Denoising Bodies to Titles: Retrieving Similar Questions with Recurrent Convolutional Models

Anonymous NAACL submission

Abstract

Question answering forums are rapidly growing in size with no automated ability to refer to and reuse existing answers. In this paper, we develop a methodology for finding semantically related questions. The task is difficult since 1) key pieces of information are often buried in extraneous details in the question body and 2) available annotations are scarce and fragmented, driven largely by participants. We design a novel combination of recurrent and convolutional models (gated convolutions) to effectively map questions to their semantic representations. The models are pre-trained within an encoder-decoder framework (from body to title) on the basis of the entire raw corpus, and fine-tuned discriminatively from limited annotations. Our evaluation demonstrates that our model yields a 10% gain over a standard IR baseline, and a 5% over standard neural network architectures (including CNNs, LSTMs and GRUs) trained analogously.[1]

1 Introduction

Question answering (QA) forums such as Stack Exchange[2] are rapidly expanding and already contain millions of questions. The expanding scope and coverage of these forums often leads to many duplicate and interrelated questions, resulting in the same questions being answered multiple times. By identifying similar questions, we can potentially reuse existing answers, reducing response times and unnecessary repeated work. Unfortunately in most forums, the process of identifying and referring to existing similar questions is done manually by forum participants with limited, scattered success.

[1] Our code and data are submitted as supplementary material.

[2] http://stackexchange.com/

Title: How can I boot Ubuntu from a USB?
Body: I bought a Compaq pc with Windows 8 a few months ago and now I want to install Ubuntu but still keep Windows 8. I tried Webi but when my pc restarts it read ERROR 0x000007b. I know that Windows 8 has a thing about not letting you have Ubuntu but I still want to have both OS without actually losing all my data ...

Title: When I want to install Ubuntu on my laptop I'll have to erase all my data. "Alonge side windows" doesnt appear
Body: I want to install Ubuntu from a Usb drive. It says I have to erase all my data but I want to install it along side Windows 8. The "Install alongside windows" option doesn't appear. What appear is, ...

Figure 1: A pair of similar questions.

The task of automatically retrieving similar questions to a given user's question has recently attracted significant attention and has become a testbed for various representation learning approaches (Zhou et al., 2015; dos Santos et al., 2015). However, the task has proven to be quite challenging – for instance, dos Santos et al. (2015) report a 22.3% classification accuracy, yielding only a 4 percent gain over a simple word matching baseline.

Several factors make the problem difficult. First, submitted questions are often long and contain extraneous information irrelevant to the main question being asked. For instance, the first question in Figure 1 pertains to booting Ubuntu using a USB stick but a large portion of the body contains tangential details that are idiosyncratic to this user such as references to *Compaq pc*, *Webi* and the error message. Not surprisingly, these features are not repeated in the second question in Figure 1 about a closely related topic. The extraneous detail can easily confuse simple word-matching algorithms. Indeed, for this reason, some existing methods for question retrieval restrict attention to the question title only. While titles (when available) can succinctly summarize the intent, they also sometimes lack crucial detail available in the question body. For example, the title of the second question does not refer to installation from a USB drive. The second main reason for difficulty arises from the available annotations, which are limited and noisy. Indeed, the pairs of questions marked as similar by forum participants are largely incomplete. Our manual inspection of a sample set of questions from AskUbuntu[3] shows that only 5% of similar pairs have been annotated by the users, with a precision of around 79%.

In this paper, we design a recurrent neural network model and an associated training paradigm to address these challenges. On a high level, our model is used as an encoder to map the title, body, or the combination to a vector representation. The resulting "question vector" representation is then compared to other questions via cosine similarity. We introduce several departures from typical architectures on a finer level. In particular, we incorporate adaptive gating in non-consecutive CNNs (Lei et al., 2015) in order to focus temporal averaging in these models on key pieces of the questions. Gating plays a similar role in LSTMs (Hochreiter and Schmidhuber, 1997), though LSTMs do not reach the same level of performance in our setting. Moreover, we counter the scattered annotations available from user-driven associations by training the model largely based on the entire unannotated corpus. The encoder is coupled with a decoder and trained to reproduce the title from the noisy question body. The methodology is reminiscent of recent encoder-decoder networks in machine translation and document summarization (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014b; Rush et al., 2015). The resulting encoder is subsequently fine-tuned discriminatively on the basis of limited annotations yielding an additional performance boost.

