Finding Similar Questions in Large-Scale Community QA Forums

by

Hrishikesh S. Joshi

B.S., Computer Science & Mathematics, MIT (2013)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2016

Author ..... Signature redacted .....
Department of Electrical Engineering and Computer Science
January 27, 2016

Certified by .. Signature redacted .....
Regina Barzilay
Professor
Thesis Supervisor

Accepted by ..... Signature redacted .....
Christopher J. Terman
Chairman, Department Committee on Graduate Theses

Finding Similar Questions in Large-Scale Community QA Forums

by

Hrishikesh S. Joshi

Submitted to the Department of Electrical Engineering and Computer Science
on January 27, 2016, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

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 6% gain over standard neural network architectures (including CNNs and LSTMs) trained analogously.[1]

Thesis Supervisor: Regina Barzilay
Title: Professor

[1]Code and data are available at `https://github.com/taolei87/rcnn`

Acknowledgments

I'd like to first and foremost thank my thesis supervisor, Prof. Regina Barzilay, for providing me the opportunity to do this work in the past year. She found time on multiple days per week despite her busy schedule to meet with me when I had questions or needed help, and I value her mentorship and genuine caring for my work on this thesis and personal learning more broadly very much. I've learned a tremendous amount in the course of the year, and I owe a large part of it to her patience in allowing me to persevere and work through issues that inevitably came up over an extended period of time over and over again.

I'd further like to thank our lab group for serving as a great source of learning and encouragement throughout the year. In particular, I'd like to thank Karthik Narasimhan and Tao Lei for their help. Karthik provided an incredible amount of guidance when I first started, and Tao's collaboration was instrumental in helping us overcome a lot of efficiency-related issues we later had. I'd like to thank Prof. Tommi Jaakkola from MIT, along with Prof. Alessandro Moschitti and Prof. Lluis Marquez from the Qatar Computing Research Institute (QCRI), for their collaboration on this work as well.

I'd like to thank my parents, Santosh and Nilima Joshi, for their never-ending support, and my numerous friends still living in the Cambridge area post-undergrad who made my time here in the past year memorable as always.

Contents

1 Introduction **13**

2 Related Work **17**

2.1 Question Retrieval 17

2.2 Answer Selection 19

2.3 Deep Learning Techniques 21

3 Question Retrieval Setup **23**

4 Neural Network Approaches **25**

4.1 Convolutional Neural Networks (CNNs) 25

4.1.1 Overview 25

4.1.2 Modifications for Text Processing 26

4.2 Long Short-Term Memory Networks (LSTMs) 28

5 Model & Training Methodology **31**

5.1 Recurrent Convolutional Neural Networks (RCNNs) 31

5.1.1 Context Dependent Weights 32

5.1.2 Pooling Strategies 33

5.2 Pre-training Using the Entire Corpus 33

6 Experimental Setup **35**

6.1 Dataset 35

6.2 Task Setup and Annotations 35

6.2.1 Training Set 36
6.2.2 Dev and Test Sets 36
6.3 Baselines and Evaluation Metrics 37
6.4 Hyperparameters 37
6.5 Word Vectors 38

7 Results **39**
7.1 Overall Performance 39
7.2 Discussion 39
7.3 Re-Ranking Analysis 42

8 Conclusion **45**
8.1 Contributions 45
8.2 Future Work 46

List of Figures

1-1 A pair of similar questions. 14

4-1 Sample CNN architecture consisting of multiple convolutions and pooling layers (Kalchbrenner and Blunsom, 2013) 27

7-1 Training loss (solid lines) versus MRR on the dev set (dotted lines) during pre-training. Red lines with diamonds are RCNNs and blue lines with triangles are LSTMs. 41

7-2 Pairs of similar questions from the test set which the RCNN model re-ranks better than the baseline models. The left column corresponds to the query question, and the right column corresponds to a question marked as similar in the 20 questions provided by BM25 based on the query question. 42

7-3 Pairs of similar questions from the test set which the RCNN model re-ranks better when the title & body are both used as opposed to title only. The left column corresponds to the query question, and the right column corresponds to a question marked as similar in the 20 questions provided by BM25 based on the query question. 43

List of Tables

6.1 Various statistics from our Training, Dev, and Test sets derived from the Sept. 2014 Stack Exchange AskUbuntu dataset. 36

7.1 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. 40

7.2 Comparative results of all methods on the question similarity task 40

7.3 Choice of pooling strategies 40

7.4 Comparison between model variants when question bodies are used or not used. Numbers are reported on the test set. 41

Chapter 1

Introduction

Question answering (QA) forums such as Stack Exchange[1] 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 bloat. Unfortunately in most forums, the process of identifying and referring to existing similar questions is done manually by forum participants with limited, scattered success.

