Experiments with Convolutional Neural Network Models for Answer Selection

Jinfeng Rao, Hua He
Department of Computer Science
University of Maryland
jinfeng@cs.umd.edu,huah@umd.edu

Jimmy Lin
David R. Cheriton School of Computer Science
University of Waterloo
jimmylin@uwaterloo.ca

ABSTRACT

In recent years, neural networks have been applied to many text processing problems. One example is learning a similarity function between pairs of text, which has applications to paraphrase extraction, plagiarism detection, question answering, and *ad hoc* retrieval. Within the information retrieval community, the convolutional neural network model proposed by Severyn and Moschitti in a SIGIR 2015 paper has gained prominence. This paper focuses on the problem of answer selection for question answering: we attempt to *replicate* the results of Severyn and Moschitti using their open-source code as well as to *reproduce* their results via a *de novo* (i.e., from scratch) implementation using a completely different deep learning toolkit. Our *de novo* implementation is instructive in ascertaining whether reported results generalize across toolkits, each of which have their idiosyncrasies. We were able to successfully replicate and reproduce the reported results of Severyn and Moschitti, albeit with minor differences in effectiveness, but affirming the overall design of their model. Additional ablation experiments break down the components of the model to show their contributions to overall effectiveness. Interestingly, we find that removing one component actually increases effectiveness and that a simplified model with only four word overlap features performs surprisingly well, even better than convolution feature maps alone.

1 INTRODUCTION

The problem of learning a similarity function between pairs of texts has broad applicability to a variety of natural language processing tasks, including paraphrase extraction, plagiarism detection, question answering, and *ad hoc* retrieval. The natural language processing community has made substantial strides in tackling this problem with approaches based on neural networks. In contrast to previous work that relies heavily on hand-crafted features, models based on neural networks are not only more effective, but also obviate the need for feature engineering.

Recently, Severyn and Moschitti [14] (henceforth, SM for short) proposed a convolutional neural network model for learning similarities between pairs of short texts, which was applied to answer selection for question answering and to tweet reranking in an *ad hoc* retrieval task. Because this work has gained prominence in the information retrieval community (garnering, for example, a large number of citations in a short amount of time), in this paper we attempted to replicate and reproduce the authors' results.

Our efforts proceeded in two steps: First, SM released their source code on GitHub,[1] which uses the Theano deep learning toolkit,[2] and from this we attempted to *replicate* the results in their paper—by running their code directly. Second, we set aside their code and attempted to *reproduce* their model *de novo* (i.e., from scratch) using the Torch deep learning toolkit.[3] Both efforts are instructive for different reasons: *Replicating* the original results using the authors' code verifies that the code does indeed correspond to the models described in the paper, and that there are no "hidden dependencies" missing in the code. *Reproducing* the model from scratch, using a completely different toolkit, represents a higher standard of verifiability—that the authors have adequately described their model, and that their results are robust with respect to idiosyncrasies in the underlying deep learning toolkits.

Contributions. We view our work as having three contributions:

- We were able to successfully replicate and reproduce the results of Severyn and Moschitti, contributing to the community's growing interests in reproducibility and related issues [2, 10, 13].
- Our ablation studies reveal new insights about the inner workings of the SM model in terms of understanding the contribution of different components: convolution feature maps, a similarity modeling component, and "extra features" that derive from traditional retrieval measures. Interestingly, we find that removing the similarity modeling component actually increases effectiveness in answer selection and that a simplified model with only four word overlap features performs surprisingly well, even better than the complete SM model without the extra features.
- We make our source code publicly available[4] so that others can build on our work. Having publicly-available implementations of the same model in two different deep learning toolkits is instructive for developers trying to understand the intricacies of neural network models and the primitive "building blocks" offered by competing toolkits.