We evaluate our model on the AskUbuntu corpus from Stack Exchange used in prior work (dos Santos et al., 2015). During training, we directly utilize noisy pairs readily available in the forum, but to have a realistic evaluation of the system performance, we manually annotate 8K pairs of questions. This clean data is used in two splits, one for development and hyper parameter tuning and another for testing. We evaluate our model and the baselines using standard information retrieval (IR) measures such as Mean Average Precision (MAP), Mean Reciprocal Rank (MRR) and Precision at n (P@n). Our full model achieves a P@1 of 64.5%, yielding 10% absolute improvement over a standard IR baseline, and 5% over standard neural network architectures (including CNNs, LSTMs and GRUs).

2 Related Work

Given the growing popularity of community QA forums, question retrieval has emerged as an important area of research. Previous work on question retrieval has modeled this task using machine translation, topic modeling and knowledge graph-based approaches (Jeon et al., 2005; Li and Manandhar, 2011; Duan et al., 2008; Zhou et al., 2013). More recent work relies on representation learning to go beyond word-based methods, aiming to capture semantic representations for more refined mappings. For instance, Zhou et al. (2015) learn word embeddings using category-based metadata information for questions. They define each question as a distribution which generates each word (embedding) independently, and subsequently use a Fisher kernel to assess question similarities. Dos Santos et al. (2015) propose an approach which combines a convolutional neural network (CNN) and a bag-of-words representation for comparing questions. In contrast to (Zhou et al., 2015), our model treats each question as a word sequence as opposed to a bag of words, and we apply a recurrent convolutional model as opposed to the traditional CNN model used by dos Santos et al. (2015) to map questions into meaning representations. Further, we propose a training paradigm that utilizes the entire corpus of unannotated questions in a semi-supervised manner.

[3] `http://askubuntu.com/`

Recent work on answer selection on community QA forums, similar to our task of question retrieval, has also involved the use of neural network architectures (Severyn and Moschitti, 2015; Wang and Nyberg, 2015; Shen et al., 2015; Feng et al., 2015; Tan et al., 2015). Similar to our work, these approaches apply neural network techniques, but focus on improving various other aspects of the model. For instance, Feng et al. (2015) explore different similarity measures beyond cosine similarity, and Tan et al. (2015) adopt the neural attention mechanism over RNNs to generate better answer representations given the questions as context.

3 Question Retrieval Setup

We begin by introducing the basic discriminative setting for retrieving similar questions. Let q be a query question which generally consists of both a title sentence and a body section. For efficiency reasons, we do not compare q against all the other queries in the data base. Instead, we retrieve first a smaller candidate set of related questions $Q(q)$ using a standard IR engine, and then we apply the more sophisticated models only to this reduced set. The goal is to rank the candidate questions in $Q(q)$ so that all the similar questions to q are ranked above the dissimilar ones. To do so, we define a similarity score $s(q, p; \theta)$ with parameters θ, where the similarity measures how closely candidate $p \in Q(q)$ is related to query q. The method of comparison can make use of the title and body of each question.

The scoring function $s(\cdot, \cdot; \theta)$ can be optimized on the basis of annotated data $D = \{(q_i, p_i^+, Q_i^-)\}$, where (q_i, p_i^+) is a correct pair of similar questions and Q_i^- is a negative set of questions deemed not similar to q_i. The candidate set during training is just $Q(q_i) = \{p_i^+\} \cup Q_i^-$. The correct pairs of similar questions are obtained from available user-marked pairs, while the negative set Q_i^- is drawn randomly from the entire corpus with the idea that the likelihood of a positive match is small given the size of the corpus. During testing, we make use of explicit manual annotations of positive and negative matches.

In the purely discriminative setting, we use a max-margin framework for learning (or fine-tuning) parameters θ. Specifically, in a context of a particular training example where q_i is paired with p_i^+, we minimize the max-margin loss $L(\theta)$ defined as

$$\max_{p \in Q(q_i)} \left\{ s(q_i, p; \theta) - s(q_i, p_i^+; \theta) + \delta(p, p_i^+) \right\},$$

where $\delta(\cdot, \cdot)$ denotes a non-negative margin. We set $\delta(p, p_i^+)$ to be a small constant when $p \neq p_i^+$ and 0 otherwise. The parameters θ can be optimized through sub-gradients $\partial L / \partial \theta$ aggregated over small batches of the training instances.

