Question Answering

Raoul Khouri and Aditya Thomas

December 16, 2017

1 Abstract

In Natural Language Processing, Question Answering is an important type of Information Retrieval task. Online question answering forums such as those managed by StackExchange allow users to post a question on a subject with the community responding with suitable answers. In the last few years, there has been an increase in their popularity and thus a corresponding explosion in the number of their users. The absence of an effective automated ability to refer to and reuse answers already available from previously posted questions, means that the community has to repeatedly spend time and energy in answering the same question. In this paper, we first explore a method for finding a related question to a posed question given supervised data from the AskUbuntu forum. We then explore methods to try and transfer the learned model over to the AskAndroid forum where we do not have supervised data.

2 Introduction

Many online question answering forums such as AskUbuntu [2] and AskAndroid [1] have rapidly grown in size. A problem with the growth in size and popularity is that questions are posted which have already been asked and answered, but there is not an automated way of determining this. Typically this information has to be manually annotated by forum users, which is slow and time consuming. This time could be better spent in responding to questions that don't have answers as of yet.

2.1 Related Work

In this paper, we first explore a baseline model to address this task that was discussed in "Denoising Bodies to Titles: Retrieving Similar Questions with Recurrent Convolutional Models" [6]. In the model discussed in the paper, a neural network is used to embed a query sentence into a single vector. These vectors encode a semantic representation of the question. To determine the similarity between one query vector and another, the cosine similarity metric is used. In our experiments we used a convolutional neural network [5] as well as a LSTM [4]. We train and test this model on the AskUbuntu dataset.

We then explore the idea of domain adaptation or transfer learning and for this part of the project, shift our focus to the AskAndroid dataset. We first find standard IR baseline metrics to evaluate against. We then apply the models obtained by training on the AskUbuntu dataset on the AskAndroid dataset, i.e. direct transfer, without any domain adaptation.

We then apply a domain adaptation techniques described in Unsupervised Domain Adaptation by Back propagation[3] and determine metrics on the AskAndroid dataset.

2.2 Experiment Standards

For all the experiments in this paper, unless stated otherwise we used, we limited running the models to only 10 epochs due to the limitation in the computational resources we were able to access. All models in this paper were run on GPU. For all optimization we used the ADAM optimizer.

3 Question Retrieval

We set up the Question Retrieval on the AskUbuntu dataset as a supervised learning task.

3.1 CNN

We encode the title and the body of the question separately and feed them to the CNN. The outputs of the CNN are averaged as discussed in "Denoising Bodies to Titles: Retrieving Similar Questions with Recurrent Convolutional Models" [6] to compute compute the output vector. We compute the output vector for our query question as well as a positive example and 20 negative examples. We then find cosine similarity between the query questions output vector and the others. We then compare the positive question's cosine similarity to that of the negative question with the highest cosine similarity to compute the loss.

We vary hyperparameters such as the learning rate, the size of the convolutional filter and the output size to determine the most suitable model based on MAP, MRR, P1, and P5 scores.

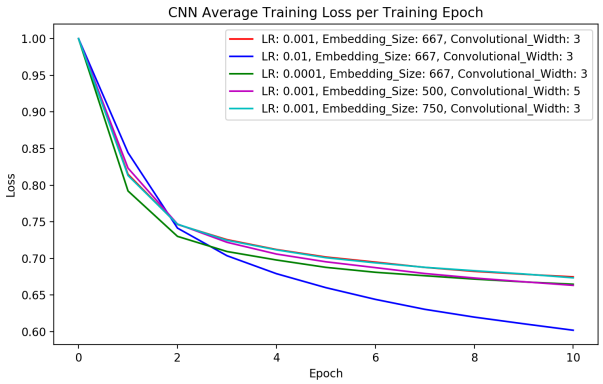

Figure 1: CNN Training Loss per Epoch

Figure 2: CNN MAP (Dashed), MRR (Dotted), P1 (Solid), dev scores for each hyper parameter setup per Epoch

CNN	MAP	MRR	P1	P5
Dev	51.9%	65.1%	53.2%	37.0%
Test	51.7%	67.2%	53.8%	35.3%

From Figure 1 we see that there is a steeper drop in the training loss for a larger learning rate as expected. For different sizes of the convolutional kernel and the output of the CNN only a very slight difference in the training loss can be observed.