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 1-1 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 1-1 about a closely related topic. The extraneous detail can easily confuse simple word-matching algorithms. Indeed, for this reason, some

[1] `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 1-1: A pair of similar questions.

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[2] showed 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 corpus. The

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

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., 2014; 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 hyperparameter 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 6% over standard neural network architectures (including CNNs and LSTMs).

Chapter 2

Related Work

The growing popularity of community QA forums has resulted in the use of community QA datasets in performing various tasks including question retrieval and answer selection. Question retrieval, the task addressed in this work, involves finding similar existing questions in the corpus given a query question. Answer selection is a closely related task that involves finding relevant existing answers given a query question, and can include finding similar questions as part of the approach though often consists of various other techniques including the use of answer-answer similarity. We hereby provide an overview of previous approaches taken in performing these tasks using community QA datasets, along with recent applications of deep learning techniques most closely related to our work.

2.1 Question Retrieval

Previous work on question retrieval in the community QA domain has used 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 has relied on representation learning to go beyond word-based methods, aiming to capture semantic representations for more refined mappings. We hereby outline various approaches in question retrieval on community QA datasets in further detail.

Translation Models Translation models previously used for this task have involved using a language modeling framework to learn word-to-word translation probabilities between words in the titles of questions (Jeon et al., 2005; Li and Manandhar, 2011). This approach represents a statistical machine learning way to assess similarity between pairs of questions. Given a pair of questions (q_1, q_2), such models generally represent the similarity $S(q_1, q_2)$ as follows:

$$S(q_1, q_2) = \prod_{w \in q_1} P(w|q_2)$$
$$P(w|q_2) = \sum_{t \in q_2} T(w|t)P(t|q_2)$$

where $T(w|t)$ refers to the probability that w is a translation of t. $T(w|w)$ is generally set to 1, and a smoothing parameter not provided here is further used to avoid zero values for $P(w|q_2)$.

Topic Modeling Duan et al. (2008) use a topic modeling approach that identifies question topic and focus for a set of questions using a tree-based model, and subsequently use this information in a language modeling framework to assess question similarity. Their work makes use of categorial information provided in the Yahoo! Answers dataset such as 'Computers & Internet -> Computer Networking.' Given a set of questions Q from a set of categories C, they generate a set of topics T consisting of certain n-grams such as nouns from the question titles, and model a topic profile as follows:

$$p(c|t) = \frac{count(c,t)}{\sum_{c \in C} count(c,t)}$$

where $count(c, t)$ denotes the frequency of topic term t in category c. Further, they define the specificity of a topic term based on the inverse of the topic profile such that a topic term with high specificty would appear in fewer categories and likely be more specific to the question as opposed to a topic term with lower specificity. Based on this, they construct a topic chain for each question with the chain of topics ordered in decreasing order of specificity, and form a tree consisting of such topic chains for the set of all questions Q. Next, they use a language modeling approach to determine a cut in the tree, labeling the nodes above the cut in the tree as the question topic (high specificity) and nodes below the cut in the tree as

the question focus (low specificity), and subsequently use this information to assess question similarity.

Knowledge Graphs Zhou et al. (2013) use a knowledge-graph based approach that leverages Wikipedia entries to identify concepts and relations between questions. They initially preprocess Wikipedia to collect a set of concepts along with semantic relations including synonyms, polysemy, hypernym, and associative relations. Next, they use a set of techniques including word disambiguation to map words in questions to relevant Wikipedia entries. Given such mappings, they devise a method for using the semantic relations between the Wikipedia entries representing the questions to assess question similarity.

Convolutional Neural Networks (CNNs) Recent work has used novel techniques incorporating neural networks for the question retrieval task. Zhou et al. (2015) propose a method for learning 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 similarity. Dos Santos et al. (2015) propose an approach which combines a convolutional neural network (CNN) and a bag-of-words representation for comparing questions.