There are two key problems that remain. First, we have to define and parametrize the scoring function $s(q, p; \theta)$. We design a recurrent neural network model for this purpose and use it as an **encoder** to map each question into its meaning representation. The resulting similarity function $s(q, p; \theta)$ is just the cosine similarity between the corresponding representations. The parameters θ pertain to the neural network only. Second, in order to offset the scarcity and limited coverage of the training annotations, we pre-train the parameters θ on the basis of the much larger unannotated corpus. The resulting parameters are subsequently fine-tuned using the discriminative setup described above.

4 Recurrent Convolutional Networks

We describe here our encoder model, i.e., the method for mapping the question title and body to a vector representation. Our approach is inspired by temporal convolutional neural networks (LeCun et al., 1998) and, in particular, its recent refinement (Lei et al., 2015), tailored to capture longer-range, non-consecutive patterns in a weighted manner. Such models can be used to effectively summarize occurrences of patterns in text and aggregate them into a vector representation. However, the summary produced is not selective since all pattern occurrences are counted, weighted by how cohesive (non-consecutive) they are. In our problem, the question body tends to be very long and full of irrelevant words and fragments. Thus, we believe that interpreting the question body requires a more selective approach to pattern extraction.

Our model successively reads tokens in the question title or body, denoted as $\{\mathbf{x}_i\}_{i=1}^l$, and transforms this sequence into a sequence of states $\{\mathbf{h}_i\}_{i=1}^l$. The resulting state sequence is subsequently aggregated into a single final vector representation for each text as discussed below. Our approach builds on (Lei et al., 2015), thus we begin by briefly outlining it. Let W_1 and W_2 denote filter matrices (as parameters) for pattern size $n = 2$. Lei et al. (2015) generate a sequence of states in response to tokens according to

$$\begin{aligned}
\mathbf{c}_{t',t} &= \mathbf{W}_1\mathbf{x}_{t'} + \mathbf{W}_2\mathbf{x}_t \\
\mathbf{c}_t &= \sum_{t'<t} \lambda^{t-t'-1}\mathbf{c}_{t',t} \\
\mathbf{h}_t &= \tanh(\mathbf{c}_t + \mathbf{b})
\end{aligned}$$

where $\mathbf{c}_{t',t}$ represents a bigram pattern, $\mathbf{c}_t$ accumulates a range of patterns and $\lambda \in [0, 1)$ is a constant decay factor used to down-weight patterns with longer spans. The operations can be cast in a "recurrent" manner and evaluated with dynamic programming. The problem with the approach for our purposes is, however, that the weighting is the same for all, not triggered by the state $\mathbf{h}_{t-1}$ or the observed token $\mathbf{x}_t$.

We refine this model by learning context dependent weights. For example, if the current input token provides no relevant information (e.g., symbols, functional words), the model should ignore it by incorporating the token with a vanishing weight. In contrast, strong semantic content words such as "ubuntu" or "windows" should be included with much larger weights. To achieve this effect we introduce *neural gates* similar to LSTMs to specify when and how to average the observed signals. The resulting architecture integrates recurrent networks with non-consecutive convolutional models:

$$\begin{aligned}
\lambda_t &= \sigma(\mathbf{W}^\lambda\mathbf{x}_t + \mathbf{U}^\lambda\mathbf{h}_{t-1} + \mathbf{b}^\lambda) \\
\mathbf{c}_t^{(1)} &= \lambda_t \odot \mathbf{c}_{t-1}^{(1)} + (1-\lambda_t) \odot (\mathbf{W}_1\mathbf{x}_t) \\
\mathbf{c}_t^{(2)} &= \lambda_t \odot \mathbf{c}_{t-1}^{(2)} + (1-\lambda_t) \odot (\mathbf{c}_{t-1}^{(1)} + \mathbf{W}_2\mathbf{x}_t) \\
&\cdots \\
\mathbf{c}_t^{(n)} &= \lambda_t \odot \mathbf{c}_{t-1}^{(n)} + (1-\lambda_t) \odot (\mathbf{c}_{t-1}^{(n-1)} + \mathbf{W}_n\mathbf{x}_t) \\
\mathbf{h}_t &= \tanh(\mathbf{c}_t^{(n)} + \mathbf{b})
\end{aligned}$$

