Transfer Learning for Question Similarity between Stack Overflow Posts

Victor Wing-Chuen Kwan
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
vwkwan@connect.ust.hk

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: questions vary significantly in formality, specificity, and style. Drawing from a wide pool of natural language processing techniques, we devise a model for question similarity that attempts to learn the semantic relationships between Stack Overflow questions using the titles and tags of posts. We demonstrate that pretraining against Quora data provides robustness against the noisy Stack Overflow dataset. We additionally provide a study into the effectiveness of our model components, and investigate the differences between the two datasets through the lens of tranferred knowledge.

1 Introduction

Stack Overflow (SO) is among the most popular developer-oriented Community Question Answering (cQA) websites. Owing to the technicality of the content, questions are asked with a wide range of specificity, formality, and style. Ahasanuzzaman et al. (2016) suggest that this diversity contributes to the prevalence of duplicate questions, with users unable to strike the right balance of keywords to find existing questions that match their own.

Consider the question pair, "Check connection is active in ASP.NET?" and "How to check if user is still present?" (QIDs: 761376, 3374120). Only by considering the *nuance* of the questions (i.e. present users maintain active connections) can we determine that these questions are duplicates. We argue that the large volume of duplicate questions on SO can be attributed to the failure of existing question retrieval systems to capture such *semantic* similarities between question pairs.

To address the question similarity task, we draw from a wide pool of Natural Language Processing (NLP) techniques. For example, we apply attention mechanisms to pinpoint informative words when constructing our sentence representations (Yang et al., 2016). We additionally leverage subword information (Bojanowski et al., 2016) to pry into the internal structure of words, which is relevant when dealing with a technical vocabulary. While there have been many attempts (dos Santos et al., 2015; Lei et al., 2016; Wang et al., 2017) tackling the question similarity task, our work differentiates itself in that the SO dataset is noisy and necessitates building around transfer learning.

Our model is built around the assumption that our non-duplicate examples are noisy. Although duplicate questions are reliably marked by users and moderators, there are no corresponding mechanisms for selecting non-duplicate questions. To address this issue, we learn robust, *sentence*-level representations based on Quora's question similarity dataset (Iyer et al., 2017), then transfer these intuitions towards identifying duplicate SO questions. This way, we take advantage of the "catchment basin" (Mou et al., 2016b) formed by pretraining to avoid overfitting to noisy examples.

In summary, this work contributes a model for question similarity built with robust sentence representations in mind. Our model achieves a fairly high test F1 score of 0.814, which is over a one point improvement over its non-pretrained counterpart. We additionally provide an ablation study exploring how our model components inform the question similarity task, as well as a study into how model size and frozen layers may affect the transfer learning process, with a focus on the differences between the Quora and SO datasets.

2 Related Work

There are a number of works focusing on duplicate question detection on SO (Zhang et al., 2015; Ahasanuzzaman et al., 2016; Zhang et al., 2018). These approaches largely rely on features that we jointly learn through our model. Xu et al. (2016) make a preliminary effort at applying Deep Neural Networks (DNNs) towards classifying the semantic relatedness of SO posts. However, their Convolutional Neural Network (CNN) based approach is trained on a comparatively small dataset, and considers significantly more information per post, including the question and answer bodies on top of our question titles and tags. We make the latter adaptation to align our task more closely with question retrieval from a user's perspective.

Many works approaching textual (e.g. document, sentence, question) matching tasks follow the Siamese network architecture (Bromley et al., 1993). This architecture breaks the matching process into two steps. The first step, creating vector representations, has been explored through encoder-decoder models (Lei et al., 2016) as well as CNNs over Bags of Words (BoW) (dos Santos et al., 2015). The second step, classifying the vector representations, has been explored through the Manhattan distance (Mueller and Thyagarajan, 2016) and multi-perspective matching (Wang et al., 2017). Parikh et al. (2016) propose forgoing the initial step, directly comparing documents on a word-by-word basis. Our model adapts the Siamese network architecture with a focus on creating sentence representations that can be transferable from the Quora domain to the SO domain.

