6.864, Fall 2017 - Neural Approaches to Question Retrieval

Codebase: https://github.com/shashank-srikant/6.864_term_project

Vadim Smolyakov*
CSAIL, MIT
vss@csail.mit.edu

Shashank Srikant*
CSAIL, MIT
shash@mit.edu

ABSTRACT

Content on the internet grows at an exponential rate. Given this growth, finding relevant information accurately becomes a critical task for the NLP community to address. More so, with this rapid growth, curating labeled datasets to build models for the wide variety of content available on the internet has become extremely time and resource intensive. In this work, we explore whether neural models are able to successfully model content similarity and retrieval tasks, and whether they can transfer knowledge from one domain, where supervised labels are available, to a domain with no available labels. Specifically, we explore content similarity in online discussion forums, where we explore the following questions - a. how effectively can neural approaches model question-answer retrieval tasks, which is, given a question and answer pair present on an online discussion forum, how effectively can neural approaches find similar pairs on that forum. b. Given a model of question-answer similarity in one online discussion forum, how effective are neural approaches in transferring that knowledge to a new, loosely related domain. In this work, we explore a baseline approach of modeling question similarity tasks on the popular Stack Exchange online community *AskUbuntu*. We show how neural architectures like LSTMs and CNNs outperform traditional approaches in information retrieval. Additionally, and importantly, we explore the problem of transferring these models to detect question-answer similarity on *Android Stack Exchange*, a similar yet different online discussion community where members troubleshoot Android related problems. We show how neural domain adaptation techniques successfully beat baseline IR techniques and direct neural transfer techniques. We also discuss some limitations and challenges in using such architectures.

1. INTRODUCTION

The problem of text similarity, and specifically, similarity of short queries or answers found on the internet, have been central to the modern NLP community. With the explosion in content on the internet, a lack of robust tools to find similar content has the risk of creating further similar and redundant content, which only exacerbates the original problem.

Another relevant and pressing concern which such an explosion and variety of content has introduced is the increased cost of building predictive NLP, NLU models which cater to such content. The variety in content requires building a model from scratch, irrespective of how closely related the content may have been to a previously built model. For instance, if we were to model the reviews written for movies, we would have to reinvest effort and time in modeling reviews written for another domain, say, food or hotels. In spite there being conceptual, semantic similarities between the tasks of reviewing movies and reviewing food, models have to be created anew. And each such modeling exercise demands a large repository of curated, preferably labeled data, which most often is not feasible to put together. The pressing challenge such a variety of content has created is to be able to learn with minimal supervision, and from loosely related datasets.

In this work, we investigate two problems - one, to model question-similarity tasks using state of the art techniques in neural modeling and two, transfer those models to a loosely related, yet equally rich domain under the constraint of having no supervised information on that new domain. Specifically, we explore the problem of finding similar question-answer pairs on the popular *Ask Ubuntu Stack Exchange*.[1] This discussion forum focuses on troubleshooting queries on Ubuntu, the popular open-source operating system. In addition to learning models to find relevant question-answer pairs on this forum, we also use this domain to transfer knowledge onto modeling question-answer similarity in *Android Stack Exchange*.[2] This is an online community, similar in setup and structure to *Ask Ubuntu*, wherein short queries and corresponding answers on Android-related troubleshooting are present.

In this work, we investigate how state of the art neural architectures for NLP applications perform on the task of question similarity. Traditionally, text similarity tasks were solved using information retrieval techniques like building an indexer and applying string search over the indexed corpus. A lot of variations to this general strategy have been explored over the years. We study how more recent techniques like RNNs and CNNs, which have shown to successfully model a variety of NLP tasks, solve the question retrieval problem. For the domain adaptation task, we explore adversarial training techniques [9; 15] as well as the performance on different architectures like Doc2Vec [14] and Siamese neural networks [16].

This study demonstrates the relevance of neural techniques

*Author order decided by tossing a fair coin.

