6.864 Final Project Report

Link to our source code:
https://github.com/hongxiangqiu/MIT-6.864-Question-Retrieval-and-Transfer-Learning

Introduction

As question answering forums are growing rapidly, it's useful to reuse or refer to the answers from the similar questions when user has an question to ask. In this project we aim to implement models that can identify similar questions and apply our model to a different domain using techniques of transfer learning. We will present the work in 3 parts: question similarity model, direct transfer models, and adversarial models.

Data

In question retrieval task we trained and evaluated our model on Stack Exchange AskUbuntu dataset(link), in domain transfer task we evaluated on Stack Exchange Android dataset(link).

Question Retrieval

Assuming we have a question to ask, and we already have a large set of question-answer corpus, the most obvious way to find a answer to a question is to find the most similar ones and use their answers. In our question retrieval setting, we try to score the similarity of any two questions.
Based on *Semi-supervised Question Retrieval with Gated Convolutions* paper, if we have an encoder to encode input questions as vector representations, then we can construct a scoring model based on the similarity of their encodings. Under this setting, we can try a variety of models as our encoders. RNN and CNN are widely used as feature extractor/ encoder in deep learning literatures, thus we attempted build models using LSTM (a variation of RNN) and CNN, and tuned their parameters looking for best performance evaluated using MAP, MRR, P@1 and P@5. (mean average precision, mean reciprocal rank, precision at 5 and precision at 1)

LSTM

LSTM is a widely used RNN model, where in additional to recurrent structure, there is a forget gate to control for how much cell state to carry on; an input gate to control in-flow information; and an output gate to control out-flow information. The advantage of using these gates is the ability to remember long-term dependency of inputs. Our best performing model has 240 hidden dimensions, and beats BM25 model mentioned by the paper by a quite large margin in all evaluation metrics.

	MAP	MRR	P@1	p@5
Dev	0.579	0.719	0.598	0.469
Test	0.580	0.708	0.559	0.439

CNN

In this part we implemented the CNN model pretty much the same as the simple CNN described in *Semi-supervised Question Retrieval with Gated Convolutions* paper. In particular, we have a convolutional layer followed by a dropout layer and finally a tanh activation layer. We have set the output hidden encoding size to be 667, which is the best parameter setting suggested by the paper. Also, after

many experiments with the hyper parameters, we find the best performance at learning rate = 1e-4 and margin = 0.3. We were able to beat the performance on development set of BM25 model performance mentioned in the paper on all the metrics, however our P@5 on test set is a little lower than BM25 performance mentioned in the paper. The exact performance of our CNN model is reported below:

	MAP	MRR	P@1	p@5
Dev	0.546	0.674	0.556	0.435
Test	0.539	0.675	0.538	0.408

Domain Transfer

Now consider the following setting: we have a large labeled question-answering corpus (similar questions are labeled), and we have an unlabeled question-answering corpus, then how should we train our model to score similarity on the unlabeled corpus using our labeled data? This is a transfer learning question, and we attempted three different methods to solve this problem: unsupervised model, direct transfer, and transfer using adversarials. In our experiments, we are given labeled data on Android dataset, and unlabeled Ubuntu corpus (except for a small portion of labeled dev and test data, but we only use these labels for evaluation purpose).

Metrics

In oppose to the metrics we used before, we are now using AUC 0.05 score as our evaluation metric. AUC 0.05 is the AUC score only considering false positive ratio < 0.05. We are using it because we have so many false negatives in our dev/test data (many similar questions are not labeled).

Unsupervised Model (TF-IDF)

Unsupervised models like tf-idf cosine-similarity doesn't require any labeled data, therefore can be applied directly on the target corpus. We simply get tf-idf vector for the pair of questions (vector of tf-idf scores of each word in the vocabulary), and score the pair using the cosine similarity of their tf-idf vectors. Using this simple model, we got quite amazing performance:

	AUC 0.05
Dev	0.707
Test	0.739

Direct Transfer

For the direct transfer part, we tried both LSTM and CNN models on the direct transfer task. We simply took the exact same structure of both models over to evaluate on the Android dataset. Except here we are using the glove embedding to represent both dataset (because original embedding can't be applied on Android data). Both models were trained on ubuntu dataset only. We will report the performance of both models below.

LSTM

In this experiment we used bi-directional instead of uni-directional LSTM as it yields better performance. Using 150 dimension encoding we are able to achieve following AUC 0.05 score.

	AUC 0.05
Dev	0.568
Test	0.540

CNN

In CNN we adopted the exact same convolutional neural net we implemented in the previous part and were able to achieve a relatively good AUC 0.05 score.