Transferable sentence representations have been investigated through both unsupervised (Le and Mikolov, 2014) and supervised (Conneau et al., 2017) techniques. For example, Conneau et al. (2017) show that representations trained against the Natural Language Inference (NLI) task can generalize well to other NLP tasks. In contrast, we transfer our sentence representations between datasets pertaining to semantically similar tasks; specifically, we are interested in transfer *to* a noisy target domain. This brings our work closer in line to that of Tomar et al. (2017), who transfer *from* a noisy source domain using the aforementioned Parikh et al. model. Our work is also similar to that of Wiese et al. (2017), who target the similarly-technical biomedical domain.

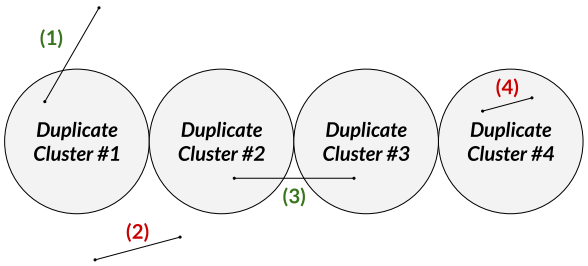


Figure 1: An overview of accepted and rejected linked question pairs. The gray circles are connected components in the duplicate question graph. We accept pairs (1) and (3) because they are *relevant* to our duplicate pairs but not transitively duplicates of one another. We reject (2) because it is irrelevant to our set of duplicate pairs, and we reject (4) because the questions are transitively duplicates of one another.

3 Dataset Construction

Our task can be formulated as a binary classification problem of whether or not two questions are duplicates. As such, we construct our SO dataset with both duplicate and non-duplicate examples.

SO Question Pairs

The SO dataset is assembled using the Stack Exchange Data Explorer (Stack Exchange Inc., 2018). The Data Explorer allows us to perform SQL queries on the SO database, returning at most 50,000 rows per query. As such, 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. (2016). 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. In total, we retrieve 137,793 question pairs for use as positive examples.

In line with the Quora Question Pairs dataset (Iyer et al., 2017), we select pairs of related questions as our negative examples. As

in Xu et al. (2016), we use the linking between posts as a proxy for question relatedness. For each linked question pair, we perform two forms of filtering, using the NetworkX package (Schult, 2008) to construct our duplicate pair clusters. The filtering process is presented in Figure 1. We ensure that question pairs cannot be transitive duplicates of one another, and that at least one of the questions must be associated with a duplicate example. We add the latter contraint to ensure that the model cannot simply memorize questions in the duplicate examples. Despite this filtering, we cannot claim that our duplicate clusters can catch all duplicate examples, so our negative examples are inevitably noisy.

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. 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 collecting these question pairs, we tokenize the question titles using spaCy (Honnibal and Johnson, 2015). We then 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

We use the splits published by Wang et al. (2017) to create our Quora dataset. However, because these splits are already tokenized using the Stanford CoreNLP (Manning et al., 2014) 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 255,045 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. Unlike those in the SO dataset, the Quora questions do not contain any tag information.

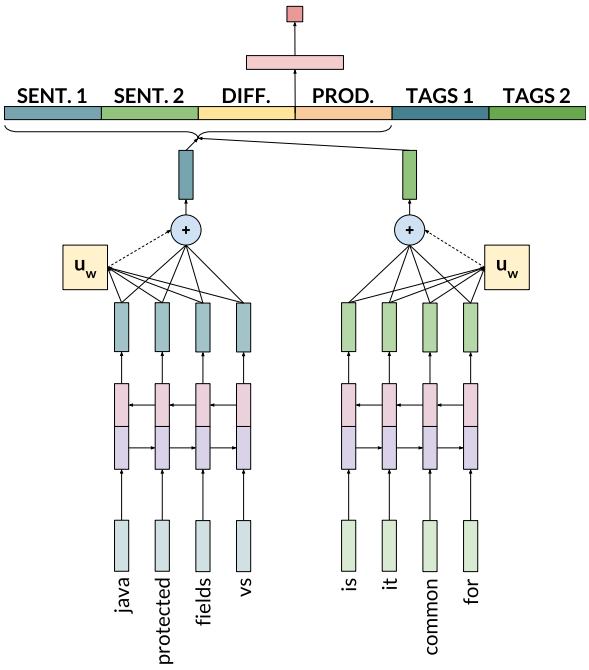


Figure 2: An overview of our model.

