Semi-Supervised and Unsupervised Question Retrieval System Using Domain Adaptation

Maryann Gong, Nick Matthews
6.864
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology

***Abstract*—In large, open-access question-answer forums, identifying similar questions is both important and difficult to achieve manually. In this work, we explore several techniques for question retrieval in both labeled and unlabeled domains. We utilize both semi-supervised and unsupervised techniques and implement neural network architectures that are trained and evaluated on the same domain. We then also explore methods for adapting these models trained on one source domain to a different target domain, including adversarial networks. Our model implementations are publicly available on Github[1].**

I. Introduction

Developers of all levels, whether beginners or specialized professionals with a decade of experience, frequently use question-answer forums. These forums, where users can find answers to frequently asked questions, are a great way to speed up the development cycle and to avoid many unnecessary hours of debugging; they are useful in professional, research, and school settings. For example, at MIT, many courses make liberal use of Piazza where students can ask each other and instructors questions about the technical material.

It is relatively common for users to unknowingly post a question that has been answered before. In the scope of one class, students can oftentimes identify a repeated question and point to the previous answer. However, on larger forums open to the public, the number of questions is much larger and it is much less feasible for these repeated questions to be resolved quickly.

Finding similar or redundant questions is much harder than a simple regex match. Similar questions can have a slightly different vocabulary and phrasing. One question can have much more extraneous detail than another, while the heart of the two questions is the same. Conversely, two questions that are very different could also share much of the same vocabulary. This noise or variance makes the problem of retrieving similar questions difficult.

In this project, we implement a question retrieval system that identifies similar questions given a subset of candidate questions. It is feasible to find a similar question by searching the entire corpus of questions instead of using a narrowed-down subset of questions, but this method is not computationally practical. Our project focuses on two main types of question retrieval systems: one is a semi-supervised question retrieval system and the other is an unsupervised question retrieval system using domain adaptation.

[1]https://github.com/maryanngong/QuestionRetrievalSystem

For the semi-supervised system, we worked with a corpus of questions from AskUbuntu. In our test and development AskUbuntu sets we have both positive and negative labels. However, in our training set, we only have positive labels while the negative samples and degree of difference are unknown, so we use 20 randomly sampled questions from the corpus as "negative" examples, under the assumption that the probability that the questions are actually positive (very similar) is negligibly low.

In our unsupervised setting, we aim to retrieve similar questions for a new dataset from the Android stack exchange. This data set has no training labels at all, so we use various transfer learning techniques, using the semi-labeled training examples from the AskUbuntu data set, in order to label similar questions in our target domain of the Android data set. In domain adaptation we use an adversarial training procedure to force our encoder to learn domain-invariant representations that work for both Ubuntu and Android questions. We also experiment with Wasserstein GANs as an exploratory demonstration of their applicability in NLP problems.

II. Related Work

There is a sizable amount of work in the field of similar question retrieval. Past research have utilized a variety of techniques, including deep learning and other methods. In [1], researchers represented their questions as bags of words and applied a CNN. We explore the use of CNNs and LSTMs as our encoder, as mentioned in [2]. To quantify similarity between our question encodings, we use the cosine similarity metric, as described in [2].

In terms of unsupervised domain adaptation for question retrieval systems, there are a variety of works in the field of domain adaptation. Some methods attempt to gradually train their network to move from the source to the target domain by gradually feeding in more target domain samples while training [3]. We directly apply the adversarial network model described in [4], where they train both an encoder and discriminator that are optimized to generate encodings that can be used to identify similar questions but are also able to generalize from the labeled to the target unlabeled domain.

We explore enhancements to this adversarial network for domain adaptation. In [5], researchers apply domain adaptation techniques to voice activity detection. In their work, they aim to improve domain adaptation by pretraining their network to minimize reconstruction loss on the target domain (without labels). In our work, we utilize this idea of minimizing reconstruction loss by incoporating an autoencoder component to our encoder model. Applying a reconstruction loss in a similar NLP setting is a part of recent work by Zhang et al. [6]. We also explore the application of modified generative adversarial networks (GANs) to our question retrieval domain adaptation problem. Researchers in [7], utilize a GAN for domain adaptation in the pixel image space. Their architecture differs from a more traditional GAN in that it is not generating the target distribution by transforming noise, but is instead transforming the source domain distribution to resemble a sample from the target distribution; this GAN-like training procedure works by adding random noise to the input - in our case, sequences of word embeddings. Furthermore, we substitute their GAN architecture for the relatively new Wasserstein GAN (WGAN) that have been shown to empirically solve many of the problems and challenges of training regular GANs. [8].

