Deep Learning for Open-Domain Question Answering

Tapas Nayak **Shamil Chollampatt** **Steven Kester Yuwono**
Department of Computer Science
National University of Singapore
13 Computing Drive
Singapore 117417
`{nayakt, shamil, kester}@comp.nus.edu.sg`

Abstract

Open-domain question answering has been a popular research topic in NLP recently. Question answering task can be viewed as a two-step process. First, obtain a list of possible answers given a question. Second, choose the best answer sentence that is relevant to answer the given question, or to rank the answer sentences based on their degree of relevance. In this paper, we mainly focus on the second part of the question answering task, namely answer sentence selection. We have reviewed, re-implemented, and thoroughly evaluated some recent studies on answer sentence selection task with neural network approaches. We have successfully reproduced the results reported in the studies, including the state-of-the-art system.

1 Introduction

Question answering task can be divided into two categories. The first category focuses on semantic parsing. We can imagine a question to be processed and transformed into a database query, and obtain the answer from an existing knowledge database. The second category is open domain question answering, where the answer to a particular question can be obtained from anywhere, not limited to a particular knowledge base. This category is closely related to information retrieval.

In this paper, we focus mainly on the second category. Open domain question answering requires a few intermediate steps before a system can produce an answer. For example, to answer the question "*What are the three primary colors in the subtractive color model?*", the system needs to identify the type of the question, and retrieve relevant documents. The retrieved documents are then read and processed again to obtain a list of possible answer sentences or phrases. Finally, the best sentence or phrase is selected and presented as an answer. In this paper, we focus on answer sentence selection task, the task which selects the best answer sentence from a list of candidate sentences to answer a factual question.

Q: What are the three primary colors in the subtractive color model?

A:

✗ An 1877 color photo by Louis Ducos du Hauron, a French pioneer of color photography.

✓ The overlapping subtractive yellow, cyan and red (magenta) image elements can clearly be seen.

✗ The color that a surface displays depends on which parts of the visible spectrum are not absorbed and therefore remain visible.

Beyond its importance in question answering task, answer sentence selection is also a stand-alone task in the information extraction field. It can be viewed as a task to measure how similar is one sentence to another sentence.

The relevance of an answer to a particular question is usually measured in terms of semantic similarity between the two sentences. Past work in this field mainly utilizes syntactic matching of parse trees [1], or discriminative models over features produced from minimal edit sequences between dependency parse trees [2] [3]. Some prior work also make use of semantic features from external resources (e.g., WordNet). The previous state-of-the-art model for this task relies on a variety of such lexical semantic resources [4].

In this paper, we will describe, review, and evaluate four models namely, bag-of-words (baseline) [5], Bigram convolutional neural network (CNN) [5], Bigram CNN with additional features [6], and lexical composition and decomposition of sentences with CNN [7]. We have also attempted to modify the aforementioned models to improve the performance of the system. The WikiQA [6] dataset is used in the experiments.

The rest of this paper is organized as follows. Section 2 gives an overview of the related work. Section 3 describes the details of the models that have been re-implemented and studied in detail. We provide the details of the evaluation and the experimental results in Section 4 and 5 respectively. Finally, we conclude the paper in Section 6.

2 Related Work

Distributed representation of words: Continuous word representations where each word is represented by a word vector or word embedding has been proven useful for many NLP tasks. Word embeddings are usually obtained from training a neural network model [8]. However training such neural networks requires a lot of time which grows exponentially with larger training data or word embedding size. Mikolov et al. [9] proposed two new models to obtain word embeddings by training continuous bag-of-words (CBOW) and skip-gram (SKIP) models, presented as *word2vec* toolkit. A more recent study presented GloVe [10] to learn the distributed representation of words as an alternative to *word2vec*. Word embeddings can be used to measure the syntactic or semantic similarity between words. Some evaluation metrics for word embeddings include word analogies [9] and word similarity. Word analogies measures the vectors' quality by doing vector addition and subtraction (e.g., Athens is to Greece as Berlin is to ______?). Word similarity task is carried out by simply finding the other closest word by cosine distance given a word using the trained word embeddings. This field of work is very similar and relevant to our work. The difference is that we are required to measure the semantic and syntactic similarity between sentences instead of words.

