Transfer Learning for Detecting Question Similarity between Stack Overflow Posts

by

Victor Wing-chuen KWAN

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Master of Philosophy
in Computer Science and Engineering

May 2018, Hong Kong
Authorization

I hereby declare that I am the sole author of the thesis.

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Victor Wing-chuen KWAN

29 May 2018
Transfer Learning for Detecting Question Similarity between Stack Overflow Posts

by

Victor Wing-chuen KWAN

This is to certify that I have examined the above MPhil thesis and have found that it is complete and satisfactory in all respects, and that any and all revisions required by the thesis examination committee have been made.

__________________________

Dr. Sunghun KIM, Thesis Supervisor

__________________________

Prof. Albert Chi-Shing CHUNG, Acting Head of Department

Department of Computer Science and Engineering
29 May 2018
I would like to thank my advisor Dr. Sunghun Kim, who, above anything, taught me to always consider the bigger picture whether in life or in research. I would also like to thank Dr. Yangqiu Song, who suggested the idea for this thesis and provided immense guidance along the way, as well as Dr. Kam-yin Wu, who offered advice on revising my introduction.

I am especially grateful for the support of my labmates, Mijung, Russell, Ken, and Fuxiang, who guided me through the graduate experience over many cups of coffee. I am also thankful for my close friends, Gabe, Richard, and Arlene, who supported me throughout the process. Finally, I would like to express my gratitude to my family and my girlfriend, Cathy, who were ever present, generous and caring throughout my graduate and thesis experience.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title Page</td>
<td>i</td>
</tr>
<tr>
<td>Authorization Page</td>
<td>ii</td>
</tr>
<tr>
<td>Signature Page</td>
<td>iii</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>iv</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>v</td>
</tr>
<tr>
<td>List of Figures</td>
<td>vii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>viii</td>
</tr>
<tr>
<td>Abstract</td>
<td>ix</td>
</tr>
<tr>
<td>Chapter 1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Chapter 2 Background</td>
<td>3</td>
</tr>
<tr>
<td>2.1 SO Question Similarity</td>
<td>3</td>
</tr>
<tr>
<td>2.2 Textual Matching</td>
<td>4</td>
</tr>
<tr>
<td>2.3 Transfer Learning</td>
<td>5</td>
</tr>
<tr>
<td>Chapter 3 Methodology</td>
<td>7</td>
</tr>
<tr>
<td>3.1 Dataset Construction</td>
<td>7</td>
</tr>
<tr>
<td>3.1.1 SO Question Pairs</td>
<td>7</td>
</tr>
<tr>
<td>3.1.2 Quora Question Pairs</td>
<td>9</td>
</tr>
<tr>
<td>3.2 Transfer Learning</td>
<td>10</td>
</tr>
<tr>
<td>3.3 Model Components</td>
<td>10</td>
</tr>
<tr>
<td>3.3.1 Creating Sentence Representations</td>
<td>10</td>
</tr>
<tr>
<td>3.3.2 Comparing Sentence Representations</td>
<td>14</td>
</tr>
<tr>
<td>Chapter 4 Experiments</td>
<td>16</td>
</tr>
<tr>
<td>4.1 Model Configuration and Training</td>
<td>16</td>
</tr>
<tr>
<td>4.2 Hyperparameter Tuning</td>
<td>16</td>
</tr>
<tr>
<td>4.3 Ablation Study</td>
<td>17</td>
</tr>
<tr>
<td>4.4 Unfreezing Model Components</td>
<td>18</td>
</tr>
<tr>
<td>Chapter 5 Results and Discussion</td>
<td>19</td>
</tr>
<tr>
<td>5.1 Under- and Overfitting Models</td>
<td>19</td>
</tr>
<tr>
<td>5.1.1 Significance Evaluation</td>
<td>20</td>
</tr>
<tr>
<td>5.1.2 Pretraining as a Regularizing Mechanism</td>
<td>21</td>
</tr>
<tr>
<td>5.2 The Efficacy of the Model Components</td>
<td>23</td>
</tr>
<tr>
<td>5.2.1 Effect of Character-Based Embeddings</td>
<td>23</td>
</tr>
<tr>
<td>5.2.2 Effect of Tag Information</td>
<td>24</td>
</tr>
<tr>
<td>5.2.3 Effect of Attention Mechanism</td>
<td>25</td>
</tr>
<tr>
<td>5.2.4 Effect of Fully-Connected Classifier</td>
<td>25</td>
</tr>
<tr>
<td>5.3 Transferable Knowledge between the Quora and SO Domains</td>
<td>26</td>
</tr>
<tr>
<td>5.3.1 Transferable Sentence Representations</td>
<td>27</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>An overview of our filtering process for building the set of non-duplicate examples.</td>
<td>8</td>
</tr>
<tr>
<td>3.2</td>
<td>An overview of our model.</td>
<td>11</td>
</tr>
<tr>
<td>3.3</td>
<td>An overview of the character-based embeddings.</td>
<td>12</td>
</tr>
<tr>
<td>5.1</td>
<td>Comparison of non-pretrained and pretrained model accuracy.</td>
<td>20</td>
</tr>
<tr>
<td>5.2</td>
<td>Comparison of attention weights produced by the pretrained and non-pretrained models, and the non-pretrained model.</td>
<td>21</td>
</tr>
<tr>
<td>5.3</td>
<td>Learning curves for the non-pretrained and pretrained models.</td>
<td>22</td>
</tr>
<tr>
<td>5.4</td>
<td>Comparison of attention weights between the full model and the model with ablated character-based embeddings.</td>
<td>24</td>
</tr>
<tr>
<td>5.5</td>
<td>Attention weights for the classifier-ablated model.</td>
<td>25</td>
</tr>
<tr>
<td>5.6</td>
<td>Comparison of attention weights between the Quora-trained model and the fully-tuned model.</td>
<td>27</td>
</tr>
<tr>
<td>5.7</td>
<td>Pearson and Spearman correlation coefficients between the attention weights generated by Quora-trained and fully-tuned models.</td>
<td>27</td>
</tr>
<tr>
<td>5.8</td>
<td>Histograms depicting the LCSRs for positive question pairs in the Quora and SO datasets, as well as those correctly identified in the SO dataset by the Quora-trained model.</td>
<td>29</td>
</tr>
</tbody>
</table>
LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>SO dataset statistics.</td>
<td>8</td>
</tr>
<tr>
<td>3.2</td>
<td>Quora dataset statistics.</td>
<td>9</td>
</tr>
<tr>
<td>5.1</td>
<td>Hyperparameter tuning results.</td>
<td>19</td>
</tr>
<tr>
<td>5.2</td>
<td>Ablation study results.</td>
<td>23</td>
</tr>
<tr>
<td>5.3</td>
<td>Model performance (Accuracy, Macro-Average F1 Score) with frozen layers.</td>
<td>26</td>
</tr>
<tr>
<td>5.4</td>
<td>Model performance (Precision, Recall, and F1 Score) with frozen layers.</td>
<td>26</td>
</tr>
</tbody>
</table>
Transfer Learning for Detecting Question Similarity between Stack Overflow Posts

by Victor Wing-chuen KWAN

Department of Computer Science and Engineering
The Hong Kong University of Science and Technology

Abstract

Developers often come to Stack Overflow to seek help about their programming problems. However, the technicality of the content makes the task of relevant question retrieval especially difficult. Drawing from a wide pool of natural language processing techniques, we devise a deep neural network model for question similarity that attempts to learn the semantic relationships between Stack Overflow questions using the titles and tags of posts. We additionally build around the idea of pretraining against a Quora dataset for added robustness against the noisy Stack Overflow dataset. Our contributions include an effective model for question similarity that leverages transfer learning for added robustness; a study into how the model components contribute towards model performance; and a study into the transferability of knowledge between the Quora and Stack Overflow domains.
CHAPTER 1
INTRODUCTION

Developers often come to Stack Overflow (SO) to seek help about their programming problems. SO is among the most popular Community Question Answering (cQA) websites, reporting over nine million visits per day and hosting over 16 million questions [1]. Given this volume of information, effective question retrieval is essential to a good user experience. However, the technicality of the content on SO makes question retrieval especially difficult.