III. Methods

A. Data Pre-Processing

For both in-domain and out-of-domain question retrieval, our data consisted of questions with both a title and body that each has variable sequence length. In order to reduce computation time and memory usage for our models, we truncated the question bodies to a maximum sequence length of 100 words. Next, in order to process our sample questions in more efficient batches, rather than one by one, we padded the question titles to the maximum length title in the batch and the question bodies to the maximum length body. Also, for the out-of-domain question retrieval task, we chose to convert the entire target domain (Android corpus) to lowercase and use specifically lowercase word embeddings since our source domain (AskUbuntu corpus) contains only lowercase tokens.

B. In-Domain Question Retrieval

For the in-domain question retrieval task, we are given a list of candidate questions that include questions both similar and dissimilar to the given query question. Our model uses the same architecture as Lei, et al. in [2]. Specifically, we first encode all of the input tokens using word embeddings, followed by a further encoding step that reduces each variable-length sequence of word embeddings to a fixed-length vector. For our embedding layer, we used pre-trained 200-dimensional uncased word embeddings that were trained on texts from StackExchange and Wikipedia. We encode each question title and body separately and then use the average of the two encodings for the final whole question encoding. Finally, we rank the candidates by measuring the distance between the encoded given query and the encoded candidate questions using cosine similarity. To train these networks, we utilize multi-margin loss.

For our encoder, we compare two architectures: 1) a simple CNN with one convolution layer and a mean pooling layer and 2) a bi-directional LSTM also with a mean pooling layer. Our simple CNN consists of a single one-dimensional convolutional layer across the sequence of words with kernel size of 3, capturing implicit trigram features and mapping each kernel to a 512-dimensional hidden state. Each dimension of the embedding vector is treated as an input channel to the convolutional layer. We then applied mean pooling across the sequence to yield a 512-dimension vector representing the entire sequence. Note: since titles and bodies are separately processed up until this point, we take the average of their values before measuring the cosine similarity. This works well because titles and bodies both contribute important information despite the bodies being much longer.

For our LSTM, we used a single bidirectional LSTM with one layer with 300 hidden units with a dropout layer with dropout probability 0.2. After feeding in the sequence of word embeddings, we mean pool over the resulting sequence of 300-dimensional hidden states.

For both our networks, we varied the hidden dimension size, added dropout between 0-0.5, and adjusted the learning rate, but found our best performance on validation set with 512 hidden units and no dropout for the CNN, 300 hidden units with 0.2 dropout for the LSTM, and learning rate of 0.001 for both, and multi-margin loss margin size of 0.5.

C. Out-of-Domain Question Retrieval

In the out-of-domain (domain adaptation, or transfer) setting, we are given the same labeled training data as the in-domain setting (Part 1), but evaluated on questions from a new domain. We consider a number of increasingly more sophisticated models to solve the domain-transfer task.

1) Baseline 1: TFIDF: For our simplest baseline we compute TF-IDF vectors for the entire Android corpus. This corpus includes text samples that show up in the development and test sets; accordingly, the performance is abnormally high and our results for TF-IDF are not directly comparable with the other models.

2) Baseline 2: Direct Transfer: For evaluating our domain adaptation techniques we chose a more representative baseline: directly transferring our model from the in-domain setting. We train the same type of encoder models from Part 1 on the original source domain using labeled groups of query and candidate questions, but this time we use the lowercased GloVe word embeddings, that are not specific to any one of our two domains. We also adjust the hidden size for the LSTM to be 240 and reduce the multi-margin loss margin size to 0.2 to increase development set performance. Because the target domain is also a similarly-structured, technical question forum, there will be some successful transfer, but overall non-negligible differences in vocabulary, question types, and style should somewhat hinder this model.

