This approach scales the constant C and slack ξi that acts as a trade-off between error minimization and margin maximization.

The other approach is margin re-scaling where margin constraint is given as: $y_i f(x_i) \geq \Delta\ (y_i\ ,y) - \xi_i$; where Δ is the loss associated with a prediction y. This scaling formulation is not invariant under scaling of the loss function but also requires to scaling the feature map. Thus, this scaling approach is rarely used. Basically, if we consider m=1 and difference of the score $s^+ - s^-$, which is equivalent to $y_i f(x_i)$ then by approximation we get the loss function of SVM. In our approach the slack is not re-scaled as the margin is considered to be constant or a hyperparameter. Considering the margin re-scaling we observe $y_i f(x_i) \geq \Delta\ (y_i\ ,y) - 1 - y_i f(x_i)$ which on simplification gives us $\Delta\ (y_i\ ,y) \leq 1$.

While learning the text representations, weight parameters are learned for embedding each word. These learnt embeddings are convolved. with kernel feature map whose weights are again learnt in the training process. The similarity matrix is calculated between the pairs by using a weight parameter learnt by the model. The above-mentioned steps consider gradient with backpropagation to update the weight that minimizes the loss function. Since we have used a batch size of 1 along with a clustering method it increases the computation time of the model by $(b+c)$ times where b is the batch size and c is the cluster size as compared to attentive pooling.

Chapter 4

Experiments

4.1 Dataset

We have applied attentive clustering to set of 13,104 training samples. The raw data was collected from a sub-reddit “Explain like I m five”. We analyzed the data and found that most of the good scoring answers were posted within 24 hours of question posted. The answer thread started diminishing after 48 hours of the post and hence we filtered out the data points which were posted after 48 hours. The answers with less than 4 words along with no noun and less than 2 verbs were also filtered out. The data were further processed for the test data by removing the questions which had less than 4 answers.

Data	Number of unique questions	Number of samples	Avg. length of answers	Avg. length of questions
Train	1595	13104	467	94
Dev	398	3226	465	86
Test-1	713	9040	427	85
Test-2	735	9030	419	87

Table5: Dataset Statistics

4.2 Experimental Settings

The texts were tokenized, lemmatized and POS tagged using NLTK (Steven, 2009). We padded the questions and answers with their maximum length for wide convolution. We utilized Glove embedding of 50 dimensions for word embedding. The out of vocabulary words were initialized randomly. The window size of CNN was taken as [1,2,3,5] with 128 filters. We adopted stochastic gradient descent optimizer for optimizing the objective function. The learning rate is taken as 1.2 and batch size is 1. We have considered loss margin as 0.1 and score weight α as 0.4.

We have collected the test data from the same subreddit but the year when the questions were posted is different. We collected 18070 samples and split them into 2 test sets. None of the samples from train or dev data are present in any of the test samples. The number of answers for a given questions in test-1 set ranges from 4-156 and for test-2 it ranges from 4-172.

4.3 Results

System	Test-1		Test-2	
	MAP (%)	MRR (%)	MAP (%)	MRR (%)
Word Embedding	57.08	62.62	59.02	65.02
Learning to rank short text pairs (Severyn et al, 2015)	64.99	72.48	66.40	74.49
Applying deep learning to answer selection (Feng at al, 2015)	67.46	73.77	68.81	75.67
Attentive Pooling (Santos dos et al, 2016)	69.30	76.73	71.25	79.92
Attentive Clustering	**71.21**	**79.70**	**73.15**	**82.42**

Table6: Performance of different Systems

Learning to rank short text pairs (Severyn et al) computes the pointwise interaction between question and answer representations by wide convolution neural network for similarity measure. Applying deep learning to answer selection (Feng at al) creates QA-CNN where the similarity measure between a question and answer is being computed for the column max pooled vectors of the convolutional representations. Attentive Pooling (Santos dos et al) computes the similarity measure between the attentive pooled representations of the input pair.