One consequence of this large number of varied questions is that many questions posted onto SO are in fact duplicates. Duplicates are questions that effectively paraphrase one another. Ahasanuzzaman et al. [2] report that many such questions stem from lack of problem knowledge, non-matching titles, and lack of knowledge of relevant terminology and buzzwords. This results in similar problems being expressed in very diverse ways. For example, Question ID (QID) 761376 and 3374120 are marked as duplicates. The former asks “Check connection is active in ASP.NET,” while the latter asks “How to check if user is still present?” Only by drawing from the context (e.g. tags) and the nuance behind the questions (i.e. a user who is present will have an active connection) can we infer that these questions are actually asking about the same thing.

More broadly, questions on SO are very susceptible to these differences in nuance. There are many ways to present relevant keywords and names (e.g. “IllegalArgumentException” vs. “Illegal Argument Exception”), and as we see from the previous example, there are numerous degrees of specificity with which questions may be asked. It follows that these question patterns require more than just keyword matching; we need a retrieval system capable of encoding semantic information to identify similar questions [3]. In SemEval-2017 [4], this problem has been formalized as the “Question-Question Similarity” task. Coupled with the notion of “Question-Comment Similarity,” which involves ranking the best answers to a question, we can think of these two steps as the components of an intelligent question-answering system.

To address the question similarity task, we draw from a wide pool of Natural Language Processing (NLP) techniques. Our model attempts to create salient question representations by taking advantage of developments in attention-based sentence representations [5]. In the context of our question similarity task, we consider the question titles to be sentences. These attention mechanisms allow us to holistically create sentence representations based on learned notions of word importance. We additionally leverage subword information [6] to pry into the internal structures of the technical SO vocabulary. While there have been many attempts [7]–[9] at similarly tackling the question similarity task, our work differentiates itself in that the SO dataset is noisy and necessitates building around the idea of transfer learning.
One variant of transfer learning \[10\] is the idea that representations accrued from a model performing some task are transferable to other tasks. In the context of NLP, this has been most often realized in the form of dense word representations \[11\]. Our model is built around the assumption that duplicate signals on SO are noisy: there are no apparent techniques to guarantee the quality of our non-duplicate examples. To address this issue, we use the aforementioned NLP techniques to learn robust, sentence representations against the Quora Question Pairs dataset \[12\]. We then transfer this knowledge towards the task of identifying duplicate SO questions. Similar to previous works examining transfer learning in the context of neural networks \[13\]–\[15\], we perform experiments that explore the transfer of knowledge between the Quora and SO domains.

Finally, this research is most closely related to existing attempts at addressing duplicate questions on SO \[2\], \[16\], \[17\]. These techniques rely on a variety of engineered features based on question title, tags, and body to identify whether pairs of questions are duplicates. We differentiate from these works, first, by learning task-specific intuitions about semantic similarity through an end-to-end model, as opposed to performing separate forms of unsupervised training (e.g. Doc2Vec \[18\]) and relying on external databases (e.g. WordNet \[19\]). Secondly, we position our work from the perspective of a user visiting SO in search of relevant questions. As such, we focus only on the question title and tags, omitting the question body.

In summary, the contributions of this research are as follows:

1. A model for question similarity built with sentence representation robustness in mind. Our best pretrained model achieves a test F1 score of 0.814, which is a 0.011 improvement over its non-pretrained counterpart.
2. An ablation study into how our model components benefit the question similarity task.
3. A study into how model size and layers frozen may affect the transfer learning process, with a focus on the differences between the Quora and SO datasets.

The remainder of this thesis is structured as follows. In Chapter\[2\], we discuss how our work relates to existing approaches in SO question similarity, textual similarity, and transfer learning. We follow this by introducing our methodology and experiments in Chapters \[3\] and \[4\] respectively. We then discuss our findings in Chapter \[5\]. Finally, we conclude and offer future directions to expand our work in Chapter \[6\].
CHAPTER 2

BACKGROUND

SemEval-2017 [4] presents an interesting perspective on the question-answering task. The overall objective of question-answering can be broken down into two steps: “Question-Question Similarity,” followed by “Question-Comment Similarity.” More concretely, given a question, if we can first identify a similar question, then select the most relevant answer to this identified question, we effectively create a question-answering system.

In general, the task of question-answering on cQA websites is very difficult. Questions found on such websites are often informal, unfocused, and exhibit language switches [3]. Ahasanuzzaman et al. [2] characterize SO questions quite similarly, reporting that many duplicate questions stem from lack of problem knowledge, non-matching titles, and lack of knowledge of relevant terminology and buzzwords. More generally, we can attribute the large number of duplicate questions to the failure of existing question retrieval systems. Specifically, to target the question similarity task, we need retrieval systems capable of encoding semantic relationships between questions [3]. Our research attempts to fill this gap in the context of SO questions.

2.1 SO Question Similarity

Existing works addressing question similarity on SO have been largely dominated by feature-based approaches. Focusing on question retrieval, Xu et al. [20] devise a system that retrieves relevant questions and extractively summarizes salient answers. Their system retrieves questions based on a combination of word embeddings and Inverse Document Frequency (IDF) metrics, and selects answer paragraphs based on a number of hand-crafted query, paragraph content, and user-oriented features.

Duplicate detection systems such as DupPredictor [16] and Dupe [2] similarly rely on feature engineering to determine question similarity. DupDetector [17] can be considered a successor to these previous duplicate detection approaches, applying similarity, relevance, and association features to classify whether a pair of questions are duplicates.
We differentiate from these works in that our model produces comparatively salient question representations through end-to-end training. Specifically, these approaches rely on unsupervised methods (e.g. Vector Space Models, Doc2Vec [18]) to encode their question information. We believe that these unsupervised methods can be effective in the context of these works because of the large amount of information that they take into account: question title, tags, and body. On the other hand, our work is positioned from a user-facing perspective, employing only the question title and tags to determine question similarity. As such, we need a feedback loop that continually refines our representations to draw out the most salient features from this limited information. This motivates our use of Deep Neural Networks (DNNs) that can facilitate the refinement of our question representations.

Xu et al. [21] make a preliminary effort at applying DNNs towards classifying the semantic relatedness between SO posts. In particular, their model employs a Convolutional Neural Network (CNN) that takes pairs of questions and classifies these pairs as duplicate, directly-linked, indirectly-linked, or isolated. We differentiate from their work by employing a more sophisticated model capable of tackling our comparatively diverse and less informative dataset. With regards to diversity, in contrast to their dataset of 8,000 Java-tagged questions, our dataset consists of 277,075 questions without any tag restrictions. With regards to informativeness, on top of question title and body, Xu et al. consider answer bodies, whereas we only focus on question title and tags. As such, our dataset necessitates the use of more complex model components, e.g. character-based embeddings and attention mechanisms, to maximize the information we can extract from our input data.