4 Model Components

Our model follows the Siamese network architecture (Bromley et al., 1993). An overview of our model is presented in Figure 2. The input to our Gated Recurrent Units (GRUs) consists of a concatenation of pretrained word embeddings and character-based embeddings.

Word Embeddings

The pretrained word embeddings utilize 300d GloVe vectors (Pennington et al., 2014) trained on the Wikipedia 2014 and Gigaword 5 corpora. Only words found in our Quora and SO training corpora are indexed by our word embeddings.

Character-Based Embeddings

Our character-based embeddings are informed by the techniques in Seo et al. (2016) and follow similar hyperparameters to Lee et al. (2017). Characters are projected into 8d character embeddings, then convolved over using kernels of sizes $[3, 4, 5]$. Each convolving kernel has 50 output channels. The output from each kernel is then max-pooled, passed through a ReLU nonlinearity, then 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 (Cho et al., 2014) to create our sentence representations. We use the concatenation of hidden states from the forward and backward GRUs as sentence representations at each timestep. As such, the dimension of our GRU output is double the GRU hidden size.

Attention Mechanism

Our attention mechanism draws from Yang et al. (2016) and can be described as follows:

$$
\begin{aligned}
u_t &= \text{ReLU}(W_w h_t + b_w) \\
\alpha_t &= \frac{\exp(u_t^T u_w + b_c)}{\sum_i \exp(u_i^T u_w + b_c)} \\
v &= \textstyle\sum_i \alpha_i h_i
\end{aligned}
$$

Where h_t is timestep t of the GRU output, u_w is a trainable context vector, and b_c is a trainable bias term. We subsequently refer to the output dimension of W_w as the "attention size". We construct our sentence representations by taking the weighted sum of each timestep of the GRU output with respect to the scores assigned by the attention mechanism. These scores are computed by taking the dot product of the transformed GRU output with the context vector.

Fully-Connected Classifier

We compare the sentence representations using two fully-connected layers. The input to our classifier draws from Mou et al. (2016a).

$$
\begin{aligned}
m &= [v_1; v_2; |v_1 - v_2|; v_1 \circ v_2; t_1; t_2] \\
h &= \text{ReLU}(W_h m + b_h) \\
o &= \text{sigmoid}(W_o h + b_o)
\end{aligned}
$$

Where v_1 and v_2 denote the representations of the first and second questions respectively; $\circ$ denotes the element-wise product; and t_1 and t_2 denote the concatenated tag embeddings for the first and second questions respectively. ; 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.

Owing to the large number of tags on SO, we compress each tag into a 16d tag embedding. Each question on SO can be at most assigned five tags, so we cap our input to contain five tags per question, assigning `[pad]` tokens in the case that there are fewer than five tags. Quora question pairs provide no tag information, so the tag input for these questions consists entirely of `[pad]` tokens.

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 predicted as a positive example.

5 Model Configuration and Training

Our model is implemented using PyTorch (Paszke et al., 2017) and the experiments are conducted on a single Nvidia GeForce GTX 1070 GPU with 8GB GDDR5 RAM. Word embeddings are initialized and frozen with GloVe vectors (Pennington et al., 2014) corresponding to the vocabulary in the Quora and SO training corpora. We use He normal initialization (He et al., 2015) to initialize our components unless we are copying weights from a pretrained model.

For training, we use the Adam optimizer (Kingma and Ba, 2014) with default settings. We fix the learning rate at `lr=1e-3` for both training and evaluation as we find that maintaining the learning rate does not result in the destruction of knowledge accumulated from pretraining. When training our model with the fully-connected layers for classification, we use the binary cross-entropy loss function. For our ablation study, we consider replacing the fully-connected layers with the Manhattan distance; in this case, we use the mean-squared error loss function as discussed in Mueller and Thyagarajan (2016).

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 fully-connected layers respectively. Where the outputs sizes are fixed, i.e. the output of the character-based embeddings and the classifier hidden layer, we additionally perform batch normalization (Ioffe and Szegedy, 2015) to speed up the convergence of our model.