where $\sigma(\cdot)$ is the sigmoid function and $\odot$ represents the element-wise product. Here $\mathbf{c}_t^{(1)}, \cdots, \mathbf{c}_t^{(n)}$ are accumulator vectors that store weighted averages of 1-gram to n-gram features. When the gate $\lambda_t = 0$ (vector) for all t, the model represents a traditional CNN with filter width n. As $\lambda_t > 0$, however, $\mathbf{c}_t^{(n)}$ becomes the sum of an exponential number of terms, enumerating all possible n-grams within $\mathbf{x}_1, \cdots, \mathbf{x}_t$ (seen by expanding the formulas). Note that the gate $\lambda_t(\cdot)$ is parametrized and responds directly to the previous state and the token in question. We refer to this model as RCNN from here on.

In order to use the model as part of the discriminative question retrieval framework outlined earlier, we must condense the state sequence to a single vector. There are two simple alternative *pooling* strategies that we have explored – either averaging over the states[4] or simply taking the last one as the meaning representation. In addition, we apply the encoder to both the question title and body, and the final representation is computed as the average of the two resulting vectors.

Once the aggregation is specified, the parameters of the gate and the filter matrices can be learned in a purely discriminative fashion. Given that the available annotations are limited and user-guided, we instead use the discriminative training only for fine tuning an already trained model. The method of pre-training the model on the basis of the entire corpus of questions is discussed next.

4.1 Pre-training Using the Entire Corpus

The number of questions in the AskUbuntu corpus far exceeds user annotations of pairs of similar questions. We can make use of this larger raw corpus in two different ways. First, since models take word embeddings as input we can tailor the embeddings to the specific vocabulary and expressions in this corpus. To this end, we run word2vec (Mikolov et al., 2013) on the raw corpus in addition to the Wikipedia dump. Second, and more importantly, we use individual questions as training examples for an auto-encoder constructed by pairing the encoder model (RCNN) with an corresponding decoder. The resulting encoder-decoder architecture is akin to those used in machine translation (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014b) and summarization (Rush et al., 2015).

Our encoder-decoder pair represents a conditional language model $P(\text{title}|\text{context})$, where the context can be any of (a) the original title itself, (b) the question body and (c) the title/body of a similar question. All possible (title, context) pairs are used dur-

[4]We also normalize state vectors before averaging, which empirically gets better performance.

ing training to optimize the likelihood of the words (and their order) in the titles. We use the question title as the target for two reasons. The question body contains more information than the title but also has many irrelevant details. As a result, we can view the title as a distilled summary of the noisy body, and the encoder-decoder model is trained to act as a denoising auto-encoder. Moreover, training a decoder for the title (rather than the body) is also much faster since titles tend to be short (around 10 words).

The encoders pre-trained in this manner are subsequently fine-tuned according to the discriminative criterion described already in Section 3.

5 Alternative models

In order to ascertain whether RCNNs are necessary for good performance, we also train three alternative benchmark encoders (LSTMs, GRUs and CNNs) for mapping questions to vector representations. LSTM and GRU-based encoders can be pre-trained analogously to RCNNs, and fine-tuned discriminatively. CNN encoders, on the other hand, are only trained discriminatively. While plausible, neither alternative reaches quite the same level of performance as our pre-trained RCNN.

LSTMs LSTM cells (Hochreiter and Schmidhuber, 1997) have been used to capture semantic information across a wide range of applications, including machine translation and entailment recognition (Bahdanau et al., 2014; Bowman et al., 2015; Rocktäschel et al., 2015). Their success can be attributed to neural gates that adaptively read or discard information to/from internal memory states.

In our context, a LSTM model can be used similarly to a RCNN model. The model successively reads tokens $\{\mathbf{x}_i\}_{i=1}^{l}$ constituting the question title or body, and transforms this sequence into states $\{\mathbf{h}_i\}_{i=1}^{l}$. Specifically, each recurrent step takes as input the token $\mathbf{x}_t$, internal state $\mathbf{c}_{t-1}$, as well as the visible state $\mathbf{h}_{t-1}$, and generates the new states $\mathbf{c}_t, \mathbf{h}_t$:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}^i\mathbf{x}_t + \mathbf{U}^i\mathbf{h}_{t-1} + \mathbf{b}^i)\\
\mathbf{f}_t &= \sigma(\mathbf{W}^f\mathbf{x}_t + \mathbf{U}^f\mathbf{h}_{t-1} + \mathbf{b}^f)\\
\mathbf{o}_t &= \sigma(\mathbf{W}^o\mathbf{x}_t + \mathbf{U}^o\mathbf{h}_{t-1} + \mathbf{b}^o)\\
\mathbf{z}_t &= \tanh(\mathbf{W}^z\mathbf{x}_t + \mathbf{U}^z\mathbf{h}_{t-1} + \mathbf{b}^z)\\
\mathbf{c}_t &= \mathbf{i}_t \odot \mathbf{z}_t + \mathbf{f}_t \odot \mathbf{c}_{t-1}\\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned}$$