Attentive Clustering	Train		Dev		Test-1		Test-2	
	69.39	76.62	66.04	71.69	71.20	79.71	73.15	82.41

Table7: Result of Attentive Clustering

We also captured the results for different α values to observe the importance of taking into account answer-answer similarity compared to the question-answer similarity.

α	Test-1		Test-2	
	MAP	MRR	MAP	MRR
0.0	70.82	79.13	71.82	80.07
0.2	70.83	79.17	72.40	81.74
0.4	71.20	79.71	**73.15**	**82.41**
0.5	**71.32**	**79.83**	72.80	81.53
0.7	71.04	78.92	72.59	81.29
1.0	71.01	79.05	72.31	80.72

Table8: Result of Attentive Clustering with different α

4.4 Analysis

In Table 6 we present the experimental results of the performance of different systems over the subreddit dataset. The results are observed in terms of mean average precision (MAP) and mean reciprocal rank (MRR) , the metrics normally used for ranking problem. MAP is the overall mean of the average precision where average precision is measured by computing the precision at every correctly returned result and then calculating the average of it.

$$MAP = \frac{\sum_{i=1}^{q} AP(i)}{q}$$

where $AP = \frac{\sum_{i=1}^{n} P(i)\times r(i)}{nr}$ and $r(i)$ is the indicator function equaling to 1 when the answer $P(i)$ is relevant and is the precision at i for all questions q.

The mean reciprocal rank (MRR) computes the quality of the ranking by considering the highest spot at which the first relevant answer has been placed. It is a statistical measure for evaluating the complete list of answers produced for the given question, ordered by the probability of the correctness (Craswell, 2009). MRR is the average of the reciprocal ranks produced for each question which is given as:

$$MRR = \frac{1}{|q|}\sum_{i=1}^{|q|}\frac{1}{rank\text{i}}$$

where ranki is the highest rank secured by as relevant answer for a question q.

According to Table 6 our attentive clustering system outperforms all the above baseline models, achieving state-of-the-art performance. We also captured the results for different α values in Table 8 to observe the importance of answer similarity. The result demonstrates that it is important to give weightage to the similarity measure between question-answer as well as answer-answer pair.

Chapter 5

Conclusions and Future Work

In this thesis, we developed a model which computes the score for the answers and rank them for a given question. We presented attentive clustering for answer selection which extends the attentive pooling for discriminative model training. It learns to compute the semantic similarity measure from the representations of the question-answer and answer-answer pairs. We demonstrated that attentive clustering with wide convolutional neural network helps the model to make better evaluations of the answers. The clustering of the answers and using their similarities to modulate the scores of question answer pairs modestly improves the performance of the model.

The model currently treats k for k-means as a hyperparameter. A further research can be carried out for learning the best value of k. Currently, the model has been applied over cleaned data. We could further extend the system to work on the raw and noisy data. Currently, the model is applied on a clean and processed data. Additionally, the model currently employs supervised learning. We could further explore how to apply answer selection with unsupervised data. The model could be further improved by annotating the question with interrogation category that is whether the question is Where, Why, Who, this could probably improve the semantic analysis by searching the relevant tokens in the answers.

Bibliography

Bachrach et al, Y. Z.-G. (2017). An Attention Mechanism for Neural Answer Selection Using a Combined Global and Local View. *arXiv:1707.01378v4*.

Bahdanau et al, D. C. (2015). Neural machine translation by jointly learning to align and translate. *ICLR.*

Belinkov, Y. M. (2015). A continuous word vector approach to answer selection in community question answering systems. *Proceedings of the 9th International Workshop on Semantic Evaluation*, (pp. 282–287).

Cao et al, Z. Q.-Y.-F. (2007). Learning to rank: From pairwise approach to listwise approach. *Proceedings of the 24th International Conference on Machine Learning, ICML* (pp. 129–136). NY, USA: ACM.