For our character inputs, we truncate each word at 10 characters, which sufficiently covers more than the 90th percentile[1] of words in our training corpora. For both training and evaluating our models, we use a batch size of 64 examples.

[1]The word at the 90th percentile has 8 characters.

Configuration	Dev Accuracy	Test Accuracy	Test F1
50, NPT	0.805	0.796	0.797
50, PT	0.808	0.795	0.795
100, NPT	0.814	0.803	0.803
100, PT	0.819	0.807	0.807
150, NPT	0.814	0.801	0.803
150, PT	**0.824**	**0.814**	**0.814**
200, NPT	0.819	0.809	0.809
200, PT	0.821	0.809	0.809

Table 1: Hyperparameter tuning results. NPT denotes "non-pretrained", while PT denotes "pretrained". Our selected model is **bolded**.

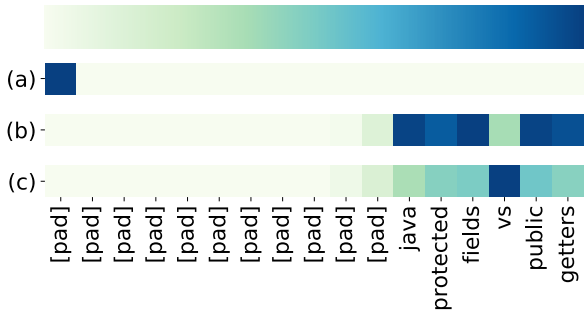


Figure 3: Attention weights for QID: 23088804. The attention weights correspond to those produced by the (a) `size=200`, non-pretrained; (b) `size=200`, pretrained; (c) `size=150`, non-pretrained models.

6 Experiments and Results

Under- and Overfitting Models

As a byproduct of hyperparameter tuning, we investigate the effect of GRU and attention sizes on model performance.

- **Model sizes.** We fix the GRU hidden and attention sizes to be equal, selecting sizes from $[50, 100, 150, 200]$ output neurons.
- **Transfer process.** We begin by training our model on the Quora dataset for 50 epochs. We then select the model from the epoch with highest dev accuracy as our Quora-trained model. When we tune this model against the SO dataset, we re-initialize the weight matrices of the fully-connected classification layers and train for another 50 epochs.
- **Model selection.** For both non-pretrained and pretrained cases, the model from the epoch with highest dev accuracy is selected for evaluation.

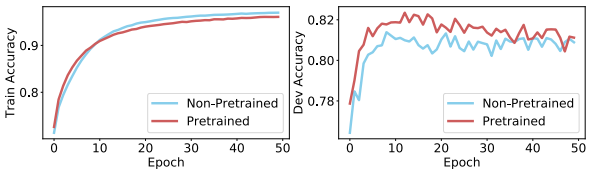
From Table 1, we see that pretraining generally allows us to converge with higher dev accuracy. However, we also see that these performance gains can fail to generalize when we evaluate the pretrained models against the test set. For example, we see that the `size=50` model suffers from underfitting, performing *worse* than its non-pretrained counterpart on the test set. Meanwhile, the `size=200` model suffers from overfitting and achieves the same accuracy as its non-pretrained counterpart.

Pretraining as a regularization mechanism. We begin our exploration of how pretraining helps us by considering the attention weights produced by our models, which we produce in Figure 3. Interestingly enough, for the non-pretrained `size=200` model, we observe that the model attends only to the last GRU timestep[2]. This would suggest that the model has learnt to entirely circumvent the attention mechanism, relying exclusively on the sentence representation produced by the backwards GRU. We contrast this with the pretrained `size=200` model, where the model shares the work between both GRU and attention mechanism. The non-pretrained `size=150` model validates that focusing on a single timestep is unusual behavior. To this end, we argue that pretraining is acting as a form of regularization, preventing the model from being overly reliant on the GRU.