Answer sentence selection: Answer selection task requires the system to select the most relevant answer to answer a given question. Answer sentence selection definitely requires both semantic and syntactic information in the sentences (i.e., the question, and the candidate answers) to measure their similarity and relevance. Current state-of the art models mostly focus on the syntactic matching between question and answers. Some examples would be a generative model to match the dependency trees of question answer pairs [11], a probabilistic model based on conditional random fields (CRF) [1], and a model which searches for the minimal edit sequences between parse trees using a tree kernel as a heuristic [2]. A more recent work utilizes rich lexical semantics to their state-of-the-art question answering matching model [4]. The model match the semantic relations of aligned words in question-answer pairs by combining lexical semantic from external resources such as WordNet with the distributed representations to capture semantic similarity. Then the features are fed into a conventional classifier.

3 Method

In this work, we replicate some of the prior work that uses neural network models and try their variants to solve this task. In all these models, the task is formulated as a binary classification task that takes a question-answer pair as input and the model estimates a probability for the given answer being a relevant answer for the given question. The words in the question and answer sentences are represented by pre-trained word vectors[1] embeddings released by [9].The various models that we implement and evaluate are described in detail below:

[1]Using *word2vec* toolkit

3.1 Bag-of-words Model (*BoW*)

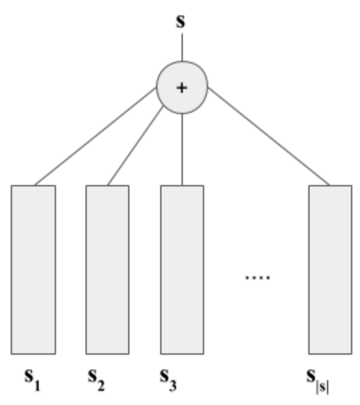

Figure 1: Sentence representation for the Bag-of-Words model

In the bag-of-words model, the question and answer sentence representations are computed by the summation of individual word vectors (Figure 3.1):

$$\mathbf{s} = \sum_{i=1}^{|\mathbf{s}|} \mathbf{s}_i \tag{1}$$

where $|\mathbf{s}|$ is the number of words in the sentence and $\mathbf{s}_i$ is the d-dimensional word vector of the i^{th} word in the sentence. The sentence representation **s** is also a d dimensional vector.

The similarity score is computed by a linear transformation to a single score with a sigmoid function which constrains this score to [0,1]. Given the sentence representations **s** and **t** of the question sentence and answer sentence, respectively, the similarity is computed by:

$$\text{sim}(\mathbf{s}, \mathbf{t}) = \sigma(\mathbf{s}^T W \mathbf{t} + b) \tag{2}$$

where W is the weight vector of dimension $d \times d$ and b is the scalar bias term of the linear transformation.

3.2 Simple CNN Models (*Bigram CNN* and *Trigram CNN*)

We use convolutional neural networks with bigram filters for representing sentences following [5] and [6](Figure 3.2). d bigram filters are used. The final sentence representation is calculated by average pooling across the sentence length resulting in a d dimensional vector. Formally, the representation of the sentence is calculated by the following equation:

$$s = \frac{1}{|s| - 1} \sum_{i=1}^{|s|-1} \tanh(T_L s_i + T_R s_{i+1} + b) \tag{3}$$

where s_i is the i-th word in the sentence and s is the sentence vector. T_L and T_R are trainable parameters, b is bias.

We also try a variant of the above model, *Trigram CNN* where convolutions are performed over three consecutive word vectors. The final similarity score is computed using linear transformation of the sentence representations as given in Equation 2.

3.3 Lexical Composition and Decomposition (*Decomp-Comp CNN*)

We implement the state-of-the-art system by [7] for the task of answer selection on the WikiQA dataset. In this model the sentence representations for the question and the answer sentences are

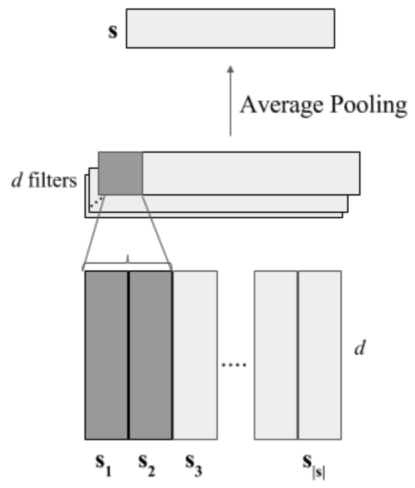

Figure 2: Sentence representation using *Bigram CNN* model

computed by decomposing into similar and dissimilar components and composing it into a single vector using a CNN. With this model, the lexical gap between the two sentences and the dissimilarities between the two sentences can be captured which gives better predictive power to the model.