[1] https://askubuntu.com

[2] https://android.stackexchange.com

Scenario	Setting	Nature of Data	Learning Paradigm	Main Concepts
$\mathcal{D}_S = \mathcal{D}_T$, $\mathcal{T}_S = \mathcal{T}_T$	Traditional Machine learning	Labelled data in source domain(s) and unlabeled data in target domain	Source and target domains are exactly the same	Learn models on training set and test on future unseen data
$\mathcal{D}_S \neq \mathcal{D}_T$, $\mathcal{T}_S = \mathcal{T}_T$	Transductive Transfer Learning	Labelled data in source domain(s) and unlabeled data from $\mathcal{P}(X_S) \neq \mathcal{P}(X_T)$	**Single source domain adaptation**	Learning common shared representation; instance weighing, parameter transfer
			Multi-source adaptation	Classifier combination; efficient combination of information from multiple sources; Feature representation
No conditions on $\mathcal{D}_S$, $\mathcal{D}_T$, but $\mathcal{T}_S \neq \mathcal{T}_T$	Inductive Transfer Learning	Unlabeled data in source domain(s) and labeled data in target domain	Self-taught learning	Extracts higher level representations from unlabeled auxiliary data to learn instance-to-label mapping with labeled target instances
		Labeled data is available in all domains	Multi-task learning	Simultaneously learns multiple tasks within (or across) domain(s) by exploiting the common feature subspace shared across the tasks
$\mathcal{D}_S \neq \mathcal{D}_T$ $\mathcal{T}_S \neq \mathcal{T}_T$	Kim et al., 2015 [13]	Labeled data in source and target domains	Transfer learning with disparate label set	Disparate fine grained label sets across domains, however, same coarse grained labels set can be invoked across domains

Table 1: A brief summary of different transfer learning approaches. $\mathcal{D}$ is the domain and $\mathcal{T}$ is the task. The boldfaced cell highlights the approach explored in this work. This summary is based on the work detailed in the survey paper by Pan et. al.[18]

in a critical NLP task like question-answer retrieval. We demonstrate how using neural approaches, one can outperform traditional information retrieval techniques while not having to invest heavily in engineering the right features to get to such performance. Additionally, we show the successful transfer of domain knowledge from one domain to another using state of the art neural transfer techniques.

This work is organized into the following sections - Section 2 discusses related work in this field. Sections 3, 4, 5 discuss various techniques to model in-domain and domain adaptation tasks for the given problem. Section 6 discusses the experiment setup and the results. Section 7 concludes our work.

2. RELATED WORK

Given the growing popularity of community QA forums, question retrieval has emerged as an important area of research. Recent work has also gone beyond word-based methods to represent this task. For instance, Feng et. al. [7] learn word embeddings using category-based metadata information for questions. They define each question as a distribution which generates each word (embedding) independently, and subsequently use a Fisher kernel to assess question similarities. Dos Santos et. al. [6] propose an approach which combines a convolutional neural network (CNN) and a bag of words representation for comparing questions. In contrast to [7], Lei et. al. [15] treat each question as a word sequence as opposed to a bag of words, and apply recurrent CNNs instead of traditional CNNs. We base our work on [15]. We apply different off-the-shelf deep learning models to encode a meaningful embedding and define similarity over these embeddings.

Domain adaptation has been an active area of research in NLP. Table 1 summarizes different settings of transfer learning [18]. Our work explores techniques in unsupervised domain adaptation, where the source and target domains ($\mathcal{D}$) are different, but share a common task (T) to achieve. The summary helps in understanding where our approach lies in this spectrum of techniques.

We study in this work the scenario where the domains vary while the tasks remain the same. This is referred to as transductive transfer learning. This is the most extensively studied settings in the transfer learning literature and can be broadly categorized as single and multi-source adaptation. Single source adaptation[4; 1; 5] primarily aims at minimizing the divergence between the source and target domains either at instance or feature levels. The general idea being identifying a suitable low dimensional space where transformed source and target domains data follow similar distributions and hence, a standard supervised learning algorithm can be trained.

Multiple methods perform unsupervised domain adaptation by matching the feature distributions in the source and the target domains. Some approaches perform this by reweighing or selecting samples from the source domain [3; 12; 10], while others seek an explicit feature space transformation that would map source distribution into the target ones [17; 11; 2]. An important aspect of the distribution matching approach is the way the (dis)similarity between distributions is measured. Here, one popular choice is matching the