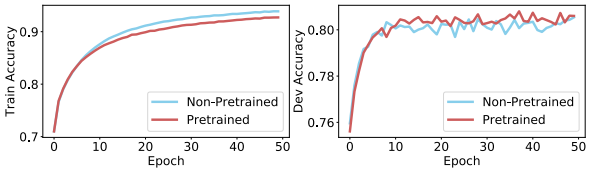
We additionally see the regularizing effects of pretraining through the learning curves, which we produce in Figure 4. Although the non-pretrained models eventually achieve higher training accuracies, we see that this is because the models tend to overfit: the dev accuracies of the pretrained models are consistently higher than those of their non-pretrained counterparts. We can additionally observe the effects of under- and overfitting in the dev plots. For both the `size=50` and `size=200` cases, the differences between the pretrained and non-pretrained curves are significantly less pronounced than in the `size=150` case.

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.

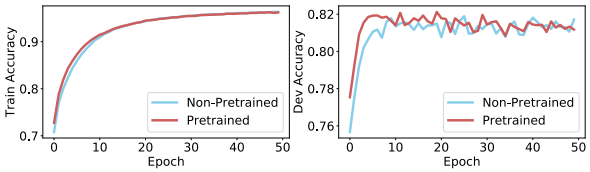
[2]We note that Conneau et al. (2017) observe similar but less exaggerated behavior when max-pooling over the outputs of a non-pretrained LSTM. In their appendix, they show that max-pooling generally *focuses* on the initial and final timesteps. Our attention mechanism takes this to the extreme, focusing *exclusively* on the final timestep of the backward GRU.



(a) `size=150` models.



(b) `size=50` models.



(c) `size=200` models.

Figure 4: Learning curves for the non-pretrained and pretrained models.

When applying McNemar's test, the null hypothesis states that there is *marginal homogeneity* between the two models, which is to say that the marginal probabilities for each outcome are the same. We reject the null hypothesis of marginal homogeneity with $P < 0.001$. Together with the improved model F1 score, we demonstrate that transferring between the Quora and SO domains is beneficial towards the question similarity task.

Model Component Efficacy

In this section, we perform an ablation study to better understand the efficacy of our model components. The following ablations require slightly more involved modifications to our model:

- **Attention mechanism.** The model with ablated attention mechanism uses a unidirectional GRU, passing the final GRU timestep to the fully-connected classifier as the sentence representation.
- **Fully-connected classifier.** The model with ablated fully-connected classifier utilizes the Manhattan distance to determine the distance between sentence representations, as explored in Mueller and Thyagarajan (2016). As this model cannot incorporate tag information, we provide a tag-ablated model for comparison.

For the remaining experiments, we pretrain and tune our ablated models over 25 epochs rather than 50 epochs due to time constraints[3]. The results of the ablation study are shown in Table 2. We generally observe that each component contributes positively to the performance of our model.

Ablated Component	Dev Accuracy	Test Accuracy	Test F1
None	0.824	0.814	0.814
Character-based Embeddings	0.818	0.811	0.812
Attention Mechanism	0.814	0.803	0.804
Fully-connected Classifier	0.666	0.660	0.660
Tags	0.812	0.805	0.806

Table 2: Ablation study results.

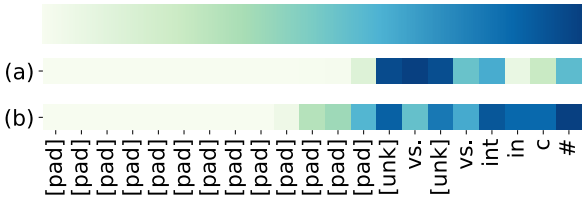


Figure 5: Comparison of attention weights between (a) the full model and (b) the model with ablated character-based embeddings on QID: 1680111. The `[unk]`s are "int32" and "int64" respectively.

Character-based embeddings. We observe that character-based embeddings *should* be especially important to the SO dataset, given that 45.8% of question pairs in the test set contain at least one word that is out of vocabulary. However, we see in Table 2 that adding the character-based embeddings only yields a modest improvement of 0.002 to test F1 score.

Figure 5 illustrates how our character-based embeddings interact with our attention mechanism for `[unk]`-containing examples. We find that without the character-based embeddings, the attention weights tend to be quite diffuse, where as 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 to classify question pairs, they may also become overly strong signals, causing the model to overfit. Pretraining the character-based embeddings on the broader SO dataset may help alleviate this problem.