3) Domain Adaptation (DA): In order to better transfer what the encoder learns in the source domain to the target domain we force it to learn domain-invariant representations (encodings) while training on labeled source-domain data in the hope that it will be able to more meaningfully encode the target domain questions. We do this by introducing an adversarial discriminator and train it to distinguish between encodings of the source domain and encodings of the target domain [4]. For our discriminator, we use a very simple feed-forward network with 100 hidden units and two fully connected layers. During our training procedure we alternate between a batch of training data (labeled source domain data) and a second batch in which the encoder encodes 20 samples from both source and target corpora which are randomized and passed to the discriminator. Let Enc represent our encoder, D represent our discriminator, and x represent our embedded sequence of words. Our new loss is then:

$$L_\theta = L_1 - \lambda_{adv} \cdot L_2$$

$$L_1 = \text{Multi-Margin-Loss}(Enc(x))$$

$$L_2 = \text{Binary-Cross-Entropy-Loss}(D(x))$$

By subtracting the discriminator loss from our original multi-margin loss and using a negative learning rate when optimizing our discriminator model, our encoder learns domain-invariant representations that better fool the discriminator. For our encoder, we use the same exact CNN structure as in the direct transfer method. We tune the hyper-parameter λ_{adv} to a best-performing value of 0.001.

4) Domain Adaptation with Reconstruction Loss: Our first expansion on our domain adaptation implementation (from Part 2) is to add the reconstruction mean square error to our loss function. One possible shortcoming of our original adversarial domain adaptation model described in the previous section is that by training the encoder to create domain-invariant question encodings, the model can create very sparse encodings. As a result, these sparse encodings lose some of the original signal information and lead to worse question retrieval performance. We mitigate this by adding a mirroring decoder to our encoder model creating an autoencoder. We use the same encoder structure as before to create our question encodings, but we also decode the same encodings and compute the mean squared error between the decoded output and the original word-embedded sequence. This mean squared error is added to our overall loss. Let the function Dec represent the decoder and Enc represent our encoder. Our reconstruction loss is

$$L_3 = \text{Mean-Squared-Error}(Dec(Enc(x)), x)$$

Our overall loss becomes:

$$L_\theta = L_1 - \lambda_{adv} \cdot L_2 + \lambda_{reconstr} \cdot L_3$$

In order to directly compare to the direct transfer and regular domain adaptation models, we keep the same CNN encoding structure and $\lambda_{adv} = 0.001$, but then tune the new hyper-parameter $\lambda_{reconstr}$ and find the best value to be at 1.

5) DA via GAN Training: Since the overall goal is to learn how to encode the target domain's input for proper similarity measurement, we hypothesized that something like a generative adversarial training schema would work well. Specifically, we introduce a third model (the transformer) tasked with transforming the sequences of word embeddings in the source domain into sequences of word embeddings that look like they came from the target domain. This transformer model is analogous to the generator model of a traditional GAN. In order to stably train our transformer, we treat it like a generator in a GAN setting by adding Gaussian noise to the input and training it adversarially against the discriminator. The discriminator tries to distinguish the word-embedded target domain data from the transformed source domain data produced by the transformer model to mimic the target domain. In the second half of each epoch, we train the original encoder model with alternating batches of source domain input with source domain labels and transformed source domain input with source domain labels. We believe this facilitates better training for the encoder model particularly early on when the input it receives from the transformer is sub-optimal. The transformer model, after applying the noise, uses a bidirectional LSTM layer with 200 hidden units, followed by a fully connected linear layer to create an output of the same size as the input but appearing as if from the target domain. For our discriminator we also use a single bidirectional LSTM layer with 200 hidden units, followed by a fully-connected layer connected to a single output unit. The encoder uses the same CNN architecture outlined in earlier sections.
Training this architecture with the standard GAN loss calculations [9] proved difficult and yielded initial poor performance. We saw dramatic improvements by instead using the Wasserstein GAN (WGAN) architecture [8]. For the WGAN setup, we use the RMSprop optimizer instead of Adam optimizer. This architecture also allowed us to train the Discriminator model to optimality at the onset of our training procedure, unlike in a traditional GAN setup. The new losses for WGAN are:

$$L_D = D(x_t) - D(T(x_s))$$

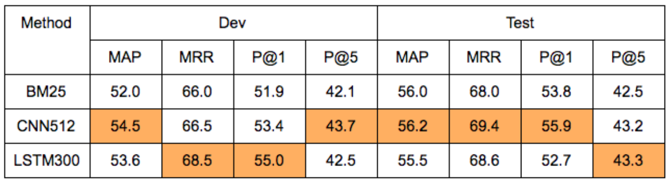
$$L_T = D(T(x_s))$$

Where D is the discriminator, T is the transformer, x_s is the source data and x_t is the target domain data. The encoder losses are the same as before, but applied once for the transformed source input and once for the untransformed source input. Additionally we need the weights of our discriminator to lie in a compact space to satisfy the Lipschitz constraints that the WGAN relies on to construct a continuously differentiable loss function that is reliable. We clamp the parameters of the discriminator network to be in the range $[-0.01, 0.01]$. Through these modifications, we bypassed common problems with exploding and vanishing gradients.

IV. Results

A. In-Domain Question Retrieval

To evaluate the performance of our model, we use 4 metrics–Mean Average Precision (MAP), Mean Reciprocal Rank (MRR), Precision at 1 (P@1), and Precision at 5 (P@5). Below in Figure 1 we show the results of our best performing CNN and LSTM on our 4 metrics on the development and test sets. The highest score for each metric is highlighted in orange. As you can see, our CNN with 512 hidden units, outperformed the baseline on every measure across both sets, whereas the LSTM with 300 hidden units beat the CNN on certain metrics. Overall, we get better performance with our simple CNN compared to our LSTM architecture.



Method	Dev				Test			
	MAP	MRR	P@1	P@5	MAP	MRR	P@1	P@5
BM25	52.0	66.0	51.9	42.1	56.0	68.0	53.8	42.5
CNN512	54.5	66.5	53.4	43.7	56.2	69.4	55.9	43.2
LSTM300	53.6	68.5	55.0	42.5	55.5	68.6	52.7	43.3

Fig. 1: MAP, MRR, P@1, P@5 for In-Domain Question Retrieval Models

B. Domain Adaptation

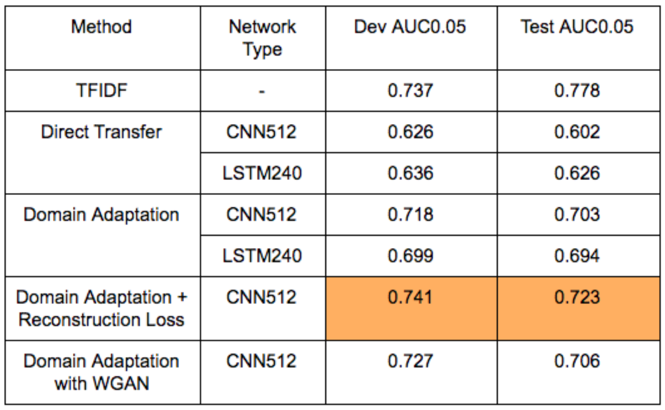
We evaluated our domain adaptation models using the $\text{AUC}_{0.05}$ metric. This metric translates to the model's likelihood of assigning a higher score to a randomly selected positive (similar match) sample when compared to a randomly selected negative sample constrained to when the false positive ratio is less than 0.05. We use this metric as opposed to classic AUC because the data has a strong class imbalance, with many more negative samples than positive matches.

When moving to question retrieval on the unlabeled Android corpus, we first evaluated our direct transfer model, which was optimized on the labeled AskUbuntu corpus and directly applied to the Android corpus with no modification. With this easy transfer, we established a baseline of about 0.626 on the development set. This was a much more reasonable baseline for comparison than the performance of the TFIDF model, which contained possible pollution between the test and development set, which could inflate its performance.

Next our initial domain adaptation model, utilizing a simple feed forward adversarial discriminator, yielded drastic improvements over the direct transfer baseline from 0.626 to 0.718 (14.7% increase) on the development set. This adversarial procedure helped force the encoder to encode the target (Android) domain questions similarly to the source domain. From there we could follow the same cosine similarity scoring procedure to retrieve similar questions with much better performance than our naive direct transfer model.