2 MODEL OVERVIEW

2.1 Network Architecture

As this paper primarily focuses on the reproducibility of results, we introduce the model architecture briefly and refer the reader to the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGIR'17, August 7–11, 2017, Shinjuku, Tokyo, Japan
© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM.
978-1-4503-5022-8/17/08…$15.00
DOI: http://dx.doi.org/10.1145/3077136.3080648

[1] https://github.com/aseveryn/deep-qa
[2] http://deeplearning.net/software/theano/
[3] http://torch.ch/
[4] https://github.com/castorini/SM-CNN-Torch

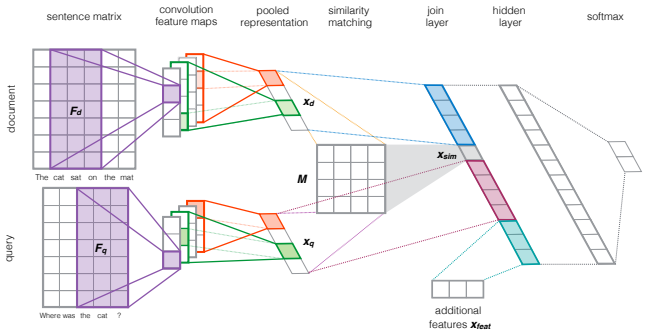

Figure 1: The SM convolutional neural network model, comprised of three main components: convolution feature maps with pooling, a similarity matrix, and extra features.

original paper for more details. The overall architecture of the SM model is shown in Figure 1, taken from the original paper for clarity. The model has a general "Siamese" structure [3] with two subnetworks processing the "query" and the "document" in parallel. This general architecture is fairly common and used in a variety of other models as well [6–8]. Implicit in this architecture is the assumption that both inputs are relatively short (i.e., sentences), since they are ultimately converted into fixed-length vector representations.

The input to each "arm" in the neural network is a sequence of words $[w_1, w_2, \ldots w_{|S|}]$, each of which is translated into its corresponding distributional vector (i.e., from a word embedding), yielding a sentence matrix. Convolution feature maps are applied to this sentence matrix, followed by ReLU activation and simple max-pooling, to arrive at a representation vector x_q for the query and x_d for the document.

At the join layer (see Figure 1), all intermediate representations are concatenated into a single vector:

$$\mathrm{x}_{\text{join}} = [\mathrm{x}_q^T; x_{\text{sim}}; \mathrm{x}_d^T; \mathrm{x}_{\text{feat}}^T] \tag{1}$$

where x_{sim} defines the *bilinear* similarity between x_q and x_d as follows:

$$\text{sim}(\mathrm{x}_q, \mathrm{x}_d) = \mathrm{x}_q^T M \mathrm{x}_d \tag{2}$$

This similarity matrix M is a parameter of the network that is optimized during training.

The bilinear similarity matrix can be viewed as an adaptation of the noisy-channel model from machine translation, which has previously been used for question answering [4]. Basically, it captures a transformation from the source embedding x_d to the target embedding $x_d^{'} = Mx_d$ where similarity to the input query x_q is maximized. The parameters are initialized randomly and optimized through back-propagation during training. Finally, x_{feat} represents additional features that are task specific—these are significant, as we discuss later.

The original SM paper examined two specific tasks: answer selection for question answering and tweet reranking in an *ad hoc* retrieval task. In this paper, we only focus on the first task for a number of reasons. Primarily, there are aspects of the SM model that make its setup somewhat unrealistic for tweet reranking (as described in their paper): specifically, the model uses as one of the "extra features" the normalized rank across *all* systems that participated in the TREC Microblog evaluations. This setup is closer to an oracle run fusion task than an actual tweet reranking task, since a system would not have access to scores from all other systems that participated in the evaluation. We have begun to examine the SM model on the tweet reranking task, but the page limitations of a short paper preclude us from diving into the full details.

Set	# Question	# Answers	% Correct
TRAIN	94	4,718	7.4%
TRAIN-ALL	1,229	53,417	12.0%
Dev	84	1,148	19.3%
Test	100	1,517	18.7%

Table 1: Statistics of the TrecQA dataset for answer selection, which contains two distinct training sets.

2.2 Answer Selection

Answer selection is an important component of an overall question answering system: given a question q and a candidate set of sentences $\{c_1, c_2, \ldots c_n\}$, the task is to identify sentences that contain the answer. In a standard pipeline architecture [15], answer selection is applied to the output of a module that performs passage retrieval based on some form of term matching. Selected sentences can then be directly presented to users or serve as input to subsequent stages that identify "exact" answers [16]. The experiments in this paper evaluate the answer selection task for question answering. Although nominally a classification task, system output is usually evaluated in terms of ranked retrieval metrics.