2.2 Textual Matching

Many works approaching textual (e.g. document, sentence, question) matching tasks follow the Siamese network architecture [22]. This architecture breaks the matching process into two steps. The first step, creating vector representations, has been explored through a variety of techniques. For example, dos Santos et al. [7] apply CNNs over Bags of Words (BoW) to create question representations. Lei et al. [8] create question representations by training an encoder-decoder network on the auxiliary task of predicting a question title based on its title and body. The second step, classifying the vector representations, has been explored through the Manhattan distance [23] and Multi-Perspective Matching [9].

Other architectures have also been proposed for textual matching. For example, Parikh et al. [24] consider the building of sentence-level representations unnecessary for Natural Language Inference (NLI), instead performing parallelizable pairwise comparisons over the input texts. Tomar et al. [25] successfully adapt this approach to the question similarity task.
Between the two architectures, we build a model following the Siamese architecture to address the limitations of our dataset. Specifically, the SO dataset is noisy, which we address by building a model that learns robust sentence-level representations. This way, we can transfer learned representations from a semantically similar task to the SO domain. Rather than focusing on the comparative aspects of our model, as in Wang et al. [9] and Parikh et al. [24], our work emphasizes building flexible and transferable sentence representations between the Quora and SO domains.

We achieve these robust representations by drawing from two interesting threads of NLP research. The first is to do with creating representations of higher-level abstractions: rather than simply using word-level representations, we can utilize attention mechanisms to create sentence-level representations [5]. The other thread is to do with using subword information [6] to infer relationships between words. One realization of this idea is through the use of CNNs over character embeddings [26]. Altogether, our model applies these techniques within the Siamese architecture to achieve robust sentence representations.

2.3 Transfer Learning

Pretraining is a form of transfer learning that has been investigated through both unsupervised [27], [28] and supervised [13], [14] techniques. Transferable sentence representations have likewise been explored along this unsupervised [18] and supervised [29] dichotomy. Conneau et al. [29] create universal sentence representations by performing supervised training against NLI data, which have been shown to generalize well to other NLP tasks. In our case, however, we leverage the datasets of semantically similar tasks to perform our transfer learning. Specifically, we are interested in transferring sentence representations from a less noisy source domain to facilitate better performance against our noisy target domain.

As such, our work is more closely related to that of Tomar et al. [25], which pretrains the aforementioned Parikh et al. model against a noisy source domain. Wiese et al. [30] similarly attempt knowledge transfer to a technical domain, applying SQuAD [31] as a source domain towards question-answering in the target biomedical domain.
There are a number of studies exploring transfer learning in the context of neural networks, from which we draw two interesting relationships that we would like to investigate in our work. The first is co-adaptation \cite{13}, in which neurons in neighboring layers may become dependent on each other during pretraining, resulting in neurons being “lost” when one layer is unfrozen while the other is not during tuning. We explore this relationship by incrementally freezing our model components and examining the effect of freezing layers on model performance. Another interesting relationship, as discussed in Mou et al. \cite{14}, is the idea that pretraining can set a “catchment basin” for the error surface. In other words, transferring sentence representations from the Quora domain can act as a form of regularization. Finally, we highlight Chung et al. \cite{15} as a similar work that explores the effects of layer freezing and dataset size on the efficacy of supervised and unsupervised transfer learning on the question-answering task.
CHAPTER 3

METHODOLOGY

3.1 Dataset Construction

In this section, we discuss the construction of our SO and Quora datasets. General statistics for the two datasets are presented in Tables 3.1 and 3.2.

3.1.1 SO Question Pairs

To assemble the SO dataset, we use the Stack Exchange Data Explorer [32]. The Data Explorer has a limit of 50,000 rows returned per query, so we iteratively fetch examples using the OFFSET N ROWS clause until we retrieve the desired number of rows. Because our data is fetched across a number of queries, we order by QID rather than RAND() to avoid fetching duplicate results.

We model our duplicate examples after the dataset used in Ahasanuzzaman et al. [2]. This dataset draws question pairs from the Mining Software Repositories (MSR) 2015 data dump, which contains a snapshot of the SO database between August 2008 and September 2014. A question is considered a duplicate of another if the questions are related using the duplicate link type in the PostLinks table, and the duplicate question has been closed. We exhaust all duplicate question pairs in this time range, fetching the QID, title, and tags for each post. Tags are words or phrases that describe the topic of the question [33]. In total, we retrieve 137,793 question pairs for use as duplicate examples.

In line with the Quora Question Pairs dataset [12], we select pairs of related questions as our negative examples. This way, we construct a dataset with negative examples that are not trivially classifiable. As in Xu et al. [21], we use the linking between posts as a proxy for question relatedness. Posts are linked when one post is referenced in the question body, answers, or comments of another post [34]. Using the Data Explorer, we identify linked question pairs using the linked link type in the PostLinks table.

1We note that our approach is somewhat different from Hoogeveen et al. [35], who supplement their dataset of duplicates on non-SO Stack Exchange forums using the built-in related post facilities. We decide against this approach because our large volume of duplicate examples would necessitate a correspondingly large number of API calls to retrieve sufficient non-duplicate examples.
### Table 3.1: SO dataset statistics.

<table>
<thead>
<tr>
<th></th>
<th>Train</th>
<th>Dev</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duplicate Examples</td>
<td>127,793</td>
<td>5,000</td>
<td>5,000</td>
</tr>
<tr>
<td>Non-duplicate Examples</td>
<td>129,282</td>
<td>5,000</td>
<td>5,000</td>
</tr>
<tr>
<td>Average Tokens</td>
<td>9.6</td>
<td>9.6</td>
<td>9.6</td>
</tr>
<tr>
<td>Average Tags</td>
<td>3.0</td>
<td>3.0</td>
<td>3.0</td>
</tr>
<tr>
<td>Vocabulary Size</td>
<td>52,841</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tagset Size</td>
<td>17,202</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 3.1:** An overview of our filtering process for building the set of non-duplicate examples. The gray circles are connected components in the duplicate question graph, and (1-4) are examples of linked question pairs. We include pairs (1) and (3) because they are relevant to our duplicate pairs and the questions are not transitively duplicates of one another. We exclude (2) because it is irrelevant to our set of duplicate pairs, as well as (4) because the questions are transitively duplicates of one another.

We then use graph-based techniques to ensure that (1) the linked question pairs are relevant to the set of duplicate pairs, and (2) the pairs are not transitively duplicates of one another. We adopt the former constraint to ensure that the model does not simply memorize questions in the duplicate examples. Our filtering method utilizes the NetworkX package to build an undirected graph between pairs of duplicate questions. We then extract the connected components of the graph, which can be considered clusters of duplicate questions. Linked question pairs are then filtered such that at least one question in each pair can be found in a cluster, and that both questions cannot be members of the same cluster. Despite this filtering, we cannot guarantee that the duplicate clusters are exhaustive, inevitably resulting in some duplicate question pairs being mislabeled as non-duplicate. As such, the negative examples in our dataset may be noisy. An illustration of our filtering process is presented in Figure 3.1.

---

2 This requirement for non-duplicate pairs is quite similar to that in Silva et al. [37], where each of their negative examples consists of a question selected from the set of duplicate questions and another randomly selected question.
Altogether, we iteratively retrieve 500,000 linked question pairs from the same time period as before, then filter these down to 139,282 non-duplicate examples, which is approximately the same as the number of duplicate examples. We additionally find that 34,225 question pairs are transitively duplicates. The remaining question pairs are discarded as irrelevant to our set of duplicate examples. Although the question pairs are not exhaustive for this time period, we maintain that the relatively large sample size should provide sufficient diversity to the dataset. More crucially, even though sorting by QID may lend to a concentration of earlier posts, we argue that the patterns with which questions are asked should not vary significantly over time. As such, we believe this sample to be a reasonable source of negative examples.