2.2 Answer Selection

Recent work on answer selection using community QA datasets has also involved the use of neural network architectures, in particular CNNs and LSTMs (Severyn and Moschitti, 2015; Wang and Nyberg, 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. We hereby outline previous neural network approaches in answer selection using community QA datasets in further detail.

Long Short-Term Memory Networks (LSTMs) Wang and Nyberg (2015) use a bi-directional, stacked LSTM model (refer to Section 4.2) to model questions and answers together. The set of word vectors corresponding to an answer is appended to the set of word vectors corresponding to the respective question, and they add a special vector in between to demarcate the split between the question and answer. The bi-directional feature of the model allows it to incorporate context before and after the word being processed as opposed to only using context from the previous word. The bi-directional and stacked features of the LSTM model serve to provide bi-directional context and greater learning ability respectively, and have been shown to improve performance when applied to recursive neural network models. Tan et al. (2015) use a similar bi-directional LSTM model that allows contextual information from both sides of a word to be used, but further have an attention-based component that uses a softmax value at each step to weight the output vector, feeding the output to a CNN layer. At each step, the LSTM model computes a softmax value based on the question embedding, and subsequently multiplies the output vector resulting from the answer embedding prior with it to the average or mean pooling step. Then, the output vector resulting from the original question representation and the modified answer is used as input to a traditional CNN model. The results show that their full model, consisting of the attention-based bi-directional LSTM with a CNN layer, achieves the best performance when compared to other related models on the TREC-QA dataset.

Convolutional Neural Networks (CNNs) Severyn and Moschitti (2015) use a traditional CNN network for creating representations of questions and answers, but instead of using cosine similarity for comparing representations[they learn an embedding matrix that operates on the question and answer representations. Further, they augment the resulting vector with additional features, and use a softmax layer to generate a similarity metric for the question and answer. Feng et al. (2015) use a traditional CNN network, and propose a set of different architectures in terms of how the question and answer representations are modeled as well as a set of comparison metrics. The different architectures vary in terms of whether weights are shared and how many hidden layers are placed before and after the CNN layer, and the results show that sharing weights and placing a hidden layer both before

and after the CNN layer provides the best performance. Further, they vary the comparison metric used, including cosine similarity, various functions such as polynomial, sigmoid, and exponential, and Geometric and Arithmetic mean of Euclidean and Sigmoid Dot product (GESD and AESD, respectively). They find that GESD and AESD outperform cosine similarity and all other functions as a metric for comparison of the output representations from the CNN model.

2.3 Deep Learning Techniques

In recent years, deep neural networks have been shown to perform well across a number of natural language processing tasks including language modeling (Bengio et al., 2003; Mikolov et al., 2010), sentiment analysis (Socher et al., 2013; Iyyer et al., 2015; Le and Zuidema, 2015), question answering (Iyyer et al., 2014), syntactic parsing (Collobert and Weston, 2008; Socher et al., 2011; Chen and Manning, 2014) and machine translation (Bahdanau et al., 2014; Devlin et al., 2014; Sutskever et al., 2014). The models proposed have varied primarily in terms of model architecture, and have included recurrent neural networks (Mikolov et al., 2010; Kalchbrenner and Blunsom, 2013), recursive models (Pollack, 1990; Küchler and Goller, 1996), and most similar to our proposed model, convolutional neural nets (Collobert and Weston, 2008; Collobert et al., 2011; Yih et al., 2014; Shen et al., 2014; Kalchbrenner et al., 2014; Zhang and LeCun, 2015; Lei et al., 2015).

The model proposed in this work is a recurrent convolutional neural network, most closely related to the work by Lei et al. (2015). Lei et al. (2015) use convolutions that apply to all non-consecutive words with a fixed decay rate which determines the level to which the prior context is used in computing the representations. We refine this model by introducing a variable decay rate based on the context such that it decreases the weights given to the prior context if the current context is relevant, and decreases the weights otherwise. Further, we introduce a semi-supervised learning approach that leverages all of the questions in the corpus, allowing us to initialize the parameter weights in the model based on pre-training that focuses on enabling the model to learn constrained representations of questions.

Chapter 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 annotations, 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 also make use of explicit manual annotations of positive matches.