	AUC 0.05
Dev	0.595
Test	0.559

Adversarial domain adaptation

In this part we implement an adversarial domain transfer model according to *Aspect-augmented Adversarial Networks for Domain Adaptation* and *Unsupervised Domain Adaptation by Backpropagation*. In addition to our original encoder-cosine similarity framework, now we additionally connect the encoder to a domain classifier component. The domain classifier component is simply a two-layer ReLU activated fully connected network. When we train our model, the encoder tries to maximize the loss of the domain classifier while minimizing the loss of label prediction; the domain classifier tries to minimize the classification loss (binary cross entropy).
The intuition behind this is that, by trying to confuse the domain classifier, the encoder has to generate encodings that are similar across the domains (i.e. the encoding of an android question is not very different to an ubuntu question), and this will force the encoder to learn common features in both domains.
In the experiment we tried both LSTM and CNN used in direct transfer and then tried to explore additional techniques to boost our performance.

Bootstrapping

The first difficulty we encountered in our experiment is that the labeled android data has far more data points than our ubuntu corpus (partially because some questions appear more than once in labelled android data), which means even we use all our ubuntu corpus the dataset will still be quite unbalanced. To deal with that we sample with replacement from the android corpus to have same number of data points (questions) as our labeled android data.
After bootstrapping we have a balanced dataset. One free bonus of this is that the accuracy reporting of our domain classifier is more informative.

LSTM

We used the same bidirectional LSTM structure as in direct transfer as our encoder, and added a simple two-layer fully connected ReLU classifier. With some parameter tuning, we got a quite good performance boost.

	AUC 0.05
Dev	0.691
Test	0.672

CNN

When we use CNN as our encoder to generate the representation of questions, we adopted the same CNN encoding structure from previous model.
Upon multiple experiments and grid searches, we were able to find the best parameter setting that generated a fairly good result. The adversarial model with CNN encoder exhibited a >10% increase in performance comparing to our direct transfer model. Below we report the performance of our model:

	AUC 0.05
Dev	0.705
Test	0.668

Further Exploration

In order to further improve the performance of our model, we tried many approaches that made intuitive sense to us.

Weighted text and Title

In many cases the title and text of a question could carry different amount of information in representing the overall semantical meaning of that certain question. We wanted our model to capture that different weights of title and text. Therefore we decided to change the mean pooling layer on our CNN model to a fully connected layer of size 2*encoding_size x encoding_size. While the change in structure made intuitive sense to us, the introduction of this fully connected layer also brought dramatic amount of parameters to the network. This make the learning of parameters a lot harder because the parameters in encoder suffered more from gradient vanishing. Upon experiments, we were able to only matched the performance we had in previous part but were not able to achieve improvement in this design.

GRU

The good performance in LSTM models inspired us to use GRU, which is similar to LSTM, but without the complexity LSTM has. It has only an update gate and reset gate. Due to its simplicity it's easier to train and usually have on par performance. In our experiment we simply replace our LSTM encoder by GRU.

	AUC 0.05(Dev)	AUC 0.05(Test)

GRU	0.709	0.675

TF-IDF as new information in embedding

We tried to add tf-idf information in our embeddings. For all words in the document, we add its tf-idf score to its embedding. The intuition behind this is to add “importance” information to words. We did our experiment on CNN but didn’t observe improvement. (LSTM should already be able to capture “importance”) After discussion we think IDF information is usually correlated to word itself (words like ‘a’, ‘the’ always have high df in any documents); and TF information is already emphasized because it’s fed as input multiple times. Therefore adding such information to our embeddings doesn’t give much improvements as the information is already captured.

	AUC 0.05(Dev)	AUC 0.05(Test)
CNN-NewEmb	0.700	0.651

Appendix: Model Performance

Question Retrieval

	Dev				Test			
	MAP	MRR	P@1	P@5	MAP	MRR	P@1	P@5
LSTM	**0.579**	**0.719**	**0.598**	**0.469**	**0.580**	**0.708**	**0.559**	**0.439**
CNN	0.546	0.674	0.556	0.435	0.539	0.675	0.538	0.408

Domain Transfer

	AUC 0.05(Dev)	AUC 0.05(Test)
Tf-Idf Similarity	***0.707***	***0.739***
Direct Transfer BiLSTM	0.568	0.540
Direct Transfer CNN	**0.595**	**0.559**
Adversarial BiLSTM	0.691	0.672
Adversarial CNN	0.705	0.668
Adversarial GRU	**0.709**	**0.675**
Adversarial CNN with NewEmb	0.700	0.651

Link to our source code:
https://github.com/hongxiangqiu/MIT-6.864-Question-Retrieval-and-Transfer-Learning