After filtering the linked question pairs, we combine the duplicate and non-duplicate examples and tokenize the question titles using spaCy \cite{spacy}. Given that question titles are already reasonably short, we opt to preserve stopwords. We sample 5,000 positive and 5,000 negative examples for test, 5,000 positive and 5,000 negative examples for dev, and utilize the remaining examples for training.

### Quora Question Pairs

Quora is a popular cQA website that targets a general audience. We use the Quora Question Pairs dataset \cite{quora} as our source domain for pretraining because it relates to a semantically similar task (i.e. question similarity) and can expose our model to a wide variety of question patterns. This dataset consists of duplicate and non-duplicate question titles, but does not provide additional tag metadata.

We use the same splits as Wang et al. \cite{wang1} to create our Quora dataset. However, because the splits published by Wang et al. are already tokenized using the Stanford CoreNLP \cite{stanford} tokenizer, we recover the original questions by matching the QIDs of the splits with the original Quora-published dataset. We then carry out tokenization using spaCy as before. Altogether, the Quora dataset consists of 149,306 positive examples and 255,042 negative examples. The splits consist of 5,000 positive and 5,000 negative examples for test, 5,000 positive and 5,000 negative examples for dev, and the remaining examples for training.
3.2 Transfer Learning

We conduct transfer learning through pretraining. Specifically, we first pretrain our model on the Quora dataset. We then re-initialize the weight matrices of the fully-connected classifier, noting that the lack of tags in the Quora dataset may result in the pretrained classifier ignoring tag information. Finally, we tune the model on the SO dataset. Our aim is for the model to learn a foundation of question patterns through the Quora dataset. This knowledge can then be transferred, in the form of sentence representations, and provide a layer of robustness when tuning against noisy SO dataset.

3.3 Model Components

Our model follows the Siamese network architecture [22]. In the context of sentence similarity, the Siamese architecture divides the comparison task into two subtasks: (1) creating sentence representations, and (2) comparing the two representations using some form of distance function. For our task, we consider the question titles to be the input sentences. We employ the Siamese architecture to develop transferable and task-specific sentence representations.

The base of our model, a Siamese Gated Recurrent Unit neural network, is adapted from Homma et al. [40]. This model connects Gated Recurrent Units (GRUs) [41] to a fully-connected classifier to detect question similarity. On top of this model, we include character-based embeddings, an attention mechanism, and tag information. We discuss our motivation behind and implementation of these components in the following subsections. An overview of our model is presented in Figure 3.2.

3.3.1 Creating Sentence Representations

The input to our GRUs consists of a concatenation of pretrained word embeddings and character-based embeddings.

Word Embeddings

The pretrained word embeddings utilize 300d GloVe vectors [42] trained on the Wikipedia 2014 and Gigaword 5 corpora. Only words found in our Quora and SO training corpora are indexed by our word embedding matrix.
Figure 3.2: An overview of our model. Concatenated word and character-based embeddings are passed as input to the bi-directional GRUs. Each timestep of the GRU is then scored against the trainable context vector $u_w$. These scores are used to compute a weighted sum of the timesteps, which is subsequently used as the sentence representation. The sentence representations, their absolute differences, their element-wise products, and their tags are then concatenated and classified using two fully-connected layers.
Character-Based Embeddings

The intuition behind using character-based embeddings is that many words in the SO dataset may be Out-Of-Vocabulary (OOV) with respect to the GloVe vectors. However, we notice that many identifiers are constructed as compound words, e.g. “UIToolbar” can be decomposed into “UI,” “Tool,” and “bar.” Character-based embeddings allow us to leverage this form of subword information. Furthermore, they are more robust than word embeddings trained on the SO corpus to the many permutations with which identifiers may be presented. For example, “IllegalArgumentException” can be divided into “IllegalArgument” and “exception,” where “IllegalArgument” may still be OOV with respect to the SO corpus.

Figure 3.3: An overview of the character-based embeddings. We convolute over the character embeddings with kernels of sizes 3, 4, and 5. These are then max-pooled, passed through the ReLU nonlinearity, and concatenated to form the character-based embeddings.
Our character-based embeddings are informed by the techniques in Seo et al. [26] and follow similar hyperparameters to Lee et al. [43]. An overview of the character-based embeddings is presented in Figure 3.3. We first split words into their constituent characters and project these into 8d character embeddings. We then perform 1d convolutions over these character embeddings using kernels of sizes 3, 4, and 5. Each convolving kernel produces an output with 50 channels. We then max-pool the convolution output and apply the ReLU activation function, resulting in a 50d vector. These are subsequently concatenated to form 150d character-based embeddings. Only characters found in our Quora and SO training corpora are indexed by our character embeddings.

GRUs

We employ bi-directional GRUs to create our sentence representations. Specifically, we use the concatenation of hidden states from the forward and backward GRUs as sentence representations centered at each timestep, passing these representations to the attention mechanism to weigh their importance. As such, our sentence representations are double the size of the GRU hidden state.

Attention Mechanism

The advantage of adding an attention mechanism is two-fold. First, we avoid overburdening the GRU with the task of creating an entirely representative sentence representation at the last timestep of computation; instead, we train the attention mechanism to determine the importance of each timestep. Second, the attention mechanism allows us to more easily interpret the model, which can aid our understanding of the pretraining process.

Our attention mechanism draws from Yang et al. [5] and can be described as follows:

\[
\begin{align*}
    u_t &= \text{ReLU}(W_u h_t + b_u) \\
    \alpha_t &= \frac{\exp(u_t^T u_w + b_c)}{\sum_i \exp(u_i^T u_w + b_c)} \\
    v &= \sum_t \alpha_t h_t
\end{align*}
\]

where \( h_t \) denotes timestep \( t \) of the GRU output; \( u_w \) denotes the trainable context vector; and \( b_c \) denotes the trainable bias term; and \( v \) denotes the sentence representation. We subsequently refer to the output size of \( W_u \) as the “attention size.”
We construct our sentence representations by taking the weighted sum of the timesteps with respect to the scores assigned by the attention mechanism. These scores are determined by first passing the GRU output through a fully-connected layer, followed by taking the dot product of each timestep with the context vector.

3.3.2 Comparing Sentence Representations

Tag Metadata

As part of our input to the fully-connected classifier, we include the tag metadata for each question pair. We believe that tag information provides useful contextual information and is appropriate for our user-oriented perspective on the question similarity task, i.e. a user can be reasonably expected to provide tag information when attempting to retrieve a question.

Due to the large number of tags on SO, we compress each tag into a 16d tag embedding. We fix our tag input at the SO maximum of five tags per question, using [pad] tokens where there are fewer than five tags. Our tag input for Quora questions consists entirely of [pad] tokens given the lack of tags in the Quora dataset.

Fully-Connected Classifier

We compare the sentence representations using two fully-connected layers. The input to our classifier draws from Mou et al. [44]:

\[ m = [v_1; v_2; \text{abs}(v_1 - v_2); v_1 \odot v_2; t_1; t_2] \]
\[ h = \text{ReLU}(W_h m + b_h) \]
\[ o = \text{sigmoid}(W_o h + b_o) \]

where \( v_1 \) and \( v_2 \) denote the representations of the first and second questions respectively; \( \text{abs}(\cdot) \) denotes the element-wise absolute value; \( \odot \) denotes the element-wise product; and \( t_1 \) and \( t_2 \) denote the concatenated tag embeddings for the first and second questions respectively. \( \cdot \) denotes the concatenation operation. \( m \) is the input to the classifier, \( h \) is the output of the hidden layer, and \( o \) is the classification output.