In the purely discriminative setting, we use a max-margin framework for learning (or fine-tuning) parameters θ. Specifically, in the 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 corresponding 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.

Chapter 4

Neural Network Approaches

4.1 Convolutional Neural Networks (CNNs)

4.1.1 Overview

CNNs (LeCun et al., 1998) have traditionally been applied to tasks related to image recognition, providing techniques for identifying objects and other attributes within images while taking into account distortion and invariance, but have more recently also been successfully applied to various NLP tasks including sentiment analysis, semantic parsing, and question-answering (Kalchbrenner et al., 2014; Kim, 2014; Kim et al., 2015; Zhang and LeCun, 2015; Gao et al., 2014).

CNNs are characterized by convolutions that enable the use of hierarchical features, shared weights, and pooling layers. A convolution refers to the operation between a weight vector $\mathbf{W}$ and a sequence $\mathbf{x}$ representing the sentence, where $\mathbf{x}_i$ is generally represented by a vector such as a word embedding. The weight vector $\mathbf{W}$ represents the filter of the convolution. Narrow convolutions consist of $|\mathbf{W}| \leq |\mathbf{x}|$, and result in a feature map of size $|\mathbf{x}| - |\mathbf{W}| + 1$, whereas wide convolutions have no restrictions on the relative lengths and result in a feature map of size $|\mathbf{x}| + |\mathbf{W}| - 1$. 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:

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

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. As shown in Figure 4-1, a CNN model can consist of multiple convolutions and pooling layers operating on varying hierarchical representations of the input.

4.1.2 Modifications for Text Processing

Lei et al. (2015) recently showed significant improvements in common tasks such as sentiment classification and news categorization resulting from further extensions of the temporal CNN model for text processing. In particular, they propose a convolution over non-consecutive words with a decay factor as opposed to using consecutive words only, and provide an efficient method for performing feature mapping operations based on tensor products as opposed to linear operations.

Let us consider an input sentence S represented by $x \in \mathbb{R}^{dxL}$, where d denotes the dimensionality of vector x_i and L denotes the length of sentence S.

Non-Linear Operations

Traditional CNN architectures model n-grams by concatenating vectors x_i through x_{i+n-1}, and subsequently learning a weight matrix W of length dn. Lei et al. (2015) instead propose non-linear operations that replace the concatenation step by a tensor product. In place of a weight matrix W as described, Lei et al. (2015) model a 3-gram (W_1, W_2, W_3) with the corresponding tensor product $W_1 \otimes W_2 \otimes W_3$, expanding the n-gram sequence into a higher dimensional tensor using tensor products. The set of resulting h filters thereby have dimensionality $d\ x\ d\ x\ d\ x\ h$, denoting a set of filters capable of learning high-dimensional