Fully-connected classifier. The largest contributor to performance among our ablated components is the fully-connected classifier. We find that

[3]Decreasing the number of epochs does not affect comparison with our full model as this model was selected at `epoch=16` and `epoch=12` for pretraining and tuning respectively.

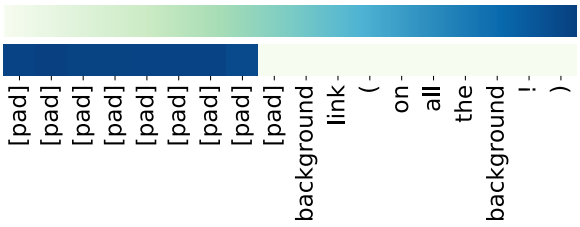


Figure 6: Attention weights for the model using the Manhattan distance as its distance metric on QID: 11545649.

Unfrozen Components	Dev Accuracy	Test Accuracy	Test F1
All	0.824	0.814	0.814
>= GRU	0.822	0.807	0.807
>= Attention mechanism	0.785	0.769	0.770
>= Classifier	0.778	0.760	0.761
None (Quora-trained)		0.522	0.560

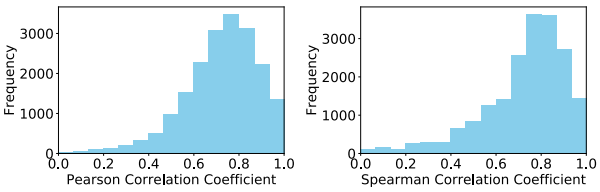
Table 3: Model performance with frozen layers. The >= indicates that components *including and after* the specified component are unfrozen for training.

our fully-connected classifier provides the necessary expressiveness to 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.

An interesting question was whether or not the sentence representations produced by the ablated model would be better transferable, given that the GRU and attention mechanism are pushed to create more expressive representations. Visualizing the attention weights, as in Figure 6, reveals that the 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 may suggest an overspecialization of the attention mechanism to produce representations that are separable by the Manhattan distance metric.

Transferable Knowledge between the Quora and SO Domains

We now consider the transferable knowledge between Quora and SO datasets by incrementally freezing the model layers. The results from our various stages of freezing are presented in Table 3. For each of these variants, we only re-initialize the weight matrices of the fully-connected classifier. Overall, we find that the largest performance increases come after we unfreeze the classifier and the GRU. Tuning the entire model produces the best results.



(a) Pearson correlation. (b) Spearman correlation.

Figure 7: Correlation between attention weights on the test set, as generated by Quora-trained and fully-unfrozen models.

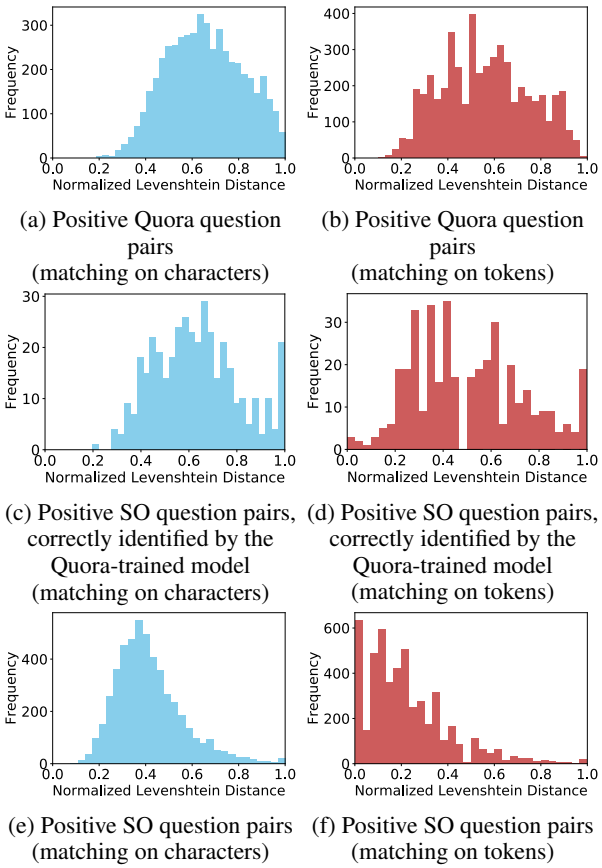
Where's the discrepancy? Despite performing semantically similar tasks, the Quora-trained model does not perform well on the SO dataset. This finding is similar to that in Mou et al. (2016b), where training on a larger NLI dataset does not necessarily translate to good performance on a separate, smaller NLI dataset.