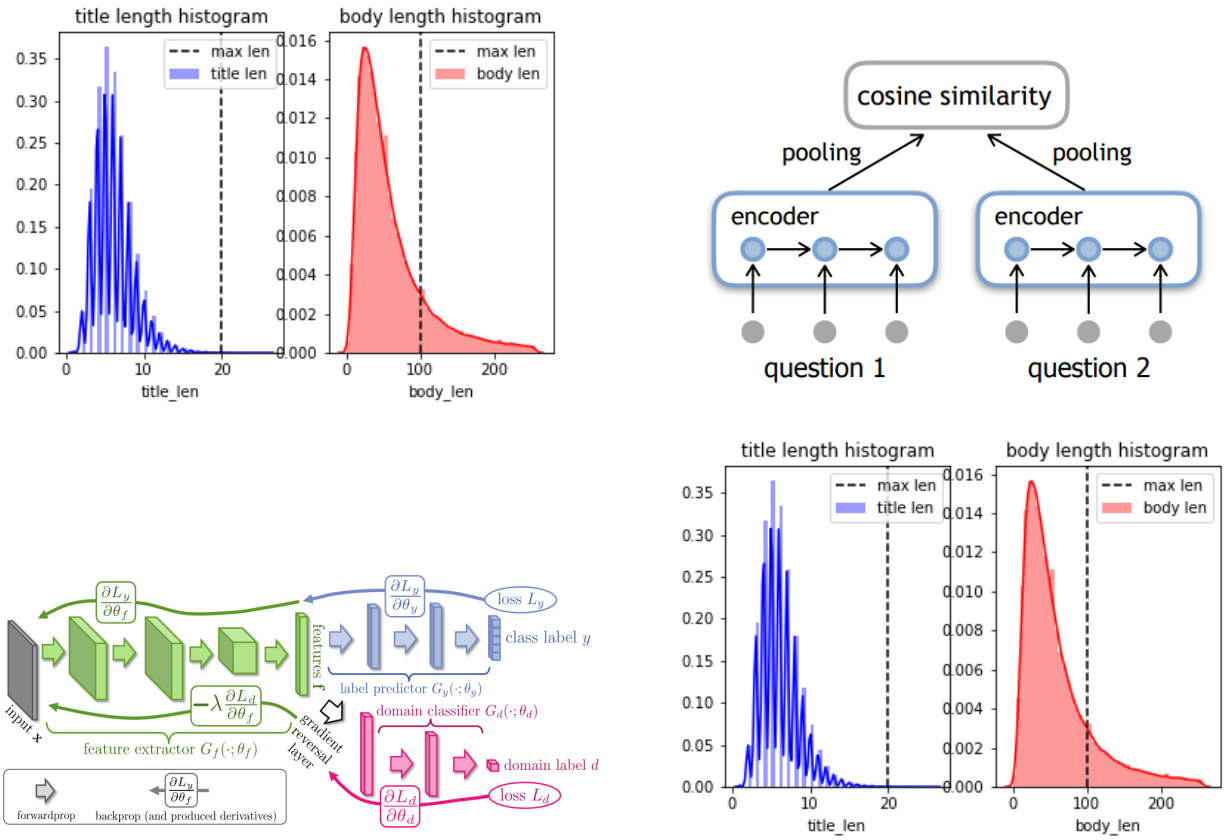

Figure 1: Summary of the *Ask Ubuntu* dataset

distribution means in the kernel-reproducing Hilbert space [3; 12], whereas [10; 8] map the principal axes associated with each of the distributions. The approach explored in this work is based on [9] and also attempts to match feature space distributions. However, this is accomplished by modifying the feature representation itself rather than by reweighing or geometric transformation. Also, the method used to measure the disparity between distributions based on their separability is implicitly different and is done by a deep discriminatively-trained classifier in our work as.

3. IN-DOMAIN QUESTION SIMILARITY

We first explore the task of question similarity by training a domain-specific classifier. The domain we have explored in this work involves the questions and answers asked on *Ask Ubuntu*. The dataset for this task was curated by [15].

	source($\mathcal{D}_S$)	target($\mathcal{D}_T$)
Description	*AskUbuntu*	*Android SE*
# of sentences	40	50
# of stop words	50	90
mean word length - title	299	300
mean word length - body	299	300

Table 2: Summary statistics of our dataset

3.1 Dataset

We describe here the dataset that we base our work on.

- The *Ask Ubuntu* dataset consists of questions and answers related to problems people face in Ubuntu, the Linux operating system
- Each query has two components - a title and main body. The average title length is XX words and the average body length is YY words long. Figure 6.4 shows the distribution of word lengths
- Each query is associated with a few ($<$ 10) positive (similar) questions and 100 randomly drawn questions from the corpus which serve as negative (dissimilar) questions.
- The dataset contained 110K such queries to train on.
- Each query-question pair is also associated with its BM25scores provided by Apache Lucene. This served as a baseline in our experiments.

3.2 Model

We model the task of in-domain question similarity as follows:

- We define a word-level deep neural network (DNN). The network takes a sequence of indices corresponding to word embeddings. We average the title and the body of each query, and use the averaged representation to compute similarity.
- We use word embeddings pre-trained on source domain and the Wikipedia.