For answer selection, the SM model uses "extra features" x_{feat} comprised of four word overlap measures between each sentence pair: word overlap and IDF-weighted word overlap computed between all words and only non-stopwords. The use of these external features aims to mitigate two issues associated with word embeddings. First, about 15% of words in the vocabulary are not found in the word embeddings; those word vectors are randomly initialized during training, which could result in suboptimal similarity learning. Second, even for those words found in the word embeddings, distributional representations sometimes can not capture the relatedness of numbers and proper nouns. However, such information is important, especially for factoid questions where many answers are either numbers or proper nouns.

3 EXPERIMENTS

The SM paper evaluates their convolutional neural network on the popular TrecQA dataset, first introduced by Wang et al. [19] and further elaborated by Yao et al. [20]. Basic statistics are shown in Table 1. The dataset contains a number of factoid questions, each of which is associated with candidate sentences that either contain or do not contain the answer (i.e., positive and negative examples). The questions are taken from the Question Answering Tracks from TREC 8–13, and the candidate answers derive from the output of track participants. Note that there are two distinct training sets, known as TRAIN and TRAIN-ALL. The TRAIN-ALL set is much larger but also contains more noise. Following previous work, the task is evaluated in terms of Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR).

We first attempted to replicate the results of SM using the authors' open-source Theano code. Results are reported in Table 2: The

Condition	MAP			MRR		
	Reported	Theano	Torch	Reported	Theano	Torch
TRAIN (full model)	0.7329	0.7325	0.7318	0.7962	0.8018	0.7950
TRAIN ($-x_{feat}$)	0.6258	0.6271	0.6408	0.6591	0.6570	0.6780
TRAIN-ALL (full model)	0.7459	0.7538	0.7428	0.8078	0.8078	0.8079
TRAIN-ALL ($-x_{feat}$)	0.6709	0.6817	0.6841	0.7280	0.7249	0.7398

Table 2: Attempts to replicate and reproduce SM results. "Reported" columns list results in the original paper. "Theano" columns list results from running the Theano code provided by SM. "Torch" columns list results from our *de novo* Torch implementation of the model. Table also presents results of removing the "extra features" from the joined representation.

columns marked "Reported" contain results copied directly from their SIGIR paper. The columns marked "Theano" contain results from our replication efforts, following the original experimental conditions. The original paper conducted an ablation analysis removing the extra features, which we also attempt to replicate (the $-x_{feat}$ condition in the table). Although we do not obtain exactly the same numbers for the different conditions, the results are quite close. Overall, we consider this replication attempt successful.

Next, we attempted to reproduce the SM model from scratch using the Torch toolkit, based as close as possible to the description in the SIGIR paper. In our efforts to reproduce the results, we attempted to match the settings described in the paper as closely as possible: choice of word embeddings, the width of the convolution filters, the number of feature maps, the optimization objective, the training regime, etc. These results are also reported in Table 2 under the columns marked "Torch". Following the original paper and our replication study, we also performed an experimental run removing the extra features (the $-x_{feat}$ condition in the table). In all cases, our numbers are quite close to both the Theano replication results and the original reported numbers in the paper—despite a completely different codebase and the use of a different deep learning toolkit. We consider this reproduction effort successful and can conclude that the model is robust to differences in the underlying deep learning toolkits.

Comparing the two implementations, we believe that our Torch code is more succinct and readable than the original Theano implementation. Our Torch implementation consists of about 50 lines of model code and about 200 lines of training code, while the Theano implementation has around 1000 lines of code for constructing the model and another 500 lines for training the model. This is because the Torch framework provides lots of modular pieces (e.g., convolution feature maps and a bilinear similarity modeling module) that are easy to combine, while the Theano version implements all these components from scratch. Although Theano offers the nice abstraction of a computation graph that supports automatic gradient computation, writing the forward computations of complex modules remains burdensome. In terms of performance, both the Torch and Theano implementations are very efficient, usually taking no more than 2–3 minutes per epoch when running on five CPU threads on a standard commodity server. We typically reach convergence in 10–20 epochs.