Figure 3.3 gives an illustration of the model architecture. Specifically, given the question sentence **S**, and the answer sentence **T**, a similarity matrix **A** is computed, where each element $a_{i,j}$ is given by:

$$a_{i,j} = \frac{\mathbf{s}_i^T \mathbf{t}_j}{\|\mathbf{s}_i\| \cdot \|\mathbf{t}_j\|} \forall \mathbf{s}_i \in \mathbf{S}, \forall \mathbf{t}_j \in \mathbf{T}. \tag{4}$$

Using the similarity matrix **A**, we find matching vectors for all words in **S** and **T** using the following function:

$$\hat{\mathbf{s}_i} = \frac{\sum_{j=k-w}^{k+w} a_{i,j} \mathbf{t}_j}{\sum_{j=k-w}^{k+w} a_{i,j}} \tag{5}$$

where $k = \arg\max_j a_{i,j}$ and w is the window size[2]. Correspondingly, the matching vectors $\hat{\mathbf{t}_i}$s for the answer sentence **T** are also computed. These matching vectors are decomposed into similar and dissimilar components by *orthogonal* decomposition:

$$\mathbf{s}_i^+ = \frac{\mathbf{s}_i \cdot \hat{\mathbf{s}_i}}{\hat{\mathbf{s}_i} \cdot \hat{\mathbf{s}_i}} \hat{\mathbf{s}_i} \quad \text{and} \quad \mathbf{s}_i^- = \mathbf{s}_i - \mathbf{s}_i^+ \tag{6}$$

Let $\mathbf{S}^+$ be a concatenated matrix representation of the similar components $\mathbf{s}_i^+$ and $\mathbf{S}^-$ be the matrix for the dissimilar components. For the answer sentence, $\mathbf{T}^+$ and $\mathbf{T}^-$ are computed similarly.

For composing the similar and dissimilar matrices, a CNN with multiple filters of different sizes and initialization is used. For each filter, a *max-pooling* operation is performed to obtain a single scalar value. The final sentence representation is a vector composing of the scalar values obtained for each of the filters.

Similar to the previous models, a linear transformation is applied between the two sentence representations along with a sigmoid function.

3.4 Using Additional Features

Following [6], we add hand-crafted features along with the similarity score outputted by the sigmoid function in our models. These features along with the previously computed similarity score are used to train a logistic regression model which computes a new score.

[2]we use a $w = 3$ for our experiments

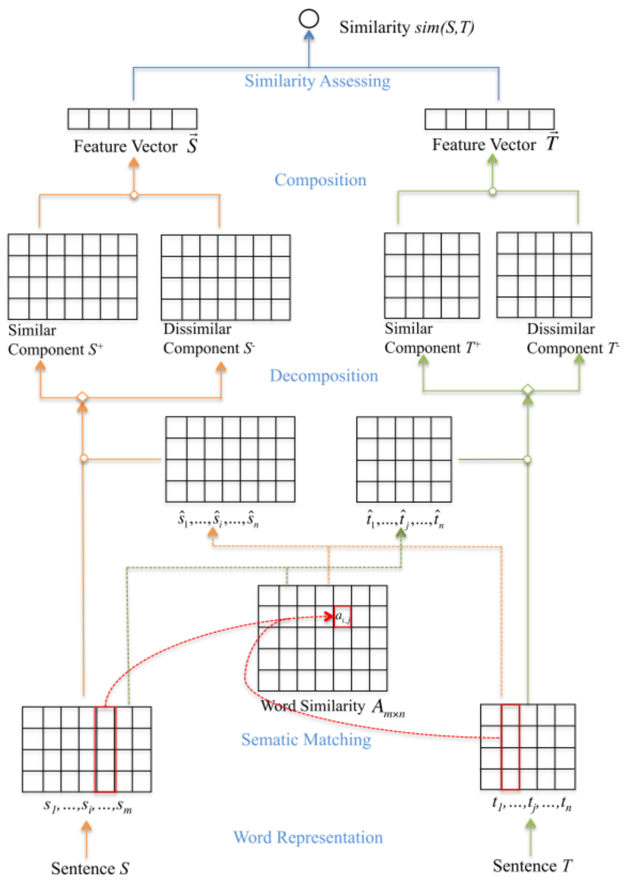

Figure 3: Decomp-Comp CNN Model Architecture [7]

Specifically we add the following features:

- word counts of question sentence and answer sentence.
- tf-idf weighted word counts of question and answer sentences.
- tf-idf weighted overlapping word counts.
- type of question.