Chen et al, D. F. (2017). Reading wikipedia to answer open domain questions. . *In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics,* (pp. 1870–1879). Vancouver, Canada: ACL.

Craswell, N. (2009). Mean Reciprocal Rank. In *LIU L., ÖZSU M.T. (eds) Encyclopedia of Database Systems.* Boston, MA: Springer.

dos Santos et al, C. B. (2015). Learning hybrid representations to retrieve semantically equivalent questions. *ACL.*

Feng at al, M. X. (2015). Applying deep learning to answer selection: A study and an open task. *arXiv preprint:1508.01585*.

Hermann et al, K. M. (2015). Teaching machines to read and comprehend. *In Advances in Neural Information Processing Systems 28*, (pp. 1684–1692).

Hu et al, B. L. (2014). Convolutional neural network architectures for matching natural language sentences. *Advances in Neural Information Processing Systems (NIPS).*

Mihaylov et al, T. N. (2016). Ranking relevant answers in community question answering using semantic similarity based on fine-tuned word embeddings. *SemEval@ NAACL-HLT*, (pp. 879–886).

Mikolov, T. S. (2013). Distributed representations of words and phrases and their compositionality. . *Advances in Neural Information Processing Systems(NIPS).*

Mohtarami, M. B.-N. (2016). Neural-based Approaches for Ranking in Community Question Answering. *Proceedings of SemEval* (pp. 828-835). San Diego, California: Association for Computational Linguistics.

Pennington, J. S. (2014). Glove: Global vectors for word representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Qiu, X. H. (2015). Convolutional neural tensor network architecture for community-based question answering. *IJCAI*, (pp. 1305–1311).

Santos dos et al, C. T. (2016). Attentive pooling networks. *arXiv preprint arXiv:1602.03609.*

Severyn et al, A. M. (2015). Learning to rank short text pairs with convolutional deep neural networks. *Proceedings of SIGIR.* Santiago, Chile: ACM.

Steven, B. L. (2009). *Natural Language Processing with Python.* O'Reilly Media Inc.

Su et al, Z. a. (2017). Enhanced Embedding based Attentive Pooling Network for Answer Selection. *Proceedings of the 6th Conference on Natural Language Processing and Chinese Computing.*

Sutskever et al, I. V. (2014). Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems.*

Tan et al, M. d. (2015). Lstm-based deep learning models for nonfactoid answer selection. *CoRR, abs/1511.04108*.

Tay, Y. ,. (2017). Cross temporal recurrent networks for ranking question answer pairs. *CoRR abs/1711.07656.*

Tsochantaridis, I. J. (2005). Large Margin Methods for Structured and Interdependent Output Variables. *Journal of Machine Learning Research 6*, 1453–1484.

Verberne et al, S. B.-A. (2008). Using Syntactic Information for Improving Why-Question Answering. *Creative Commons Attribution-Noncommercial-Share Alike 3.0*.

Wang et al, D. N. (2015). A long short-term memory model for answer sentence selection in question answering. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing.*

Wang et al, M. M. (2010). Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. *The Proceedings of the 23rd International Conference on Computational Linguistics.*

Wang et al, Z. M. (2016). Sentence similarity learning by lexical decomposition and composition. *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers.*

Wikipedia. (2018, October). Retrieved from https://en.wikipedia.org/wiki/Mean_reciprocal_rank

Xu et al, K. B. (2015). Show, attend and tell: Neural image caption generation with visual attention. *In Proceedings of the 32nd International Conference on Machine Learning, ICML*, (pp. 2048–2057). Lille, France.

Yih et al, W.-t. C.-W. (2013). Question answering using enhanced lexical semantic models. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguist.*

Yu et al, L. H. (2014). Deep learning for answer sentence selection. *NIPS Deep Learning Workshop.*

Zhang et al, X. ,. (2017). Attentive Interactive Neural Networks for Answer Selection in Community Question Answering. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence.*