where $\mathbf{i}$, $\mathbf{f}$ and $\mathbf{o}$ are *input*, *forget* and *output* gates, respectively. Given the visible state sequence $\{\mathbf{h}_i\}_{i=1}^{l}$, we can aggregate it to a single vector exactly as with RCNNs. The LSTM encoder can be pre-trained in the same way as well.

GRUs A GRU is another comparable unit for sequence modeling (Cho et al., 2014a; Chung et al., 2014). Similar to the LSTM unit, the GRU has two neural gates that control the flow of information:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}^i\mathbf{x}_t + \mathbf{U}^i\mathbf{h}_{t-1} + \mathbf{b}^i)\\
\mathbf{r}_t &= \sigma(\mathbf{W}^r\mathbf{x}_t + \mathbf{U}^r\mathbf{h}_{t-1} + \mathbf{b}^r)\\
\mathbf{c}_t &= \tanh(\mathbf{W}\mathbf{x}_t + \mathbf{U}(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b})\\
\mathbf{h}_t &= \mathbf{i}_t \odot \mathbf{c}_t + (1 - \mathbf{i}_t) \odot \mathbf{h}_{t-1}
\end{aligned}$$

where i and r are *input* and *reset* gate respectively. Again, the GRUs can be trained in the same way.

CNNs Convolutional neural networks (LeCun et al., 1998) have also been successfully applied to various NLP tasks (Kalchbrenner et al., 2014; Kim, 2014; Kim et al., 2015; Zhang and LeCun, 2015; Gao et al., 2014). As models, they are different from LSTMs since the temporal convolution operation and associated filters map *local chunks* (windows) of the input into a feature representation. Concretely, if we let n denote the filter width, and $\mathbf{W}_1, \cdots, \mathbf{W}_n$ the corresponding filter matrices, then the convolution operation is applied to each window of n consecutive words as follows:

$$\begin{aligned}
\mathbf{c}_t &= \mathbf{W}_1\mathbf{x}_{t-n+1} + \mathbf{W}_2\mathbf{x}_{t-n+2} + \cdots + \mathbf{W}_n\mathbf{x}_t\\
\mathbf{h}_t &= \tanh(\mathbf{c}_t + \mathbf{b})
\end{aligned}$$

The sets of output state vectors $\{\mathbf{h}_t\}$ produced in this case are typically referred to as feature maps. Since each vector in the feature map only pertains to local information, the last vector is not sufficient to capture the meaning of the entire sequence. Instead, we consider *max-pooling* or *average-pooling* to obtain the aggregate representation for the entire sequence.

6 Experimental Setup

Dataset We use the Stack Exchange AskUbuntu dataset used in prior work (dos Santos et al., 2015). This dataset contains 167,765 unique questions, each consisting of a title and a body[5], and a set of user-marked similar question pairs. We provide various statistics from this dataset in Table 1.

[5] We remove stop words and limit body length to 100.

Corpus	# of unique questions	167K
	Avg length of title	5.8
	Avg length of body	39.6
Training	# of unique questions	12,584
	# of user-marked pairs	16,391
Dev	# of query questions	200
	# of annotated pairs	200×20
	Avg # of positive pairs per query	5.8
Test	# of query questions	200
	# of annotated pairs	200×20
	Avg # of positive pairs per query	5.5

Table 1: Various statistics from our Training, Dev, and Test sets derived from the Sept. 2014 Stack Exchange AskUbuntu dataset.