We add the above features with the output score of *Bigram CNN*, *Trigram CNN* and *Decomp-Comp CNN* models separately.

4 Evaluation

Following prior work, we evaluate the systems using two metrics: Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR). Both measures are commonly used in ranking based systems like information retrieval system, where a system is required to extract and rank a list of documents for a given query.

4.1 Mean Average Precision (MAP)

Precision is defined as follows:

$$\text{precision} = \frac{|\text{relevant answers}| \cap |\text{retrieved answers}|}{|\text{retrieved answers}|} \tag{7}$$

where |relevant answers| is the number of answer sentences which are relevant for a particular question from given candidate answer sentences and |retrieved answers| is number of answer sentences selected by system for a particular question. Average precision for a query q is defined as follows:

$$\text{AveP}(q) = \frac{1}{|\text{relevant answers}|} \sum_{k=1}^{n} (\text{precision}(k) \times \text{rel}(k)) \tag{8}$$

where precision(k) is the precision at the k^{th} retrieved answer sentence and rel(k) is the boolean indicator which indicates if the k^{th} answer is relevant or not. Finally, the mean average precision or MAP is defined as follows:

$$\text{MAP} = \frac{1}{|Q|} \sum_{q \in Q} \text{AveP}(q) \tag{9}$$

where $|Q|$ is the number of questions used for evaluation.

4.2 Mean Reciprocal Rank (MRR)

Mean reciprocal rank or MRR is defined as follows:

$$\text{MRR} = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{\text{rank}_q} \tag{10}$$

where $|Q|$ is the number of questions used for evaluation and rank$_q$ is the rank of first relevant answer sentence in the retrieved answer sentences by system.

5 Experiments

5.1 Dataset

Microsoft released the WikiQA[3] dataset containing 3,047 real queries sampled from Bing search engine queries and the 29,258 answer sentences obtained from Wikipedia pages based on user clicks. The data can be considered as question-answer sentence pairs associated with a binary label `1` (indicating that the answer sentence is appropriate for the question) and `0` (otherwise). The dataset is split into 70% training, 20% testing and 10% for validation.

5.2 Implementation Details

We use publicly available Google's *word2vec*[4] word vectors with dimension 300 for all our experimental models. An <unk> token with random vector of dimension 300 added to the Word2Vec for unknown words. All the models are implemented using Keras [12] deep learning toolkit and we use Theano [13] as the backend.

In bag-of-words model, we have used batch size of 100 and ran the model for 50 epochs. We use the development data to pick the best model from the 50 epochs and that model is applied on test set. For Bigram CNN model and Trigram CNN, we use a batch size of 10 and total 20 epochs. Like the bag-of-words model, development data is used to pick the best model. In decomposition and composition based CNN model (*Decomp-Comp CNN*), we use unigram, bigram and trigram filters. For each n-gram, we have used 500 different filters with different initializations, resulting in a total of 1500 filters. After convolution and max-pooling, a 1500-dimensional vector is obtained for every sentence. We ran the model for 5 epochs with batch size of 10. Development data set is used to pick the best model. For Bigram CNN, Trigram CNN and Decomp-Comp CNN models, the answer sentences are truncated up to 40 words.

Logistic regression is used when count based feature set is applied along with question - answer similarity score. We have used the Keras dense layer with a bias to implement the logistic regression. We ran the model for 20 epochs with batch size of 1 and like earlier development data set is used to pick the best model.

Our code is publicly available on GitHub[5].

[3]https://www.microsoft.com/en-us/research/publication/wikiqa-a-challenge-dataset-for-open-domain-question-answering/

[4]https://code.google.com/archive/p/word2vec/

[5]https://github.com/nayakt/Answer-Sentence-Selection

5.3 Results

We evaluate the systems on the WikiQA test set. Following prior work, we evaluate using the MRR and MAP metrics after eliminating the questions which has no relevant answers and questions where all the answers are relevant. The results are evaluated using the official evaluation script of TREC tasks (`trec_eval`). We include the results we obtained for the re-implemented systems along with the published results. We also report the results for the model variants that we experimented with.