We begin our exploration for *why* this is the case by comparing the words attended to by the Quora-trained and fully-unfrozen models. Specifically, we calculate the Pearson and Spearman rank-order correlation coefficients between the attention weights of question pairs in the test set[4]. The distribution of the correlations are presented in Figure 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. Considering the two metrics together, we can conclude that the two models not only share similar notions of relative importance, but also weigh the magnitude of importance quite similarly.

We then consider the model performance when only the classifier is unfrozen. This results in a 0.201 increase in test F1 score. Considering that attention weights are strongly correlated, and that sentence representations appear to encode the necessary information for questions to be classifiable, the difference then is likely to do with the *criteria* with which questions are considered duplicates.

The most significant shortcoming of the Quora-trained model is its exceptionally low recall (0.080) on positive examples. An initial inspection reveals that the positive pairs correctly identified from the SO dataset generally appear to be monotonically aligned, e.g. "**difference between static**

[4]Both values are calculated after trimming the `[pad]` tokens from the beginning of each question. Furthermore, we exclude 22 out of the 20,000 questions in the test set because the correlation coefficients cannot be calculated for questions where the attention weights are all the same.



(a) Positive Quora question pairs (matching on characters)

(b) Positive Quora question pairs (matching on tokens)

(c) Positive SO question pairs, correctly identified by the Quora-trained model (matching on characters)

(d) Positive SO question pairs, correctly identified by the Quora-trained model (matching on tokens)

(e) Positive SO question pairs (matching on characters)

(f) Positive SO question pairs (matching on tokens)

Figure 8: NLD histograms.

class and **singleton** pattern ?" and "what is the main **difference between static class** & **singleton** class" (QIDs: 519520, 23566750). We notice a similar trend when we consider the positive examples in the Quora dataset. We verify this alignment more rigorously using the Normalized Levenshtein Distance (NLD) between each question pair:

$$\text{NLD}(q_1, q_2) = \frac{|\text{LCS}(q_1, q_2)|}{\max(|q_1|, |q_2|)}$$

Where LCS is the Longest Common Subsequence. The histograms for character-based and token-based NLDs are presented in Figure 8.

Examining the *character*-based overlaps, we find that the distribution of positive examples in the Quora dataset follows a similar distribution to those that were correctly identified in the SO dataset. On the other hand, we see that the distribution of all positive examples in the SO dataset is skewed to the left. This would suggest that the model *assumes* a certain level of monotonic alignment between the question pairs, which is an assumption that does not translate well when we take the entire SO dataset into consideration.

We additionally notice differences in vocabulary when we consider the distributions of *token*-based overlaps. In contrast to the fairly consistent matching between tokens in positive Quora examples, Figure 8f reveals that there are very few token matches between positive SO question pairs. We see that this difference is far more exaggerated than when we compare the character-based overlaps. We conjecture that the gulf between the token- and character-based overlaps may arise due to vocabulary splitting, e.g. "ValueError" vs. "Value Error" when describing a camel-case identifier. This would suggest that the SO dataset necessitates better usage of the character-based embeddings, which is a likely adaptation when the GRU is unfrozen for tuning (yielding a 0.037 improvement to test F1 score).

Overall, we argue that the relative homogeneity of Quora question pairs in contrast to SO question pairs is a likely reason for the weak performance of the Quora-trained model on the SO dataset. We find that this homogeneity is a result of policy: Quora has very strict policies (Quora, 2017) on how questions may be written, requiring that they should be complete, concise questions. Regarding the diversity of question structures, we may consider applying components that encode question information position-invariantly, e.g. CNNs, such that structural conventions are not embedded into the sentence representations.