From Figure 2, all the models performed similarly. We observe that the model with the largest number of parameters was slightly better. The table above shows our best CNN model's (LR = 0.001, Output Size = 750, Window Size = 3) dev and test scores. Running on GPU it took roughly 1 hour to train each model for 10 epochs.

3.2 LSTM

We implement a bidirectional single layer LSTM. The representations of the question title and body are fed in a similar manner to the LSTM as the CNN model. The hidden states of the LSTM are averaged to compute the output. The loss is computed in a similar way as the CNN model.

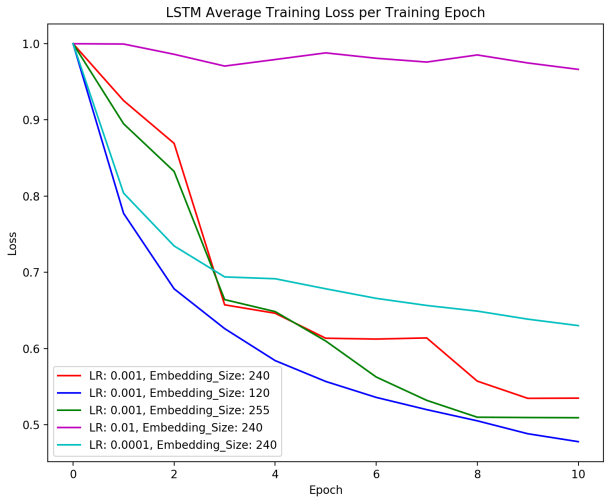

Figure 3: LSTM Training Loss per Epoch

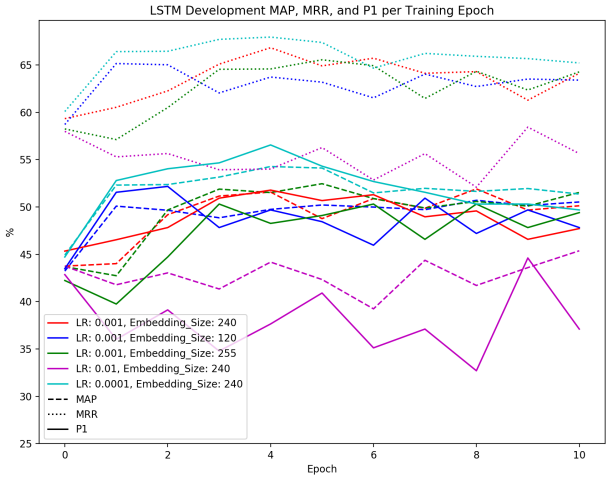

Figure 4: LSTM MAP (Dashed), MRR (Dotted), P1 (Solid), dev scores for each hyper parameter setup per Epoch

LSTM	MAP	MRR	P1	P5
Dev	54.3%	67.9%	56.6%	40.5%
Test	52.7%	67.7%	55.2%	39.1%

From Figure 3 we can see that most of the models converged except for the model with the higher learning rate (0.01). Figure 4 we can see that all the models that converged performed similarly. Our best LSTM model's (LR = 0.0001, Output Size = 240) dev and test scores can be seen in the table above. Running on GPU it took roughly 2 hours to train each model for 10 epochs.

4 Transfer Learning

Though the results for Question Retrieval were good it required supervised data which is unfortunately difficult to acquire. It is not practical to acquire an entire data set for every QA forum on the internet.

In this section we analyze making a Question Retrieval model for the AskAndroid forum, we have test data similar to the AskUbuntu dataset that is hosted on github this data is only used for evaluation. First we analyze common unsupervised methods for building a Question Retrieval model. Then we explore directly taking the models trained using the supervised AskUbuntu data and evaluating on the Android dataset. Finally we explore an adversarial transfer learning method for improving the results.