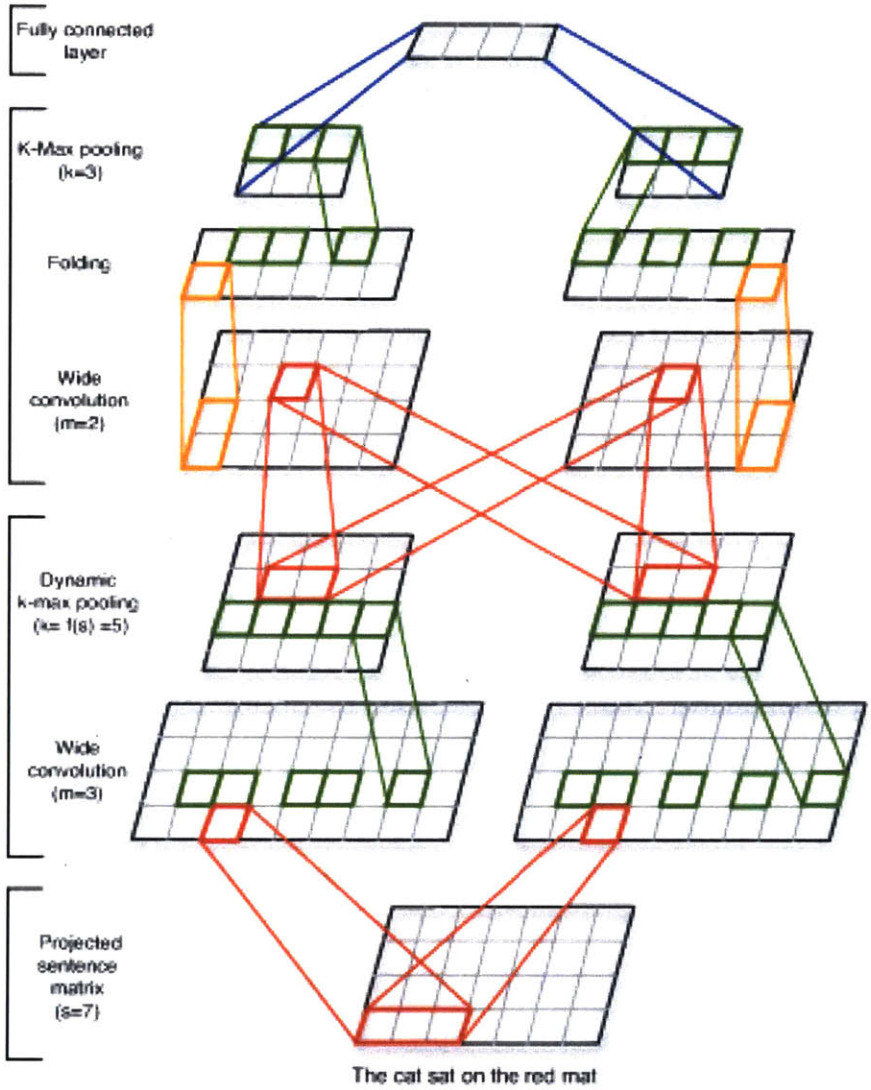

Figure 4-1: Sample CNN architecture consisting of multiple convolutions and pooling layers (Kalchbrenner and Blunsom, 2013)

interactions between n-gram features. The size of the set of h filters leads to parametric explosion for common values of d such as $d = 300$, and Lei et al. (2015) provide an efficient method for representing the tensor products in Kruskal form using low-rank factorization.

Modeling Non-Consecutive Words

Applications of CNN models to text processing have generally involved learning parameters based on n-gram features representing consecutive words. Lei et al. (2015) propose the use of non-consecutive words in learning, using a decay factor that provides greater importance to n-grams closer together. The motivation behind this extension is to enable the model to more accurately learn phrases such as "it was not so good", where sub-phrases such as "not ... good" provide relevant meaning when taken together but are not necessarily consecutive in terms of their word ordering in the sentence. Lei et al. (2015) provide an efficient implementation using dynamic programming that enables modeling of non-consecutive n-grams with a runtime linear in terms of the sequence length of sentence S.

4.2 Long Short-Term Memory Networks (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.

The LSTM 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 of the LSTM network 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 pair of states $\mathbf{c}_t, \mathbf{h}_t$ according to

$$\mathbf{i}_t = \sigma(\mathbf{W}^i\mathbf{x}_t + \mathbf{U}^i\mathbf{h}_{t-1} + \mathbf{b}^i)$$
$$\mathbf{f}_t = \sigma(\mathbf{W}^f\mathbf{x}_t + \mathbf{U}^f\mathbf{h}_{t-1} + \mathbf{b}^f)$$

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

where **i**, **f** and **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.

Chapter 5

Model & Training Methodology

5.1 Recurrent Convolutional Neural Networks (RCNNs)

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 effective 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

$$\mathbf{c}_{t',t} = \mathbf{W}_1\mathbf{x}_{t'} + \mathbf{W}_2\mathbf{x}_t$$

$$\mathrm{c}_t = \sum_{t'<t} \lambda^{t-t'-1} \mathbf{c}_{t',t}$$
$$\mathrm{h}_t = \tanh(\mathbf{c}_t + \mathbf{b})$$

where $\lambda \in [0, 1)$ 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 $\mathbf{x}_t$.

5.1.1 Context Dependent Weights

We refine this model by learning context dependent weights. For example, if the current input token provides no relevant information (e.g., stop words, punctuation), 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:

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

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.

5.1.2 Pooling Strategies

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[1] 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.

5.2 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 the 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., 2014) 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 during 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

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

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 de-noising 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.

Chapter 6

Experimental Setup

6.1 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, and a set of user-marked similar question pairs. We provide various statistics from this dataset in Table 6.1.