Table 1: Performance comparison of various models

Replicated Systems	**MAP**	**MRR**
Decomp-Comp CNN	**0.7086**	**0.7281**
Bigram CNN + features	0.6364	0.6509
Bigram CNN	0.6196	0.6286
Bag of Words	0.4812	0.4870
Our Variants		
Decomp-Comp CNN$_{multiple}$ + features	0.6995	0.7208
Decomp-Comp CNN + features	0.6960	0.7149
Trigram CNN + features	0.6467	0.6647
Trigram CNN	0.6242	0.6402
Best Published Results		
Decomp-Comp CNN [7]	0.7058	0.7226
Relational CNN [14]	0.6951	0.7107
ABCNN [15]	0.6921	0.7108
AP-CNN [16]	0.6886	0.6957
Bigram CNN [6]	0.6190	0.6281
Bigram CNN + features [6]	0.6520	0.6652

5.4 Discussion

Bag of words model is unable to capture the relationships among words in a sentence and due to which model also fails to learn those relationships. This is the reason why bag of words model does not work well for this task. Bigram and Trigram models captures some amount of relationship among words in a sentence and thus improved the MAP and MRR significantly. Bigram and Trigram models just capture word relationships, but do not capture the keyword information between question and answer sentences. Identifying important words between query and documents is a critical task for any information retrieval system. Bigram and Trigram models do not incorporate those information. Term frequency and inverse document frequency based features help to identify those kind of information. Thus, incorporating count based features along with similarity score improved the score.

In decomposition and composition model, cosine based similarity measurement metric is used to determine similarity score among words of question and answer sentence. Then local sliding window of size w (we used w=1, w=2 and w=3) is used to capture word relationships in similar and dissimilar parts for question and answer sentence. Orthogonal decomposition helps to segregate the similar and dissimilar parts of question and answer sentence. Features which represents the similar parts help to identify the answer sentences which are related to question and features which represents the dissimilar parts help to identify the answer sentences which are not related. Due to this distinction, decomposition and composition based CNN models give the highest MAP and MRR score. One observation we see that if we add count based features along with Decomp-Comp CNN similarity/dissimilarity score, performance degrades. One reason may be that the there are total 7 features based on question type and term-frequency and inverse document frequency. Similarity/dissimilarity score derived from above Decomp-Comp CNN model is very rich regarding containing the word relationships and sentence semantics. Adding this important feature with other count based features may reduce the importance of this feature significantly in logistic regression based learning. That may be the reason why we see performance degradation in Decomp-Comp CNN count based model. To verify this conclusion, we have added the similarity/dissimilarity score 5 times (Decomp-Comp CNN$_{multiple}$ + features) in the feature list and we saw further improvement, though the score was still below the Decomp-Comp CNN model.

6 Conclusion

We compared various deep learning-based systems for the task of answer selection. These systems use neural networks for representing sentence pairs and calculating the relevance of an answer sentence to a given question. We successfully replicated the state-of-the-art[17] for this task. Our replication of the other models also confirms with the original results presented in the papers. Some of the variants of these models we experimented with further improves the performance over their respective baselines to achieve competitive performance.

References

[1] Mengqiu Wang and Christopher D Manning. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1164–1172. Association for Computational Linguistics, 2010.

[2] Michael Heilman and Noah A Smith. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1011–1019. Association for Computational Linguistics, 2010.

[3] Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. Answer extraction as sequence tagging with tree edit distance. In *HLT-NAACL*, pages 858–867. Citeseer, 2013.

[4] Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. Question answering using enhanced lexical semantic models. 2013.

[5] Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. Deep learning for answer sentence selection. *CoRR*, abs/1412.1632, 2014.

[6] Yi Yang, Wen-tau Yih, and Christopher Meek. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of EMNLP*, 2015.

[7] Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. Sentence similarity learning by lexical decomposition and composition. *CoRR*, abs/1602.07019, 2016.

[8] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003.

[9] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[10] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[11] Mengqiu Wang, Noah A Smith, and Teruko Mitamura. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP-CoNLL*, volume 7, pages 22–32, 2007.

[12] François Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[13] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[14] Aliaksei Severyn and Alessandro Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2015.

[15] Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272, 2016.

[16] Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. Attentive pooling networks. *CoRR*, abs/1602.03609, 2016.

[17] Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proceedings of EMNLP-CoNLL*. Association for Computational Linguistics, 2007.