Task Setup and Annotations User-marked similar question pairs on QA sites are often known to be incomplete. In order to evaluate this in our dataset, we took a sample set of questions paired with 20 candidate questions retrieved by a search engine trained on the Ubuntu data. The search engine used is the well-known BM25 model (Robertson and Zaragoza, 2009). Our manual evaluation of the candidates showed that only 5% of the similar questions were marked by users, with a precision of 79%. Clearly, this low recall would not lead to a realistic evaluation if we used user marks as our gold standard. Thus, we needed manual annotations for the dev and test sets. Unfortunately, annotating all pairs (hundreds of thousands) is too costly also because this task requires experts in the domain and consequently it is not suitable for Mechanical Turk-based approaches. For this reason, we formulated the problem as a re-ranking task of the first 20 most similar questions retrieved by the BM25 model. This choice is rather reasonable as, in a real-world scenario, the user would like to see just a short list of similar questions.

Training Set We use user-marked similar pairs as positive pairs in training since the annotations have high precision and do not require additional manual annotations, allowing us to use a much larger training set. We use random questions from the corpus paired with each query question p_i as negative pairs in training. We randomly sample 20 questions as negative examples for each p_i.

Dev and Test Sets We re-constructed the new dev and test sets consisting of the first 200 questions from the dev and test sets provided by dos Santos et al. (2015). For each of the above questions, we retrieved the top 20 similar candidates using BM25 trained on Ubuntu and manually annotated the resulting 8K pairs as similar or non-similar.[6]

	d	$\|\theta\|$	n	Pooling
LSTMs	240	423K	-	mean-pooling
GRUs	280	404K	-	mean-pooling
CNNs	667	401K	3	mean-pooling
RCNNs	400	401K	2	last state

Table 2: The configuration of neural network models tuned on the dev set. d is the hidden dimension, $|\theta|$ is the number of parameters and n is the filter width of the convolution operation.

Baselines and Evaluation Metrics We evaluated neural network models—including **CNNs**, **LSTMs**, **GRUs** and **RCNNs**—by comparing them with the following baselines:

- **BM25**, we used the BM25 similarity measure provided by Apache Lucene.
- **TF-IDF**, we ranked questions using cosine similarity based on a vector-based word representation for each question.
- **SVM**, we trained a re-ranker using SVM-Light (Joachims, 2002) with a linear kernel incorporating several similarity measures from the DKPro similarity package (Bär et al., 2013).

We evaluated the models based on the following IR metrics: Mean Average Precision (MAP), Mean Reciprocal Rank (MRR), Precision at 1 (P@1), and Precision at 5 (P@5).

Hyper-parameters We performed an extensive hyper-parameter search to identify the best model for the baselines and neural network models. For the **TF-IDF** baseline, we tried n-gram feature order $n \in \{1, 2, 3\}$ with and without stop words pruning. For the **SVM** baseline, we used the default SVM-Light parameters whereas the dev data is only used to increase the training set size when testing on the

[6]The annotation task was initially carried out by two expert annotators, independently. The initial set was refined by comparing the annotations and asking a third judge to make a final decision on disagreements. After a consensus on the annotation guidelines was reached (producing a Cohen's kappa of 0.73), the overall annotation was carried out by only one expert.

Method	Dev				Test			
	MAP	MRR	P@1	P@5	MAP	MRR	P@1	P@5
BM25	52.0	66.0	51.9	42.1	56.0	68.0	53.8	42.5
TF-IDF	54.1	68.2	55.6	45.1	53.2	67.1	53.8	39.7
SVM	53.5	66.1	50.8	43.8	57.7	71.3	57.0	43.3
LSTMs	58.7	72.5	60.0	47.1	58.1	71.2	58.3	43.3
CNNs	58.3	73.0	61.1	47.0	57.8	71.3	58.0	43.4
GRUs	58.9	73.3	61.4	47.1	58.5	72.0	59.4	43.5
RCNNs	59.9	73.4	61.8	49.4	62.6	75.0	64.2	46.0
LSTMs + pre-training	58.8	73.4	60.6	47.7	59.0	70.3	56.7	43.5
GRUs + pre-training	59.7	74.3	62.1	47.2	59.5	71.9	58.8	44.2
RCNNs + pre-training	**61.5**	**75.8**	**65.2**	**49.9**	**63.9**	**76.2**	**64.5**	**47.8**

Table 3: Comparative results of all methods on the question similarity task

test set. We also tried to give higher weight to dev instances but this did not result in any improvement.