Next, the enhancement of our domain adaptation model by adding reconstruction error to our loss modestly improved performance from 0.718 to 0.741 (additional 3.2% increase). This reconstruction loss helped prevent the encoder from making the question encodings too sparse to fool the discriminator. Enforcing penalty for loss of resolution in reconstruction ensures that enough information about the question is maintained throughout the encoding process.

Finally, our alternative WGAN for domain adaptation also yielded strong improvements over the baseline direct transfer with $\text{AUC}_{0.05}$ of 0.727 and achieved moderate improvement over our basic adversarial domain adaptation network. The generator, or in our case transformer, model was able to successfully transform source domain questions into fake target domain questions, simulating having labeled data in the target domain. Our total results are displayed in Figure 2. Our best performing implementation is highlighted in orange.



Method	Network Type	Dev AUC0.05	Test AUC0.05
TFIDF	-	0.737	0.778
Direct Transfer	CNN512	0.626	0.602
	LSTM240	0.636	0.626
Domain Adaptation	CNN512	0.718	0.703
	LSTM240	0.699	0.694
Domain Adaptation + Reconstruction Loss	CNN512	0.741	0.723
Domain Adaptation with WGAN	CNN512	0.727	0.706

Fig. 2: AUC0.05 for Out-of-Domain Question Retrieval Models

V. Conclusion

These experiments demonstrated that neural network encoders are powerful tools for building question retrieval models. When working with labeled data, using simple single-layer CNN and LSTM encoders improves question retrieval performance over the BM25 ranking function baseline. These single layer neural networks also did not use any non-linear activation layers and still yielded comparable performances. These same encoding models are also powerful tools in the unsupervised setting. Without modification, both the CNN and LSTM models perform with scores of more than 0.62 $\text{AUC}_{0.05}$ on the unseen target domain. By incorporating domain adaptation with a simple adversarial discriminator, we were able to boost our performance with the CNN model by 14.7% and LSTM model by 9.91% from the direct transfer baselines. Further enhancements brought even larger improvements. By incorporating reconstruction error into our domain adaptation model, we improved our CNN performance by 18.4%. Our experiments using a WGAN for domain adaptation led to comparable improvements of 16.1%.

For all these model architectures, we used the same simple single-layer encoding network. We expect further improvement with more complex encoders and more fine-grained hyper-parameter tuning. The WGAN model performed well, but the project's limited scope did not afford the opportunity to play extensively with many tweaks commonly used in the adversarial setting e.g. normalization techniques, introducing more noise at multiple stages in training, and extensive hyper-parameter tuning. Furthermore, in these experiments we use pre-trained word embeddings that are not specific to any of our source or domain corpora. We suspect that training our own word embeddings specifically on these types of question-answer forum corpora should also help improve performance. The domain adaptation architectures corroborate success others have had applying adversarial models to NLP tasks [6]. We additionally demonstrated that the Wasserstein GAN works well in NLP tasks.

VI. Acknowledgements

Thank you to Professor Regina Barzilay, Tao Lei, and the rest of the MIT NLP course staff for all their help and guidance through this research project. Also thank you to Alexir Pan and Agustinus Kristiadi for their instructive articles about Wasserstein GANs.

References

[1] G. Zhou, Y. Liu, F. Liu, D. Zeng, and J. Zhao, "Improving question retrieval in community question answering using world knowledge." in *IJCAI*, vol. 13, 2013, pp. 2239–2245.

[2] T. Lei, H. Joshi, R. Barzilay, T. Jaakkola, K. Tymoshenko, A. Moschitti, and L. Marquez, "Semi-supervised question retrieval with gated convolutions," *arXiv preprint arXiv:1512.05726*, 2015.

[3] R. Gopalan, R. Li, and R. Chellappa, "Domain adaptation for object recognition: An unsupervised approach," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 999–1006.

[4] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *International Conference on Machine Learning*, 2015, pp. 1180–1189.

[5] X.-L. Zhang, "Unsupervised domain adaptation for deep neural network based voice activity detection," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 6864–6868.

[6] Y. Zhang, R. Barzilay, and T. Jaakkola, "Aspect-augmented adversarial networks for domain adaptation," *arXiv preprint arXiv:1701.00188*, 2017.

[7] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," *arXiv preprint arXiv:1612.05424*, 2016.

[8] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[9] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.