\(^3\)We decide against concatenating tags to the end of our question titles as this may be disruptive to the learned question patterns.
While the Mou et al. formulation directly uses the difference between a hypothesis and a premise, we use the absolute difference between our question representations to make the difference symmetric. This adaptation is inspired by Homma et al. [40], although we use the element-wise absolute difference rather than squared difference in our model, noting that the absolute difference is more robust to extreme values.

The size of the first fully-connected layer output is fixed at four times the GRU hidden size. The second fully-connected layer produces a single output, which is scaled between $[0, 1]$ using the sigmoid function. This serves as the classification output, which is then subject to a threshold of $>= 0.5$ to be considered a positive example.
CHAPTER 4

EXPERIMENTS

4.1 Model Configuration and Training

Our model is implemented using PyTorch [45] and the experiments are conducted on a single Nvidia GeForce GTX 1070 GPU with 8GB GDDR5 RAM. Our word embeddings are initialized and frozen with GloVe vectors [42], and are subject to a max $l_2$ norm of 1.0. We use He normal initialization [46] to initialize the other components unless their weights are to be copied from a pretrained model.

For training, we use the Adam optimizer [47] with default settings. We fix the learning rate at $lr=1e^{-3}$ for both training and evaluation as we find that, unlike in Bowman et al. [48], maintaining the learning rate does not result in the destruction of knowledge accumulated from pretraining. When training our model with the fully-connected classifier, we use a binary cross-entropy loss function. For one of our ablation study variants, we replace the fully-connected classifier with the Manhattan distance; in this case, we use the mean-squared error loss function as performed by Mueller and Thyagarajan [23].

With regards to regularization mechanisms, we find that gradient clipping and weight regularization are unnecessary for the model to converge. However, we apply dropout with $p=0.2$, $p=0.1$, and $p=0.5$ for word embeddings, character-based embeddings, and the hidden layer of the fully-connected classifier respectively. Where the outputs are fixed size, i.e. the character-based embeddings and the hidden layer of the fully-connected classifier, we additionally perform batch normalization [49] to speed up the convergence of our model.

We fix the size of the character input for each word at 10 characters, which sufficiently covers more than the 90th percentile\footnote{The word at the 90th percentile has 8 characters.} of word lengths in our training corpora. For both training and evaluating our models, we use a batch size of 64 examples.

4.2 Hyperparameter Tuning

This experiment aims to explore the regularizing effects of pretraining [14] with respect to model size. To do so, we consider the non-pretrained and pretrained performance of our model over a variety of GRU and attention sizes. We fix the GRU and attention sizes to be equal, selecting sizes from 50, 100, 150, and 200 output neurons.
For our pretrained models, we pretrain against the Quora dataset for 50 epochs then tune against the SO dataset for 50 epochs. When tuning the model, we begin by re-initializing the weight matrices of our fully-connected classifier. We then initialize the other components with weights from the Quora-trained model with highest Quora dev accuracy. After performing this initialization, we freeze the word embedding matrices with GloVe vectors.

For our non-pretrained models, we train each model for 50 epochs. For both pretrained and non-pretrained models, we select the model with highest dev accuracy for evaluation.

4.3 Ablation Study

We evaluate the efficacy of our model components by performing an ablation study. We introduce four variants of our model:

1. The first removes character-based embeddings from the model;
2. The second removes the attention mechanism from the model. Instead of relying on all the timesteps to inform the sentence representation, we use a uni-directional GRU and utilize its final hidden state as the sentence representation;
3. The third removes the fully-connected classifier from the model. Instead, we use the Manhattan distance as in Mueller and Thyagarajan [23]:

   \[ o = \exp(-||v_1 - v_2||_1) \]

   where \( v_1 \) and \( v_2 \) are the sentence representations, and \( o \) is the classification output, which is then subject to a threshold of \( > 0.5 \) to be considered a positive example.

   We note that this variant omits tag information, motivating the final variant.
4. The fourth suppresses tag information from the fully-connected classifier. This allows us to fairly compare performance with and without the fully-connected classifier.

We fix our model size based on the best tuned model from the first experiment. Each of these variants are pretrained against the Quora dataset then tuned against the SO dataset, using the same initialization scheme as before. Both pretraining and tuning are conducted over 25 epochs, from which we select the model with highest dev accuracy for evaluation.
4.4 Unfreezing Model Components

The final experiment aims to explore the relationship between Quora and SO datasets through the lens of transferable knowledge. Following the experiments in Yosinski et al. [13] and Mou et al. [14], we do so by incrementally unfreezing layers with the following variants:

1. Unfreezing the classifier and tag embeddings;
2. Unfreezing the classifier, tag embeddings, and attention mechanism;
3. Unfreezing the classifier, tag embeddings, attention mechanism, and GRU.

We copy our model weights from the Quora-trained model that acted as the base for the best tuned model in the first experiment. We then initialize the model using the same initialization scheme as before. Tuning is conducted over 25 epochs, from which we select the model with highest dev accuracy for evaluation.
CHAPTER 5

RESULTS AND DISCUSSION

5.1 Under- and Overfitting Models

As a byproduct of hyperparameter tuning, we investigate the effect of GRU hidden and attention sizes on model performance. From Figure 5.1a, we see that pretraining generally allows us to converge with higher dev accuracy. This suggests that there is some form of knowledge transfer between the Quora and SO domains.

However, Figure 5.1b reveals that we still need to be careful about the generalizability of our models. For example, we find that the size=200 model likely suffers from overfitting, performing identically to its non-pretrained counterpart on the test set. Worse yet, we see that the size=50 model suffers from underfitting, exhibiting diminished performance compared to its non-pretrained counterpart. In this case, the model is possibly confused by the pretraining.

We also notice an interesting phenomenon where the best performing pretrained model (size=150) does not necessarily have the best performing non-pretrained counterpart (size=200). To this end, we encourage the careful tuning of hyperparameters when carrying out pretraining.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Dev Accuracy</th>
<th>Test Accuracy</th>
<th>Test F1 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>50, NPT</td>
<td>0.805</td>
<td>0.796</td>
<td>0.797</td>
</tr>
<tr>
<td>50, PT</td>
<td>0.808</td>
<td>0.795</td>
<td>0.795</td>
</tr>
<tr>
<td>100, NPT</td>
<td>0.814</td>
<td>0.803</td>
<td>0.803</td>
</tr>
<tr>
<td>100, PT</td>
<td>0.819</td>
<td>0.807</td>
<td>0.807</td>
</tr>
<tr>
<td>150, NPT</td>
<td>0.814</td>
<td>0.801</td>
<td>0.803</td>
</tr>
<tr>
<td>150, PT</td>
<td><strong>0.824</strong></td>
<td><strong>0.814</strong></td>
<td><strong>0.814</strong></td>
</tr>
<tr>
<td>200, NPT</td>
<td>0.819</td>
<td>0.809</td>
<td>0.809</td>
</tr>
<tr>
<td>200, PT</td>
<td>0.821</td>
<td>0.809</td>
<td>0.809</td>
</tr>
</tbody>
</table>

Table 5.1: Hyperparameter tuning results. NPT denotes “non-pretrained,” while PT denotes “pretrained.” Our selected model is **bolded**.
Figure 5.1: Comparison of non-pretrained and pretrained model accuracy.