- To train the classifier to find similar and dissimilar questions for a given query, we associate each query with two sets of texts - the set of similar questions as marked by users on *Ask Ubuntu*, and 100 randomly sampled questions from the corpus which serve as dissimilar pairs. The user-defined questions are used only during training. Since user-marked similar questions are not comprehensive and serve only as a weak proxy, the system is tested on expert-marked similar questions.
- We use cosine similarity to rank similar questions by computing the cosine of the angle between the query encoding and its associated candidate question.
- We use a max-margin framework for learning parameters θ of our network. Specifically, in a context of a particular training example where q_i is paired with p_i^+, we minimize the max-margin loss $L(\theta)$ defined as

$$\max_{p^- \in Q(q_i)} \{s(q_i, p^-; \theta) - s(q_i, p_i^+; \theta) + \delta(p, p_i^+)\} \quad (1)$$

  where $\delta(.,.)$ denotes a non-negative margin. We set $\delta(p, p_i^+)$ to be a small constant when $p \neq p_i^+$ and 0 otherwise. The parameters θ can be optimized through sub-gradients $\partial L / \partial \theta$ aggregated over small batches of the training instances [15].
- The neural network essentially serves as an encoder to generate a representation in a low-dimension space.
- We compare performance on two neural network architectures: Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN) and a Convolutional Neural Network (CNN).
- Similarity is evaluated on standard information retrieval metrics like *Precision@1*, *Precision@5*, *Mean Average Precision*, and *Mean Reciprocal Rank*. The inputs to each of these metrics is a ranked list of scores. We pass in the cosine similarity scores obtained between the query and each of the positive and negative questions associated with that query, as our ranked list. The BM25scores for each of these pairs is used as a baseline similarity score and their evaluations on the metrics mentioned above are considered the baseline performance indicators for this approach.

3.3 CNN

We describe the CNN architecture implemented in this work. We detail the inputs to the different layers of the network, which should help illustrate the architecture. We define a CNN with the following layers -

- **Input vector X** of dimensions: (**B**, **W**) (**B**: batch size, **W**: max # of words in sentence in batch)
- **Embedding layer** Pass **X** through embeddings layer. Resulting dimension: (**B**, **W**, **D**) (**D**: # of embeddings per word)
- **Convolution, ReLU** Pass the above input through a convolution layer and then through ReLU. Resulting dimension: (**B**, **K**, **Y**)(**K**: Depth of filter, i.e. # of filters of given height, **Y**: # of values sliding the filter over the input)
- **Pooling** Pass through mean pooling. Resulting dimension: (**B**, **K**)
- The resulting encoded embedding of dimension (**B**, **K**) for **B** words in a given batch.

3.4 LSTM

TODO: describe the architecture, add a figure for both CNN and LSTM We describe the CNN architecture implemented in this work. We detail the inputs to the different layers of the network, which should help illustrate the architecture. We define a CNN with the following layers -

- **Input vector X** Dimensions: (**B**, **W**) (**B**: batch size, **W**: max # of words in sentence in batch)
- **Embedding layer** Pass **X** through embeddings layer. Resulting dimension: (**B**, **W**, **D**) (**D**: # of embeddings per word)
- **Convolution, ReLU** Pass the above input through a convolution layer and then through ReLU. Resulting dimension: (**B**, **K**, **Y**)(**K**: Depth of filter, i.e. # of filters of given height, **Y**: # of values sliding the filter over the input)
- **Pooling** Pass through mean pooling. Resulting dimension: (**B**, **K**)
- The resulting encoded embedding of dimension (**B**, **K**) for **B** words in a given batch.

4. DOMAIN ADAPTATION (DA)

We use adversarial training techniques [9] to adapt to a domain having no labeled data, having trained a model on a domain with labeled data. In this section and the next, we describe the dataset we worked on, the architecture for the adversarial model we use, and other models for DA we tried.

4.1 Dataset

We worked with two datasets, one as our source domain, wherein we had labeled data and another as our target domain, where we had no labeled data. The task on both these datasets was the same - to predict similarity between questions. The source domain in our experiments was the *AskUbuntu* dataset, described in the previous section. The target domain was the *Android Stack Exchange*, a question-answering site on Android-related issues. Table 2 describes the summary statistics of this dataset. For the target domain, we had XX sentences to test. Each query was expert-annotated with similar and dissimilar queries they corresponded to. A TF-IDF representation was used as a baseline for evaluating domain adaptation.