7 Conclusion

In this work, we address the question similarity task over SO question pairs. The SO dataset is unique in that the vocabulary is technical and rather diverse. Furthermore, our constructed dataset is noisy owing to our coarse methods when supplementing the dataset with negative examples.

To address this issue, we built a model with the intention of translating intuitions about question pairs from the Quora domain to the SO domain. We find that transfer learning is an effective technique for increasing the robustness of models over the question similarity task, and may serve as an effective regularization mechanism. We additionally find that each of our model components contributes positively to model performance. Finally, we identify, with help from the model, interesting structural differences in what duplicate questions look like between the two datasets despite their tackling the same question similarity task.

References

Muhammad Ahasanuzzaman, Muhammad Asaduzzaman, Chanchal K. Roy, and Kevin A. Schneider. 2016. Mining duplicate questions in stack overflow. In *Proceedings of the 13th International Conference on Mining Software Repositories*, MSR '16, pages 402–412, New York, NY, USA. ACM.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a "siamese" time delay neural network. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS '93, pages 737–744, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP '14, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the 2015 IEEE International Conference on Computer Vision*, ICCV '15, pages 1026–1034, Washington, DC, USA. IEEE Computer Society.

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, EMNLP '15, pages 1373–1378, Lisbon, Portugal. Association for Computational Linguistics.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 448–456. JMLR.org.

Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. 2017. First quora dataset release: Question pairs.

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

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1188–II–1196. JMLR.org.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, EMNLP '17, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

Tao Lei, Hrishikesh Joshi, Regina Barzilay, Tommi Jaakkola, Kateryna Tymoshenko, Alessandro Moschitti, and Lluís Màrquez. 2016. Semi-supervised question retrieval with gated convolutions. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '16, pages 1279–1289, San Diego, California. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, ACL '14, pages 55–60.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016a. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, ACL '16, pages 130–136, Berlin, Germany. Association for Computational Linguistics.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016b. How transferable are neural networks in nlp applications? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, EMNLP '16, pages 479–489. Association for Computational Linguistics.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, AAAI '16, pages 2786–2792. AAAI Press.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, EMNLP '16, pages 2249–2255, Austin, Texas. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP '14, pages 1532–1543.

Quora. 2017. What are the main policies and guidelines for questions on quora?

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)*, ACL '15, pages 694–699, Beijing, China. Association for Computational Linguistics.

Daniel A. Schult. 2008. Exploring network structure, dynamics, and function using networkx. In *In Proceedings of the 7th Python in Science Conference*, SciPy '08, pages 11–15.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Stack Exchange Inc. 2018. Stack exchange data explorer.

Gaurav Singh Tomar, Thyago Duque, Oscar Täckström, Jakob Uszkoreit, and Dipanjan Das. 2017. Neural paraphrase identification of questions with noisy pretraining. In *Proceedings of the 1st Workshop on Subword and Character Level Models in NLP*, SCLeM '17, pages 142–147, Copenhagen, Denmark. Association for Computational Linguistics.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI '17, pages 4144–4150.

Georg Wiese, Dirk Weissenborn, and Mariana Neves. 2017. Neural domain adaptation for biomedical question answering. In *Proceedings of the 21st Conference on Computational Natural Language Learning*, CoNLL '17, pages 281–289, Vancouver, Canada. Association for Computational Linguistics.

Bowen Xu, Deheng Ye, Zhenchang Xing, Xin Xia, Guibin Chen, and Shanping Li. 2016. Predicting semantically linkable knowledge in developer online forums via convolutional neural network. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, ASE '16, pages 51–62, New York, NY, USA. ACM.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '16, pages 1480–1489, San Diego, California. Association for Computational Linguistics.

Wei Emma Zhang, Quan Z. Sheng, Jey Han Lau, Ermyas Abebe, and Wenjie Ruan. 2018. Duplicate detection in programming question answering communities. *ACM Trans. Internet Technol.*, 18(3):37:1–37:21.

Yun Zhang, David Lo, Xin Xia, and Jian-Ling Sun. 2015. Multi-factor duplicate question detection in stack overflow. *Journal of Computer Science and Technology*, 30(5):981–997.