4.1 Unsupervised Baselines

In order to evaluate the performance and significance of our models we use BM25 and TF-IDF (both implementations were written in house and are included in our github repository). These methods are common techniques used in Information Retrieval and provide meaningful results for Question Retrieval. Our results can be seen in the table below.

	Dev	Test
Model	AUC(0.05)	AUC(0.05)
BM25	50.3%	55.8%
TF-IDF	44.6%	48.4%

4.2 Direct Transfer

The first idea that comes to mind is simply taking the model trained on AskUbuntu and using it on Android. The model should have learned general things about the QA embedding; however, this should not produce as good of a result

as if we had trained the model on the Android data set. In this section, we tune the models trained from supervised learning on AskUbuntu and evaluate on AskAndroid.

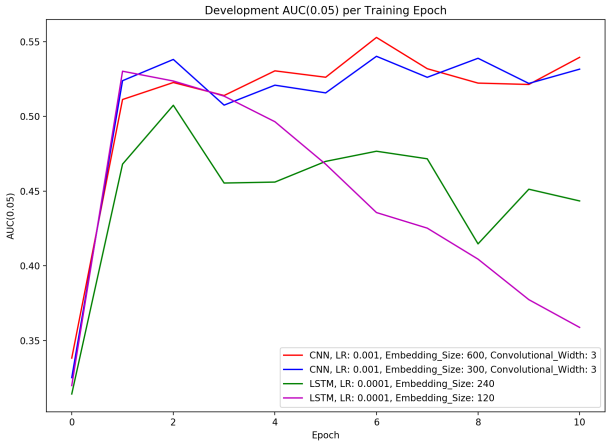

Figure 5: Direct Transfer Dev AUC(0.05) for each hyper parameter setup per Epoch

	Dev	Test
Model	AUC(0.05)	AUC(0.05)
Best CNN	53.2%	52.6%
BEST LSTM	53.0%	51.0%

Here we see that the models perform roughly just as well as the standard IR methods. Our best performing CNN model had a convolutional width of 3 and an output size of 500 with a learning rate of 0.001. Our best performing LSTM model had a embedding size of 240 and LR 0.0001. As shown in Figure 5, we noticed that the LSTM models started to overfit in the direct transfer method; however, the CNN models seemed to be more robust to overfitting.

4.3 Adversarial Transfer

Now we explore an Adversarial method discussed in Unsupervised Domain Adaptation by Back propagation[3] on our problem. This Adversarial method requires an embedding model as before and a classification model that determines if an embedding came from AskUbuntu or Android. The embedding model now has two losses first the original loss of having an embedding vectored that has high cosine similarity for similar questions as well as the negative of the classifiers loss, so Total Loss $=$ Embedding Loss $- \lambda *$ classification loss. Notice that now we must tune this λ. This means that the embedding model will be rewarded for having an embedding vector that has high cosine similarity for similar question but also for making that embedding vector indistinguishable from both datasets. The intuition is that if the embedding vector is indistinguishable to the classifier model then the embedding from AskAndroid will be more similarly embedded than those in AskUbuntu so the transfer will be better.

To test this Adversarial transfer model we fixed the classification model to being a feed forward neural network with a hidden size of 10 and a binary output (0 for Ubuntu 1 for Android) and used Binary Cross Entropy for the loss, with a learning rate of -0.001 (negative since it wants to increase the loss found in total loss).

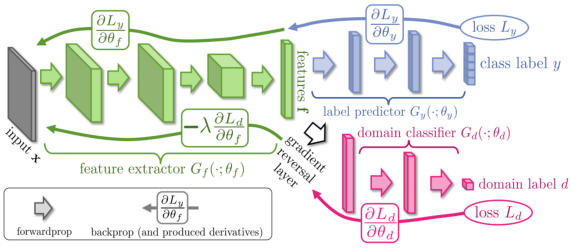

Figure 6: Domain Adaptation by Back propagation Model [3]

	Dev	Test
Model	AUC(0.05)	AUC(0.05)
fixed CNN	61.7%	61.4%
dynamic CNN	73.2%	71.4%
fixed LSTM	74.6%	72.2%
dynamic LSTM	75.3%	73.1%