6.2 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

Corpus	# of unique questions	167K
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 6.1: Various statistics from our Training, Dev, and Test sets derived from the Sept. 2014 Stack Exchange AskUbuntu dataset.

reasonable as, in a real-world scenario, the user would like to see a short list of similar questions.

6.2.1 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 p_i as negative pairs in training. We randomly sample 20 questions as negative examples for each query question p_i.

6.2.2 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[1].

[1]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.

6.3 Baselines and Evaluation Metrics

We evaluated the neural network models—including **CNNs**, **LSTMs** 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). This model has been shown to provide state-of-the-art results in sentence similarity challenges.

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).

6.4 Hyperparameters

We performed an extensive hyperparameter grid search to identify the best models 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 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 hyperparameters 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 had a comparable number of parameters. The default configurations of LSTMs, CNNs and RCNNs are shown in Table 7.1. 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.

6.5 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.

Chapter 7

Results

7.1 Overall Performance

Table 7.2 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 7.2 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 7.3, our RCNN model outperforms CNNs and LSTMs regardless of the two pooling strategies explored. We also observe that simply using the last hidden

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

Table 7.1: 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.

state as the final representation achieves better results for the RCNN model.

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
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
RCNNs + pre-training	**61.5**	**75.8**	**65.2**	**49.9**	**63.9**	**76.2**	**64.5**	**47.8**

Table 7.2: Comparative results of all methods on the question similarity task

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
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 7.3: Choice of pooling strategies

Question body Table 7.4 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 4% improvement with both model variations. The RCNN model's greater improvement

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 7.4: Comparison between model variants when question bodies are used or not used. Numbers are reported on the test set.

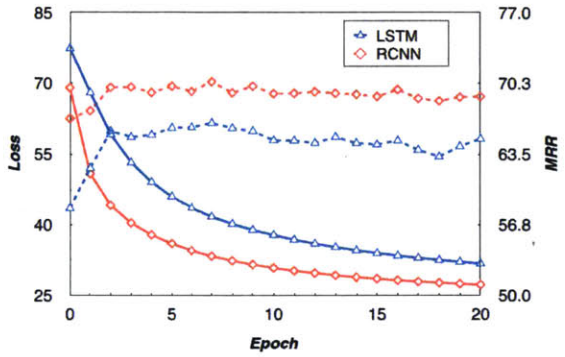

Figure 7-1: Training loss (solid lines) versus MRR on the dev set (dotted lines) during pre-training. Red lines with diamonds are RCNNs and blue lines with triangles are LSTMs.

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 7-1, 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 network does not perform as well as the RCNN model across a range of learning rates and dropout rates during pre-training. The LSTM encoder obtains only 65% MRR, 5% worse than the RCNN model's MRR performance. As a result, we do not observe

Questions	Similar Questions
Title: What is the difference between installing Ubuntu on a USB device and a laptop hardisk? **Body:** Now, I have a laptop with Windows 8. For various reasons, I want to install Ubuntu that I can carry with me on the various PCs I work with. The same installation so that I don't have to constantly take care of installing new things and dependencies...	**Title:** Ubuntu installation on USB **Body:** I have a 8GB pendrive which i want to use as my HDD I want the Ubuntu 14.04 to be installed on USB and i want possibility to save files on the USB, update the system and so on...
Title: How can I manually edit the unity top panel? **Body:** After the recent update of 11.10, I am left with two battery indicators, for some reason–the new one, with the options to show the remaining time and bring up the power settings, and the old one, which doesn't do anything when clicked. How can I manually edit the top panel listings to remove this artifact?	**Title:** How to rearrange the panel indicators? **Body:** How do I move Panel (Not Launcher) icons in Ubuntu Unity? I'm using Ubuntu 11.10 x64 with Unity environment and I need to rearrange the indicators on the top panel. How can I do that?
Title: Install ubuntu from USB drive **Body:** I have a laptop with no cd/dvd drive. I want to remove win 8 and put ubuntu instead. I am not sure if there is UEFI in this laptop (its the cheapest). I don't want to buy a usb optical drive just for this. Is there a reliable and EASY way to install ubuntu from a USB drive onto my new laptop ?...	**Title:** Can't boot from USB when installing Ubuntu **Body:** I downloaded Ubuntu 11.10 32bit and put it in on USB drive using Universal USB Installer fromhttp://www.pendrivelinux.com/universal-usb-installer-easy-as-1-2-3/ . I then restarted my laptop. I pressed Esc and chose "Boot Management", but there was no option for booting from USB. The only options I could see were boot from hard drive and boot from internal CD/DVD. What should I do?...