In Tables 3a and 3b, we report a number of results on the same experimental conditions from the literature: these are taken from an ACL wiki page that nicely summarizes the state of the art in this answer selection task [1]. We see that, without the extra features,

Reference	MAP	MRR
Yih et al. [21] (2013)	0.709	0.770
He et al. [6] (2015)	0.717	0.800
SM (full model)	0.732	0.795

(a) TRAIN

Reference	MAP	MRR
Wang et al. [17] (2015)	0.713	0.791
Miao et al. [11] (2015)	0.734	0.812
He et al. [6] (2015)	0.762	0.830
He and Lin [7] (2016)	0.755	0.825
Rao et al. [12] (2016)	0.780	0.834
SM (full model)	0.743	0.808

(b) TRAIN-ALL

Table 3: Effectiveness of the SM model on the TrecQA dataset under the TRAIN and TRAIN-ALL conditions, compared to other results from the literature.

the SM model performs well below the state of the art. With the complete model, we can see that its effectiveness is reasonably competitive but still below the best system today.

3.1 Ablation Study

At a high-level, the SM model consists of three distinct components: the output of the convolution feature maps after pooling x_q and x_d, the x_{sim} similarity matrix, and the "extra features" x_{feat}. From a philosophical perspective, it can be argued that the "extra" word overlap features go against the "spirit" of neural network modeling, in that one of the major advantages of neural networks is the avoidance of manual feature engineering (which is what these extra features represent). Instead, one hopes that the network would discover correlates of these features automatically. As a point of comparison, the work of He et al. [6] requires no additional input features other than the original sentences.

To better understand the contributions of the various components to effectiveness, we build on the ablation experiments in the previous section, removing each of the components in turn and in combination. For these experiments, we used our Torch implementation. Results are shown in Table 4.

The first three columns are the three categories of features used in the fully-connected layer, with a "–" and "+" entry denoting the removal or inclusion of each type of feature. The remaining columns show MAP and MRR scores on the TRAIN and TRAIN-ALL conditions. Note that in addition to the bilinear similarity modeling

x_q, x_d	x_{feat}	x_{sim}	TRAIN		TRAIN-ALL	
			MAP	MRR	MAP	MRR
+	−	−	0.638	0.669	0.631	0.690
		bilinear	0.641	0.678	0.684	0.740
		cosine	0.661	0.720	0.644	0.727
		dot	0.636	0.681	0.655	0.707
−	+	−	0.648	0.716	0.649	0.709
+	+	−	**0.747**	**0.812**	**0.762**	**0.815**
		bilinear	0.732	0.795	0.743	0.808
		cosine	0.735	0.774	0.742	0.804
		dot	0.684	0.740	0.735	0.794

Table 4: Results of an ablation analysis removing different components of the SM model.

($x_{sim} = x_q^T M x_d$) used in the original SM model, we also replaced it with a simple cosine similarity and dot product between the query and document feature maps. These alternative formulations of x_{sim} are also shown in the third column.

Looking at the first block of rows where the extra features x_{feat} are discarded, we see that adding the similarity feature x_{sim} consistently improves effectiveness. This is no surprise as answer selection is basically a matching task, and thus the joint representation of x_{sim} contributes additional signal beyond the query and document features. Comparing the TRAIN and TRAIN-ALL conditions, we observe a larger improvement in the data-rich condition.

In the second block of rows, we show results with only the four word overlap features. This condition essentially uses the fully-connected layer as a learning-to-rank module, which reduces the number of model parameters from ∼100K to ∼1200. Surprisingly, this simple feature engineering baseline works well, and actually better than the SM model without the extra features (Row "+ − bilinear"). Note that this simple model still contains two layers with non-linear activation between them, so it is learning useful non-linear transformations of the input features for the task. This finding suggests that the lexical matching signals captured in word overlaps are more effective than the convolution feature maps alone (given the amount of training data we have), which seems consistent with the findings of Guo et al. [5]. Note that this simple baseline is more effective than all approaches prior to around 2013 (e.g., [9, 18, 19]), according to the ACL Wiki [1].

The third block of rows shows the results of varying the x_{sim} component while retaining the other two. An interesting finding is that the model achieves the best effectiveness in both the TRAIN and TRAIN-ALL conditions when the similarity feature is removed. In other words, removing x_{sim} actually makes the model better than the full model described by SM! Furthermore, looking at the first and third blocks of rows, we see that cosine similarity is generally comparable to bilinear similarity for the x_{sim} component. Cosine similarity, however, is far simpler in requiring no parameter tuning. Dot product similarity is consistently less effective due to its lack of normalization.

The above finding, combined with the "+ − −" and "− + −" conditions, suggests that the effectiveness of the lexical word overlap features and the convolution feature maps are additive—these two components together explain the effectiveness of the SM model.