5.1.1 Significance Evaluation

To evaluate the statistical significance of our results, we apply the continuity-corrected version of McNemar’s test on the test set predictions produced by the non-pretrained and pretrained size=150 models. When applying McNemar’s test, the null hypothesis states that there is marginal homogeneity between the two models. After computing the McNemar test statistic, we reject the null hypothesis with $P < 0.001$. Together with the improved F1 score, we demonstrate that transferring between the Quora and SO domains is indeed beneficial towards the task of identifying similar questions.
5.1.2 Pretraining as a Regularizing Mechanism

Following Mou et al.’s [14] claim that pretraining sets a “catchment basin” for the error surface, we explore the idea of pretraining as a regularizing mechanism. We begin by considering the attention weights produced by our models. In Figure 5.2a, we surprisingly find that the non-pretrained size=200 model focuses exclusively on the last timestep of the backwards GRU. This would suggest that the model considers the attention mechanism redundant, relying only on the GRU for its sentence representations.\footnote{We note that Conneau et al. [29] observe similar but less exaggerated behavior when max-pooling over the outputs of a non-pretrained LSTM. In their appendix, they show that max-pooling generally focuses on the initial and final timesteps.} In contrast, from Figure 5.2b, we see that the pretrained size=200 model shares the work of creating sentence representations between the GRU and the attention mechanism. We confirm that this is generally the case by considering the non-pretrained size=150 model in Figure 5.2c. We can think of the pretrained size=200 model behavior as a result of regularization: rather than circumventing the attention mechanism, pretraining encourages the model to utilize its full expressiveness when creating its sentence representations.

---

Figure 5.2: Comparison of attention weights produced by the pretrained and non-pretrained size=200 models, and the non-pretrained size=150 model on QIDs: 2279662, 23088804.
Figure 5.3: Learning curves for the non-pretrained and pretrained models. The graphs on the left depict the train accuracy, while the graphs on the right depict the dev accuracy.
<table>
<thead>
<tr>
<th>Ablated Component</th>
<th>Dev Accuracy</th>
<th>Test Accuracy</th>
<th>Test F1 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>0.824</td>
<td>0.814</td>
<td>0.814</td>
</tr>
<tr>
<td>Character-based Embeddings</td>
<td>0.818</td>
<td>0.811</td>
<td>0.812</td>
</tr>
<tr>
<td>Attention Mechanism</td>
<td>0.814</td>
<td>0.803</td>
<td>0.804</td>
</tr>
<tr>
<td>Fully-connected Classifier</td>
<td>0.666</td>
<td>0.660</td>
<td>0.660</td>
</tr>
<tr>
<td>Tags</td>
<td>0.812</td>
<td>0.805</td>
<td>0.806</td>
</tr>
</tbody>
</table>

Table 5.2: Ablation study results.

We additionally observe the regularizing effects of pretraining through the learning curves, which we produce in Figure 5.3. Although the non-pretrained models eventually achieve higher training accuracies, we notice that these models also have a tendency to overfit. In particular, we find that the dev accuracies of the pretrained models are typically higher than their non-pretrained counterparts, suggesting that the non-pretrained models are fitting to noise in the training dataset. The learning curves further demonstrate the effects of model under- and overfitting. For both the size=50 and size=200 cases, we observe that the gaps between pretrained and non-pretrained curves are significantly less pronounced than the gap in the size=150 case.

5.2 The Efficacy of the Model Components

In this section, we evaluate the efficacy of the model components through the results of our ablation study. Each of our ablations are similarly pretrained and tuned for fair comparison with our selected model, the pretrained size=150 model. For the remaining experiments, we pretrain and tune the models over 25 epochs rather than 50 epochs due to time constraints\footnote{Decreasing the number of training epochs to 25 does not affect comparison with our full model as this model was selected at epoch=16 and epoch=12 for pretraining and tuning respectively.}2. The results of the ablation study are shown in Table 5.2. We generally observe that each component contributes positively to the performance of our model.

5.2.1 Effect of Character-Based Embeddings

The rationale behind adding character-based embeddings is to have a backup mechanism for OOV words, as well as to take advantage of subword information in our word representations. Being able to handle OOV words should be especially important for the SO dataset, given that 45.8% of question pairs in the test set contain at least one word that is OOV. However, we see in Table 5.2 that the character-based embeddings only contribute a modest improvement of 0.002 to our test F1 score.
To better understand the character-based embeddings, we examine the attention weights produced by our full and ablated models. In Figure 5.4, we illustrate the attention weights produced by the models on a positive question pair that is correctly classified by the full model but incorrectly classified by the ablated model. We find that without the character-based embeddings, the attention weights tend to be quite diffuse, whereas with the embeddings, the attention mechanism is able to pinpoint the keywords in the question.

As such, character-based embeddings appear to be a double-edged sword. While they provide much-needed information for the model to pinpoint relevant information, the model may also have a tendency to overfit to these character-based embeddings. A potential solution to this problem would be to train the character-based embeddings separately on the broader SO corpus, allowing a wider range of relationships to be captured.

5.2.2 Effect of Tag Information

Our results show that tag metadata provides helpful contextual information towards the classification of question pairs, yielding an improvement of 0.008 to test F1 score. One possible reason for this improvement is to do with how questions are presented on SO. Question titles are typically displayed with their tags, leading some questioners to opt out from repeating the tags in their titles. For example, consider the following duplicate question pair:

**QID 3183210:** ruby on rails more elegant way to authenticate that users can edit only their own content

**QID 21632236:** how to only user to edit / update / delete their own questions?
This question pair is correctly identified by the full model but incorrectly identified by the ablated model. We can see how this misclassification arises: without the support of its tags, we cannot determine whether the two questions pertain to the same context. However, because both questions are tagged `<ruby-on-rails>`, this ambiguity is resolved in the full model.

### 5.2.3 Effect of Attention Mechanism

As we see in previous examples, the attention mechanism reliably highlights keywords that we think of as important to the question titles. The ability of the model to consider all the timesteps yields an improvement of 0.010 to test F1 score.

### 5.2.4 Effect of Fully-Connected Classifier

The largest contributor to performance among our ablated components is the fully-connected classifier. Specifically, we replace the classifier with the Manhattan distance, which has been reported by Mueller and Thyagarajan [23] to create geometrically coherent sentence representations. However, we find that the SO dataset requires the expressiveness of our fully-connected classifier to successfully compare the question pairs. Comparing the tag-ablated and classifier-ablated models, we see that the former outperforms the latter on test set F1 score by a large margin of 0.146.

One of the arguments proposed by Mueller and Thyagarajan [23] for using the relatively simple Manhattan distance is the idea that their LSTM would, in turn, be pushed to create more expressive representations. Although we do not perform any experiments exploring the transferability of these representations to our full model, we provide an example of the attention weights produced by the classifier-ablated model in Figure 5.5. We find that the classifier-ablated model attends to all the timesteps after the question title relative to the backwards GRU. This is quite different from what we see with the fully-connected classifier, which attends to the words in the question title. We conjecture that this is a form of overspecialization by the attention mechanism to produce representations that are separable by the Manhattan distance.
<table>
<thead>
<tr>
<th>Unfrozen Components</th>
<th>Dev Accuracy</th>
<th>Test Accuracy</th>
<th>Test F1 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>All (Fully-tuned)</td>
<td>0.824</td>
<td>0.814</td>
<td>0.814</td>
</tr>
<tr>
<td>&gt;= GRU</td>
<td>0.822</td>
<td>0.807</td>
<td>0.807</td>
</tr>
<tr>
<td>&gt;= Attention Mechanism</td>
<td>0.785</td>
<td>0.769</td>
<td>0.770</td>
</tr>
<tr>
<td>&gt;= Classifier and Tag Embeddings</td>
<td>0.778</td>
<td>0.760</td>
<td>0.761</td>
</tr>
<tr>
<td>None (Quora-trained)</td>
<td>0.522</td>
<td>0.560</td>
<td>0.560</td>
</tr>
</tbody>
</table>

Table 5.3: Model performance (Accuracy, Macro-Average F1 Score) with frozen layers. The >= indicates that all layers including and above the specified component(s), as presented in Figure 3.2, are unfrozen for training. The dev accuracy of the Quora-trained model is omitted as the model was selected against the Quora dev dataset.