For all the neural network models, we used Adam (Kingma and Ba, 2014) as the optimization method with the default setting suggested by the authors. We optimized other hyper-parameters with the following range of values: learning rate $\in \{1e-3, 3e-4\}$, dropout (Hinton et al., 2012) probability $\in \{0.1, 0.2, 0.3\}$, CNN feature width $\in \{2, 3, 4\}$. We also tuned the pooling strategies and ensured each model has a comparable number of parameters. The default configurations of LSTMs, GRUs, CNNs and RCNNs are shown in Table 2. We used MRR to identify the best training epoch and the model configuration. For the same model configuration, we report average performance across 5 independent runs.[7]

Word Vectors We ran word2vec (Mikolov et al., 2013) to obtain 200-dimensional word embeddings using all Stack Exchange data (excluding StackOverflow) and a large Wikipedia corpus. The word vectors are fixed to avoid over-fitting across all experiments.

7 Results

7.1 Overall Performance

Table 3 shows the performance of the baselines and the neural encoder models on the question similarity task. The results show that our full model, RCNNs with pre-training, achieves the best performance across all metrics on both the dev and test sets. For instance, the full model gets a P@1 of 64.5% on the test set, outperforming the word matching-based method BM25 by over 10 percent points. Further, our RCNN model also outperforms the other neural encoder models and the baselines across all metrics. The ability of the RCNN model to outperform the other models indicates that the use of non-consecutive filters and a varying decay factor is effective in improving performance beyond traditional neural network models.

Table 3 also demonstrates the performance gain from pre-training the RCNN encoder. The RCNN model when pre-trained on the entire corpus consistently gets better results across all the metrics.

7.2 Discussion

Pooling strategy We analyze the effect of various pooling strategies for the neural network encoders. As shown in Table 4, our RCNN model outperforms other neural models regardless of the two pooling strategies explored. We also observe that simply using the last hidden state as the final representation achieves better results for the RCNN model.

Question body Table 5 compares the performance of the TF-IDF baseline and the RCNN model when using question titles only or when using question titles along with question bodies. TF-IDF's performance changes very little when the question bodies are included (MRR and P@1 are slightly better but MAP is slightly worse). However, we find that the inclusion of the question bodies improves the performance of the RCNN model, achieving a 2% to

[7] For pre-training experiments, we explore pre-trained models with different learning rates $\in \{1e-3, 3e-4, 1e-4\}$, and pick the model with best dev performance. We then perform 5 independent fine-tuning runs.

Method	Dev				Test			
	MAP	MRR	P@1	P@5	MAP	MRR	P@1	P@5
CNNs, max-pooling	58.8	72.2	59.9	47.3	58.2	71.4	57.6	44.9
CNNs, mean-pooling	58.3	73.0	61.1	47.0	57.8	71.3	58.0	43.4
LSTMs, last state	56.9	70.4	57.6	46.1	57.8	69.8	56.6	42.8
LSTMs, mean-pooling	58.7	72.5	60.0	47.1	58.1	71.2	58.3	43.3
GRUs, last state	58.6	72.2	59.0	48.0	60.3	72.5	61.3	45.2
GRUs, mean-pooling	58.9	73.3	61.4	47.1	58.5	72.0	59.4	43.5
RCNNs, last state	**59.9**	73.4	61.8	**49.4**	**62.6**	**75.0**	**64.2**	**46.0**
RCNNs, mean-pooling	59.7	**74.3**	**62.5**	48.6	59.6	71.9	58.5	44.9

Table 4: Choice of pooling strategies

TF-IDF	MAP	MRR	P@1
title only	54.3	66.8	52.7
title + body	53.2	67.1	53.8
RCNNs, mean-pooling	**MAP**	**MRR**	**P@1**
title only	55.6	68.7	54.8
title + body	59.6	71.9	58.5
RCNNs, last state	**MAP**	**MRR**	**P@1**
title only	58.9	73.0	61.5
title + body	62.6	75.0	64.2

Table 5: Comparision between model variants when question bodies are used or not used. Numbers are reported on the test set.

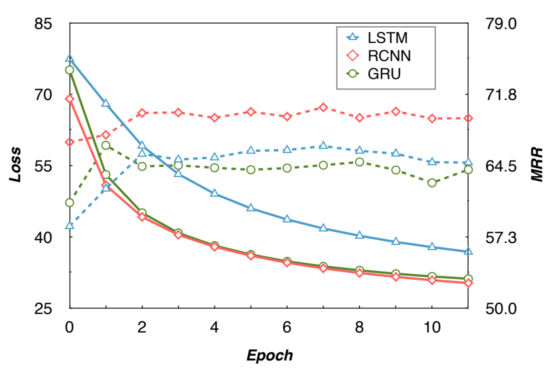

Figure 2: Training loss (solid lines) versus MRR on the dev set (dotted lines) during pre-training. Red lines with diamonds are RCNNs, blue lines with triangles are LSTMs and green lines with circles are GRUs.