Figure 7-2: Pairs of similar questions from the test set which the RCNN model re-ranks better than the baseline models. The left column corresponds to the query question, and the right column corresponds to a question marked as similar in the 20 questions provided by BM25 based on the query question.

a clear improvement by fine-tuning the LSTM encoder (when comparing with the result without pre-training and fine-tuning).

7.3 Re-Ranking Analysis

RCNN vs. Other Models The RCNN model's performance gain over baseline methods and other neural network methods can be explained by the model's ability to better re-rank certain types of similar questions containing semantically similar words. Figure 7-2 shows

Questions	Similar Questions
Title: No Ubuntu partitions on installed dual boot system **Body:** I have both Ubuntu 12.04 and Windows 7 and dual booting. Ubuntu has 10GB of space and I'm running out of space. I want to increase this to 100GB, so I went to the Windows disk manager, it shows 100MB FAT16, 14GB recovery NTFS, 540GB OS NTFS. There are no partitions shown for Ubuntu/Linux in either the Windows disk manager or gparted. I used Windows disk manager to free up 100GB of space on my hard drive, is there a way to transfer my installed Ubuntu onto the new partition?...	**Title:** Issue with partitioned disk while trying to install ubuntu 13.10 along with Windows 8 **Body:** I have an installed windows 8 OS on my C drive and looking forward to install Ubuntu 13.10 as a dual boot alongside Windows 8 OS. I have the following partition set up, System reserved: 100 MB C: 60 GB (NTFS) D: 40 GB (NTFS) E: 50 GB (NTFS) Unallocated: 60 GB I am trying to install ubuntu on the unallocated spaceâĂę
Title: HTC X920d a.k.a HTC Butterfly **Body:** In the near future will you be building Unbuntu version for HTC X920d aka butterfly	**Title:** Can i install Ubuntu on HTC Butterfly **Body:** After using Ubuntu on PC, I also wanted to try on my "HTC Butterfly". But not dare to tried out what will happen or my data(Android OS) can be restore back or somethings else. So surfing on Google for information but only other HTC devices (like HTC Desire hd, HTC Sensation, etc) are show that can install Ubuntu. If I can, I also want to dual boot on my android phone like PC!...

Figure 7-3: Pairs of similar questions from the test set which the RCNN model re-ranks better when the title & body are both used as opposed to title only. The left column corresponds to the query question, and the right column corresponds to a question marked as similar in the 20 questions provided by BM25 based on the query question.

pairs of similar questions derived from the test set for which the RCNN model re-ranks much better than the other models. In each instance, the RCNN model is able to find similarities between semantically similar words such as 'installing' and 'installation' in the first pair and 'install' and 'installing' in the third pair, allowing the RCNN model to get relative improvements in performance over word-based models such as TF-IDF.

RCNN model using title only vs. title & body Related work in the community QA domain (see Chapter 2) has largely used question titles only due to their lack of noise and short lengths. The RCNN model however shows improved performance with the inclusion of the question bodies. Figure 7-3 shows pairs of questions which the RCNN model re-ranks better when the question bodies are included. In each instance, we can see that there are semantically similar words in the title of a question and body of another, such as

'partitions' and 'partitioned' in the first pair, or the presence of semantically similar words within question bodies, such as 'HTC' or 'butterfly' in the bodies of the second pair. The RCNN model's ability to model such semantic similarities within the noisy, complex question bodies is largely derived from the improvements outlined in Section 5.1.

Chapter 8

Conclusion

8.1 Contributions

RCNNs model 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. The resulting RCNNs model achieves a P@1 of 64.5% on the test set, outperforming the word matching-based method BM25 by over 10 percent points, and further outperforms the other neural encoder models and the baselines across all metrics.

Incorporation of Noisy Question Bodies As described in Chapter 1, questions in large-scale QA forums often contain meaningful information embedded in long, noisy question bodies that is not always directly referenced in the question titles. In our approach, we demonstrate the ability to effectively model the title and body representations together to create question representations that provide state-of-the-art performance when compared with standard baseline methods including CNNs and LSTMs.