<table>
<thead>
<tr>
<th>Unfrozen Components</th>
<th>Test P0</th>
<th>Test P1</th>
<th>Test R0</th>
<th>Test R1</th>
<th>Test F0</th>
<th>Test F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>All (Fully-tuned)</td>
<td>0.825</td>
<td>0.803</td>
<td>0.796</td>
<td>0.831</td>
<td>0.811</td>
<td>0.817</td>
</tr>
<tr>
<td>&gt;= GRU</td>
<td>0.823</td>
<td>0.792</td>
<td>0.782</td>
<td>0.832</td>
<td>0.802</td>
<td>0.812</td>
</tr>
<tr>
<td>&gt;= Attention Mechanism</td>
<td>0.784</td>
<td>0.756</td>
<td>0.743</td>
<td>0.795</td>
<td>0.763</td>
<td>0.775</td>
</tr>
<tr>
<td>&gt;= Classifier and Tag Embeddings</td>
<td>0.769</td>
<td>0.752</td>
<td>0.744</td>
<td>0.777</td>
<td>0.756</td>
<td>0.764</td>
</tr>
<tr>
<td>None (Quora-trained)</td>
<td>0.512</td>
<td>0.696</td>
<td>0.965</td>
<td>0.080</td>
<td>0.669</td>
<td>0.143</td>
</tr>
</tbody>
</table>

Table 5.4: Model performance (Precision, Recall, and F1 Score) with frozen layers. The >= indicates that all layers including and above the specified component(s), as presented in Figure 3.2, are unfrozen for training. P* denotes precision, R* denotes precision, and F* denotes F1 score. Furthermore, 0 denotes the negative (i.e. non-duplicate) label, while 1 denotes the positive (i.e. duplicate) label.

5.3 Transferable Knowledge between the Quora and SO Domains

We now consider the transferable knowledge between Quora and SO datasets by incrementally freezing the model components. The results from our various stages of freezing are presented in Tables 5.3 and 5.4. We use >= to denote that all layers including and above the specified component(s), as presented in Figure 3.2, are unfrozen for training.

Overall, we observe that freezing across logical components appears to avoid the problem of co-adaptation [13], in which neurons are “lost” between adjacent frozen and unfrozen layers. Specifically, we see performance improvements for each additional unfrozen component, with the largest performance gains occurring after unfreezing the classifier and tag embeddings, and the GRU. Nonetheless, we achieve our best performing model when we tune the full model.
5.3.1 Transferable Sentence Representations

From the results of our unfreezing experiment, we find that our sentence representations are transferable. We determine this by considering two metrics. First, we observe that when we pretrain our model with the Quora dataset and tune only the classifier and tag embeddings, the model achieves a reasonable test F1 score of 0.761. This suggests that the information encoded by the Quora-trained components is sufficient for determining question similarity in the SO dataset.

Our second observation is to do with the attention weights produced by the Quora-trained and fully-unfrozen models. Specifically, we find that these models attend to similar words, as demonstrated in Figure 5.6. In this example, we notice that both models emphasize “ordinal,” “column,” and [unk] in the first question, and “column” and [unk] in the second question.
To further analyze this behavior, we compute the Pearson and Spearman rank-order correlation coefficients between the attention weights produced by the two models over SO test set question pairs.\(^3\) We calculate the former as a measure for how the magnitudes of the attention weights relate to one another. The latter measures how similarly the models rank the GRU timesteps. The distributions of the correlations are presented in Figure 5.7. We find that the medians for the correlations are 0.738 and 0.765 for Pearson and Spearman correlation coefficients respectively, indicating strong correlation by both measures.

Putting these two observations together, we can say that the sentence representations not only encode salient information, but the models also consider the importance of words quite similarly. We conclude from these factors that the sentence representations are transferable between the Quora and SO domains.

5.3.2 Where’s the Discrepancy?

Despite performing semantically similar tasks, the Quora-trained model does not perform well on the SO dataset. This finding is similar to that in Mou et al.\(^{[14]}\), where training a model on a larger NLI dataset does not necessarily translate to strong performance on a separate, smaller NLI dataset. According to Mou et al., this is likely to do with the output layer becoming too specific to the larger dataset. In this subsection, we attempt to explore why a similar discrepancy arises between the Quora and SO datasets.

From Table 5.4, we see that the greatest shortcoming of the Quora-trained model on the SO dataset is its low recall on positive examples. An initial inspection reveals that the positive pairs correctly identified from the SO dataset by the Quora-trained model are often monotonically aligned. We notice a similar trend when we consider the positive examples in the Quora dataset. The following Quora question pair demonstrates monotonic alignment:

**QID 732446:** now donald trump is the president of us , what is the impact on india ?

**QID 732447:** what will be the impact of trump presidency on india ?

To verify this alignment more rigorously, we consider the Longest Common Subsequence Ratio (LCSR)\(^{[50]}\) between question pairs:

---

\(^3\)Both values are calculated after trimming the [pad] tokens from the beginning of each question. We exclude 26 out of the 20,000 questions in the test set because correlation coefficients cannot be computed over questions where the attention weights are all the same or over empty questions.
Figure 5.8: Histograms depicting the LCSRs for positive question pairs in the Quora and SO datasets, as well as those correctly identified in the SO dataset by the Quora-trained model.
\[ \text{LCSR}(q_1, q_2) = \frac{|\text{LCS}(q_1, q_2)|}{\max(|q_1|, |q_2|)} \]

where LCS is the Longest Common Subsequence.

We present the histograms for character and token overlaps in Figure 5.8. Although these LCSRs only match on a syntactic level, rather than on a semantic-level, we can still derive some insights from their distributions. On the whole, we notice that monotonic alignment may be a feature used by the Quora-trained model to identify duplicate examples, and that the SO dataset requires better usage of character-based embeddings to determine the similarity between question pairs.

Examining the token overlaps, we notice that (1) the distributions of the positive examples in the Quora dataset and those that were correctly identified in the SO dataset are quite similar, and (2) the amount of overlap, even by this non-semantic form of matching, is quite significant. These factors suggest that the Quora-trained model may be relying on monotonic alignment as a feature for determining question similarity. We further support this claim by examining the token overlap over all positive SO question pairs. In Figure 5.8f, we notice that there is hardly any token overlap for these examples. We believe that this lack of alignment is a factor behind the poor recall over the positive SO examples.

We attribute the monotonic alignment between Quora question pairs to Quora’s strict guidelines on asking questions [51]. These guidelines require that questions are written as complete and concise sentences ending with question marks. In contrast, SO only recommends that each question title should summarize a specific problem [52], inviting a wider range of question patterns.

When we compare the token and character distributions, we observe that the latter distributions are reasonably shifted to the right, given that the matching is performed at a more granular level. However, we find that the distribution over all positive SO pairs shifts most dramatically. We speculate that these additional overlaps arise because of identifier splitting, e.g. “ValueError” vs. “Value Error,” which we can only identify through character-level comparisons. As we discuss in Section 3.3.1, we utilize character-based embeddings to enable such comparisons.