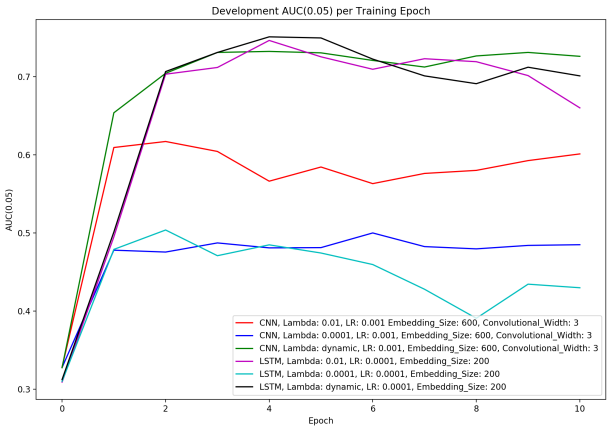

Figure 7: Adversarial Transfer Dev AUC(0.05) for each hyper parameter setup per Epoch

Using this adversarial training regiment we can see that the scores are much better then the scores achieved by the direct transfer and the IR model. We noticed that the model performance is highly dependent on the λ parameter. For the CNN model our best set of hyper-parameters was learning rate of 0.001, output size of 600, and convolutional width of 3, with lambda = 0.01. For the LSTM model our best set of hyper parameters was learning rate 0.0001, output size of 200 and lambda = 0.01.

4.4 Further Work

One of the issues with the explored Adversarial Transfer method proposed in the previous section is that it adds an additional tuning parameter: the adaptation factor λ. λ must be tuned and can cause issues with training the model. In order to suppress noisy signal from the domain classifier at the early stages of the training procedure it must start low. Instead of fixing the adaptation factor λ as done in the previous part, we explore gradually changing λ from 0 to 1 using the following equation proposed in Unsupervised Domain Adaptation by Backpropagation: $\lambda_p = \frac{2}{1+e^{-\gamma * p}} - 1$[3]. In this equation γ is set to 0.5 and p is the training progress (% of way through training). When using the dynamic lambda in Figure 7 we see that the models perform the best when using the dynamic lambda and requires much less tuning.

5 Conclusion and Results

With the growth in size and popularity of online QA forums Question Retrieval is a very important problem to solve. In this paper we have looked at a supervised learning method discussed in "Denoising Bodies to Titles: Retrieving Similar Questions with Recurrent Convolutional Models" [6] that performs better that the standard IR methods; unfortunately, it requires significant amount of difficult-to-acquire training data. So looked into the Transfer methods and found that the Adversarial method discussed in Unsupervised Domain Adaptation by Backpropagation [3] yielded the best results improving our AUC(0.05) by about 10%. These findings are very important to the field of NLP and can greatly improve response times in online forums.

6 Individual Contributions

For the first part of the project Raoul worked on the CNN model and data loader, while Aditya worked on the LSTM model. For the second part Raoul spearheaded the work on the unsupervised and adversarial transfer model. Aditya worked on the direct transfer model. Both Raoul and Aditya contributed GPU resources for running all the experiments in this paper.

7 Code

The code for the project was written in python and uses the pytorch library for Neural Network computation. The code for this project can be found at https://github.com/Keyrat06/Question_Answering

References

[1] Stackexchange - ask android. https://android.stackexchange.com/, 2017. [Online; accessed Dec 2017].

[2] Stackexchange - ask ubuntu. https://askubuntu.com/, 2017. [Online; accessed Dec 2017].

[3] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation, 2015.

[4] Sepp Hochreiter and Jrgen Schmidhuber. Long short-term memory. 9:1735–80, 12 1997.

[5] Yann Lecun and Yoshua Bengio. *Convolutional networks for images, speech, and time-series*. MIT Press, 1995.

[6] Tao Lei, Hrishikesh Joshi, Regina Barzilay, Tommi S. Jaakkola, Kateryna Tymoshenko, Alessandro Moschitti, and Lluís Màrquez i Villodre. Denoising bodies to titles: Retrieving similar questions with recurrent convolutional models. *CoRR*, abs/1512.05726, 2015.