4% improvement with both model variations. The RCNN model's greater improvement as compared to TF-IDF's from the inclusion of the question bodies illustrates the ability of the model to pick out components that pertain most directly to the question being asked from the long, descriptive question bodies.

Pre-training Note that, during pre-training, the last hidden states generated by the neural encoder are used by the decoder to reproduce the question titles. It would be interesting to see how such states capture the meaning of questions. As shown in Figure 2, we compute the question similarities using these representations and evaluate MRR on the dev set. The representations generated by the RCNN encoder perform quite well, resulting in over 70% MRR without the subsequent fine-tuning.

The LSTM and GRU networks do not perform as well as the RCNN model across a range of learning rates and dropout rates during pre-training. The GRU encoder obtains only 66% MRR, 4% worse than the RCNN model's MRR performance. Consequently, we do not observe a clear improvement by fine-tuning the LSTM or GRU encoder (when comparing with the result without pre-training and fine-tuning).

8 Conclusion

In this paper, we employ gated convolutions to map questions to their semantic representations, and demonstrate their effectiveness on the task of question retrieval in community QA forums. This architecture enables the model to glean key pieces of information from lengthy, detail-riddled user questions. Pre-training within an encoder-decoder framework (from body to title) on the basis of the entire raw corpus is integral to the model's success.

In future work, we plan to expand this model for the task of answer selection. Since this task has similar challenges to question retrieval, both pre-training and gated convolutions are likely to benefit the overall performance. In addition, we plan to employ pre-trained representations for the title generation task.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Daniel Bär, Torsten Zesch, and Iryna Gurevych. 2013. Dkpro similarity: An open source framework for text similarity. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 121–126, Sofia, Bulgaria, August. Association for Computational Linguistics.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Cicero dos Santos, Luciano Barbosa, Dasha Bogdanova, and Bianca Zadrozny. 2015. Learning hybrid representations to retrieve semantically equivalent questions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 694–699, Beijing, China, July. Association for Computational Linguistics.

Huizhong Duan, Yunbo Cao, Chin-Yew Lin, and Yong Yu. 2008. Searching questions by identifying question topic and question focus. In *ACL*, pages 156–164.

Minwei Feng, Bing Xiang, Michael R Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. *arXiv preprint arXiv:1508.01585*.

Jianfeng Gao, Patrick Pantel, Michael Gamon, Xiaodong He, Li Deng, and Yelong Shen. 2014. Modeling interestingness with deep neural networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jiwoon Jeon, W Bruce Croft, and Joon Ho Lee. 2005. Finding similar questions in large question and answer archives. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 84–90. ACM.

T. Joachims. 2002. Optimizing search engines using clickthrough data. In *ACM SIGKDD KDD*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, pages 1700–1709.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2015. Character-aware neural language models. *arXiv preprint arXiv:1508.06615*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2015. Molding cnns for text: non-linear, non-consecutive convolutions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1565–1575, Lisbon, Portugal, September. Association for Computational Linguistics.

Shuguang Li and Suresh Manandhar. 2011. Improving question recommendation by exploiting information need. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1425–1434. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*.

Stephen Robertson and Hugo Zaragoza. 2009. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *SIGIR*.

Yikang Shen, Wenge Rong, Nan Jiang, Baolin Peng, Jie Tang, and Zhang Xiong. 2015. Word embedding based correlation model for question/answer matching. *arXiv preprint arXiv:1511.04646*.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ming Tan, Bing Xiang, and Bowen Zhou. 2015. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*.

Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. In *ACL*, July.

Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*.

Guangyou Zhou, Yang Liu, Fang Liu, Daojian Zeng, and Jun Zhao. 2013. Improving question retrieval in community question answering using world knowledge. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 2239–2245. AAAI Press.

Guangyou Zhou, Tingting He, Jun Zhao, and Po Hu. 2015. Learning continuous word embedding with metadata for question retrieval in community question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 250–259, Beijing, China, July. Association for Computational Linguistics.