4 CONCLUSIONS

There is no doubt that deep learning applied to information retrieval is a "hot" emerging area, which makes it even more important to verify published results and to ensure that we as a community are building on a solid foundation. In this work, we have successfully replicated and reproduced a recent SIGIR paper that has gained prominence: for answer selection in question answering, we have not only validated the design of the SM model, but also deepened our understanding of how its various components interact and contribute to overall effectiveness.

REFERENCES

[1] ACL. 2017. Question Answering (State of the art). http://www.aclweb.org/aclwiki/index.php?title=Question_Answering_(State_of_the_art). Accessed: 2017-05-01.

[2] Jaime Arguello, Matt Crane, Fernando Diaz, Jimmy Lin, and Andrew Trotman. 2015. Report on the SIGIR 2015 Workshop on Reproducibility, Inexplicability, and Generalizability of Results (RIGOR). *SIGIR Forum* 49, 2 (2015), 107–116.

[3] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature Verification Using a "Siamese" Time Delay Neural Network. In *NIPS*. 737–744.

[4] Abdessamad Echihabi and Daniel Marcu. 2003. A Noisy-Channel Approach to Question Answering. In *ACL*. 16–23.

[5] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-hoc Retrieval. In *CIKM*. 55–64.

[6] Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-Perspective Sentence Similarity Modeling with Convolutional Neural Networks. In *EMNLP*. 1576–1586.

[7] Hua He and Jimmy Lin. 2016. Pairwise Word Interaction Modeling with Deep Neural Networks for Semantic Similarity Measurement. In *NAACL*. 937–948.

[8] Hua He, John Wieting, Kevin Gimpel, Jinfeng Rao, and Jimmy Lin. 2016. UMD-TTIC-UW at SemEval-2016 Task 1: Attention-Based Multi-Perspective Convolutional Neural Networks for Textual Similarity Measurement. In *SemEval*. 662–667.

[9] Michael Heilman and Noah A. Smith. 2010. Tree Edit Models for Recognizing Textual Entailments, Paraphrases, and Answers to Questions. In *HLT-NAACL*. 1011–1019.

[10] Jimmy Lin, Matt Crane, Andrew Trotman, Jamie Callan, Ishan Chattopadhyaya, John Foley, Grant Ingersoll, Craig Macdonald, and Sebastiano Vigna. 2016. Toward Reproducible Baselines: The Open-Source IR Reproducibility Challenge. In *ECIR*. 408–420.

[11] Yishu Miao, Lei Yu, and Phil Blunsom. 2015. Neural Variational Inference for Text Processing. *arXiv:1511.06038*.

[12] Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-Contrastive Estimation for Answer Selection with Deep Neural Networks. In *CIKM*. 1913–1916.

[13] Jinfeng Rao, Jimmy Lin, and Miles Efron. 2015. Reproducible Experiments on Lexical and Temporal Feedback for Tweet Search. In *ECIR*. 755–767.

[14] Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks. In *SIGIR*. 373–382.

[15] Stefanie Tellex, Boris Katz, Jimmy Lin, Gregory Marton, and Aaron Fernandes. 2003. Quantitative Evaluation of Passage Retrieval Algorithms for Question Answering. In *SIGIR*. 41–47.

[16] Ellen M. Voorhees. 2002. Overview of the TREC 2002 Question Answering Track. In *TREC*.

[17] Di Wang and Eric Nyberg. 2015. A Long Short-Term Memory Model for Answer Sentence Selection in Question Answering. In *ACL*. 707–712.

[18] Mengqiu Wang and Christopher D. Manning. 2010. Probabilistic Tree-Edit Models with Structured Latent Variables for Textual Entailment and Question Answering. In *COLING*. 1164–1172.

[19] Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy Model? A Quasi-Synchronous Grammar for QA. In *EMNLP-CoNLL*. 22–32.

[20] Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013. Answer Extraction as Sequence Tagging with Tree Edit Distance. In *HLT-NAACL*. 858–867.

[21] Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question Answering Using Enhanced Lexical Semantic Models. In *ACL*. 1744–1753.

Acknowledgments. This research was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada, with additional contributions from the U.S. National Science Foundation under CNS-1405688. We'd like to thank Aliaksei Severyn for sharing additional details about the SM model and for commenting on an earlier draft of this paper.