\footnote{We note that this large amount of overlap may be a factor behind the success of the Parikh et al. [24] model on the Quora dataset, as examined in Tomar et al. [25]. The Parikh et al. model considers sentence representations unnecessary, instead performing pairwise comparisons between the words of the two question titles.}
Given that the shifts for the positive Quora examples and correctly identified positive SO examples are comparatively smaller, we can assume that the Quora-trained model can rely more heavily on word embeddings, which operate on the granularity of tokens, to identify similar words in the Quora dataset. However, the model would need to better utilize the character-based embeddings to successfully identify the similarities over all the positive SO pairs. We suspect that the model makes this adaptation when we unfreeze for the components \(>=\) GRU. This variant of unfreezing yields a 0.037 improvement to F1 score compared to when we only unfreeze components \(>=\) Attention Mechanism.

Overall, we argue that the relative homogeneity of question patterns and vocabulary among the positive Quora question pairs, in contrast to the SO question pairs, is a likely reason for the weak performance of the Quora-trained model on the SO dataset. We already address the issue of vocabulary through character-based embeddings. To address the diversity of question patterns, we may consider utilizing components that encode question information position-invariantly, e.g. CNNs, such that structural conventions are not embedded into the sentence representations.

5.3.3 Evaluating the False Positives

A final remark from Table 5.4 is that our models generally have lower recall on negative examples. A possible reason for this is the noisiness of the dataset. While our positive examples come with fairly strong guarantees, it is likely that some of the negative examples have not been closed as duplicates, which we empirically confirm by examining question pairs from the false positives. Selecting the first 100 (out of 1018) false positives, we observe that 17 questions are in fact true positives based on the duplicate classes laid out in the duplicate question guidelines [53]. We argue that the ability of our model to discern these examples speaks to its robustness despite the noisy data.

Within this sample, we additionally find that some SO question titles are worded ambiguously or provide insufficient information to be recognized as non-duplicate. Recall that our non-duplicate questions are related in that the question pairs link to one another. In the following example, the questioner in QID 5033497 explicitly asks for an explanation for an answer found in QID 1124534:

**QID 5033497:** in regards to array size calculation

**QID 1124534:** computing length of array
However, the relationship between the two questions cannot be inferred from just the question titles or the tags, which are `<c++>`, and `<c++>` and `<arrays>` respectively. We would need the full question body to understand that the questions are non-duplicate.

We also find that some questions are linked by answerers as suggested solutions. The titles to these solutions are often named quite similarly to the original questions. For example, an answerer references QID 665941 as a potential approach to solving QID 14121810:

**QID 14121810**: sharing security context between few web applications  
**QID 665941**: any way to share session state between different applications in tomcat?

For this question pair, our model fails to determine that “security context” is sufficiently different from “session state.” More broadly, our model can benefit from a wider catalog of semantic relationships between words on SO, potentially by training word embeddings on the entire SO corpus.
CHAPTER 6

CONCLUSIONS AND FUTURE DIRECTIONS

In this work, we attempt to address the question similarity task over SO question pairs. SO question pairs are unique in that the vocabulary is technical and rather diverse. Furthermore, our constructed dataset is noisy owing to the lack of a clear-cut technique for guaranteeing the correctness of our non-duplicate labels.

To address this issue, we build a robust model and apply transfer learning techniques to translate intuitions about question pairs from the Quora domain to the SO domain. We draw the following conclusions from our results:

1. Transfer learning is an effective technique for improving model robustness over the question similarity task, and may serve as an effective regularization mechanism. We find that our best pretrained model effectively detects similar questions with test F1 score 0.814. This is a 0.011 improvement to F1 score over its non-pretrained counterpart.

2. We find that each of our model components contributes positively to performance on the SO dataset. In particular, we find that our fully-connected classifier provides the necessary expressiveness to distinguish duplicate questions, and that the attention mechanism allows us to holistically summarize question titles. Interestingly, we notice that the character-based embeddings allow the model to pinpoint important words, but the model may have a tendency to overfit to these embeddings.

3. Our model effectively captures sentence representations that are transferable between Quora and SO domains. We also identify, with help from the model, interesting structural differences between duplicate questions in the two datasets despite their tackling the same question similarity task.

We see a number of directions to extend this work. With regards to our model, we propose training the character-based embeddings on the broader SO corpus to prevent the embeddings from overfitting to our dataset. We would also like to explore the use of position-invariant components, e.g. CNNs, to produce our sentence representations. Specifically, we are interested in whether using such components will allow us to continue capturing salient question title information, but in a manner that is robust to the structural differences between the questions in the Quora and SO datasets.

Our work can also be extended in terms of the information we provide the model. For example, we can consider developing an asymmetric model, where one side of the model only takes the question title and tags, while the other side takes the question title, tags, and body. We can use a hierarchy of attention mechanisms [5] to summarize the question body. This way, we address a number of the information gaps that we identified in Section 5.3.3.
Another direction is the use of social information to inform our model with likely topics that the user may be interested in. For example, we can inject information about previously asked, commented, or voted questions, and utilize this information to narrow the search space for similar questions. This can be especially useful when addressing ambiguous questions, such as the question explored in Section 5.2.2. If we know a user has been engaging with questions from a particular topic, it is likely that following questions will be from a similar domain. More broadly, we can consider using various forms of unsupervised pretraining on the SO corpus to enrich our model.

With respect to our specific experiment design decisions, we can extend our work by considering other techniques for non-duplicate example selection. As explored in Section 3.1.1, we use related questions to create our non-duplicate dataset. We do this with intention of encouraging the model to distinguish specific differences between question pairs, rather than relying on surface differences between the questions. However, it would be interesting to explore how strongly this assumption holds. For example, we can construct a dataset similarly to Silva et al. [37], pairing a duplicate question with a random question, and explore how this form of sampling affects model performance. It would be intriguing to see whether this dataset confuses the model or makes the task comparatively easier.

One final direction may be to vary the size of our tuning dataset, as in Chung et al. [15]. Specifically, we note that both of our Quora and SO datasets are comparably sized. Given the robustness of our sentence representations, as demonstrated in Section 5.3.1, we can further explore how large a dataset is really necessary to achieve our eventual performance.


of words and phrases and their compositionality,” in Proceedings of the 26th Interna-
tional Conference on Neural Information Processing Systems - Volume 2, ser. NIPS ’13,

(visited on 04/15/2018).

neural networks?” In Proceedings of the 27th International Conference on Neural In-
formation Processing Systems - Volume 2, ser. NIPS ’14, Montreal, Canada: MIT Press,

Methods in Natural Language Processing, ser. EMNLP ’16, Austin, Texas: Association


[16] Y. Zhang, D. Lo, X. Xia, and J.-L. Sun, “Multi-factor duplicate question detection in
stack overflow,” Journal of Computer Science and Technology, vol. 30, no. 5, pp. 981–
997, Sep. 2015.

programming question answering communities,” ACM Trans. Internet Technol., vol. 18,

[18] Q. Le and T. Mikolov, “Distributed representations of sentences and documents,” in Pro-
cedings of the 31st International Conference on International Conference on Machine
Learning - Volume 32, ser. ICML’14, Beijing, China: JMLR.org, 2014, pp. II-1188–II-
1196.


mary to developers’ technical questions,” in Proceedings of the 32nd IEEE/ACM Interna-
tional Conference on Automated Software Engineering, ser. ASE ’17, Urbana-Champaign,

knowledge in developer online forums via convolutional neural network,” in Proceedings
of the 31st IEEE/ACM International Conference on Automated Software Engineering,