Use of Large-Scale Unsupervised Data In using the entire corpus to pre-train the encoder part of the RCNN model, we illustrate the ability to use unsupervised data to improve performance. In most community QA sites including Stack Exchange, similar question annotations made by eligible users represent only a fraction of the true set of similar questions, making it difficult for traditional supervised approaches to perform well. In effectively

using the entire corpus in an unsupervised manner to improve performance, we provide a novel approach for enabling learning when dealing with insufficient or impartial supervised training data.

8.2 Future Work

Answer Selection In the question retrieval task addressed in this work, we develop a methodology for finding similar questions in large-scale QA forums. From the standpoint of a user using such a forum, finding a question already asked that's similar to the question they query can be helpful in finding relevant information quickly, but further value can potentially be provided to the user by suggesting a pre-existing answer to the question. The most obvious extension of our work along this avenue would be to return the most likely answer given to the similar question(s) identified, and there are other potential avenues of using the representation learning in our work, including incorporation of answers within the question representation framework.

Other Datasets This work focused solely on the AskUbuntu dataset from Stack Exchange, and the dataset choice was largely based on prior work with the dataset along with the lack of noise and relatively restricted domain of question topics in it. Extending this work to other growing QA forums, such as Quora, and QA forums in other cultural domains, such as Qatar Living, would provide additional grounds for testing the models and assessing their performance on a variety of other question types commonly found in online QA forums.

Bibliography

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

[Bär et al.2013] 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.

[Bengio et al.2003] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

[Bowman et al.2015] 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.*

[Chen and Manning2014] Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750.

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

[Collobert and Weston2008] R. Collobert and J. Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *International Conference on Machine Learning, ICML.*

[Collobert et al.2011] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

[Devlin et al.2014] Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *52nd Annual Meeting of the Association for Computational Linguistics.*

[dos Santos et al.2015] 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.

[Duan et al.2008] 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.

[Feng et al.2015] 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.*

[Gao et al.2014] 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.*

[Hinton et al.2012] 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.*

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Iyyer et al.2014] Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014. A neural network for factoid question answering over paragraphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 633–644, Doha, Qatar, October.

[Iyyer et al.2015] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. 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 1681–1691, Beijing, China, July.

[Jeon et al.2005] 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.

[Joachims2002] T. Joachims. 2002. Optimizing search engines using clickthrough data. In *ACM SIGKDD KDD.*

[Kalchbrenner and Blunsom2013] 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.

[Kalchbrenner et al.2014] 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.*

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

[Kim2014] Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014).*

[Kingma and Ba2014] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

[Küchler and Goller1996] Andreas Küchler and Christoph Goller. 1996. Inductive learning in symbolic domains using structure-driven recurrent neural networks. In *KI-96: Advances in Artificial Intelligence*, pages 183–197.

[Le and Zuidema2015] Phong Le and Willem Zuidema. 2015. Compositional distributional semantics with long short term memory. In *Proceedings of Joint Conference on Lexical and Computational Semantics (*SEM).*

[LeCun et al.1998] 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.

[Lei et al.2015] 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.

[Li and Manandhar2011] 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.

[Mikolov et al.2010] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

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

[Pollack1990] Jordan B Pollack. 1990. Recursive distributed representations. *Artificial Intelligence*, 46:77–105.

[Robertson and Zaragoza2009] Stephen Robertson and Hugo Zaragoza. 2009. *The probabilistic relevance framework: BM25 and beyond.* Now Publishers Inc.

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

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

[Severyn and Moschitti2015] Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *SIGIR.*

[Shen et al.2014] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the companion publication of the 23rd international conference on World wide web companion*, pages 373–374. International World Wide Web Conferences Steering Committee.

[Shen et al.2015] 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.*

[Socher et al.2011] Richard Socher, Cliff C. Lin, Andrew Y. Ng, and Christopher D. Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 26th International Conference on Machine Learning (ICML).*

[Socher et al.2013] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, October.

[Sutskever et al.2014] 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.

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

[Wang and Nyberg2015] Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. In *ACL*, July.

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

[Zhang and LeCun2015] Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *arXiv preprint arXiv:1502.01710.*

[Zhou et al.2013] 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.

[Zhou et al.2015] 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.