4.2 Model

We model this task of unsupervised domain adaptation using a neural adversarial approach [9]. The core idea is as follows -

- We define an encoder which takes in data from the source domain and learns a representation which predicts the similarity for tasks in its own domain, given that we have labels for this domain. This is similar to the in-domain learning described in the previous section. We refer to this network as the *encoder network* here on.
- While the primary goal of predicting $\mathcal{Y}_{\text{source}}$ remains, we define an adversarial objective to guide the direction of the embeddings encoded by the encoder network in a way to make them abstract enough to be able to discriminate $\mathcal{Y}_{\text{target}}$ as well.
- We do this by setting up another network to learn this adversarial objective. We refer to this network as *adversarial network* here on. We set the goal of the

adversarial network to predict whether a given datapoint input to it originates from either the source or target dataset. The idea behind setting up this objective is that if the origin of the input data is being discriminated well by the adversarial network, then the features are still specific enough to discriminate between the two datasets. What we want is to however have an abstract enough feature space which, in spite of predicting the labels of the source well, is unable to distinguish between the source and target dataset.

- The objective of the adversarial network is set up in a way to propogate a negative weight to the encoder whenever it successfully discriminates between source and target. This ensures the successful transfer of the encoded embeddings to the target domain. Figure 4.2 depicts this arrangement.

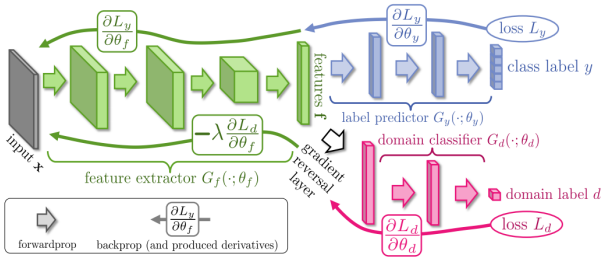

Figure 2: Adversarial network for unsupervised domain adaptation investigated in this work. Image reference: Ganin et al., 2015 [9]

5. EXPLORING DOMAIN ADAPTATION

We explored two other techniques to carry out unsupervised domain transfer - Doc2Vec [14] and Siamese recurrent networks [16]. We describe each briefly here:

- Doc2Vec Paragraph vectors, or doc2vec, were proposed by Le and Mikolov (2014) as a simple extension to word2vec to extend the learning of embeddings from words to word sequences. doc2vec is agnostic to the granularity of the word sequence it can equally be a word n-gram, sentence, paragraph or document. In this paper, we use the term "document embedding" to refer to the embedding of title and body.
- **Siamese networks** TODO

6. EXPERIMENTS AND RESULTS

We pose the following research questions, whose answers we investigate in this work

- **E1:** How well do deep neural encoders solve the task of in-domain question retrieval. How does it compare to baselines and the state of the art? What are the limitations.
- **E2:** How effective are deep adversarial neural methods in adapting from one domain to another to learn a common task. How does it compare to the baselines and state of the art? What are the limitations?
- **E3:** How effective is doc2vec in adapting knowledge from one domain to another.
- **E4:** How effective are Siamese networks in adapting knowledge from one domain to another.

6.1 Experiment 1

To answer **E1**, we measure the performance of our neural encoders against a baseline of BM25scores provided for each query in the test set of the corpus. These scores are precomputed for each query-positive and query-negative pair. The ranked list of these scores per query is used as a baseline performance on the four metrics we choose (see Section 3 for details on the metrics).

Implementation specifics: In our implementation, we ensure that stop words are removed from the corpus. This reduces the average sentence length by 30%. Each input in a batch that we pass to our network consists of words in a query, one positive question corresponding to it and all its corresponding negative questions. We pad unseen words in the pre-trained embeddings with 0s. We also equate sentence length in a batch by padding with 0s. The 0-padding is taken into account while mean pooling the intermediate results in the network. We do not train the pre-trained embedding layer. In this work, to limit computational requirements, we consider only the first 10 words in the title and the first 100 words in the body of each sentence. Further, we score only 40 random negative questions per query instead of the full 100 provided in the dataset. All hyperparameters are tuned on the dev set. The number of trainable network parameters are constrained to a budget of 450K parameters so that the expressability of the models are comparable. Table XX describes the various hyperparameters of the encoder and adversarial network. The model is trained on XX samples and tested on YY samples. All experiments were carried out on a XXX machine. Each model was run for XX epochs to train. Each epoch took XX minutes, and we were able to train the network on the full training-set in under 180 minutes.

Results: Table 5 tabulates the performance of neural encoders on the task of question retrieval and similiarity. CNN and LSTM denote the performance of our two neural encoders while BM25 denotes the baseline scores gathered from Apache Lucene. Against each of our models, we also record the number of trainable parameters used. The performance is on the scale 0-1, where higher the number, better the retrieval. We analyze and discuss these results in Section 7.

CNN		LSTM	
Parameter	**Value**	**Parameter**	**Value**
output channels	250	hidden size	128 × 2
loss margin	0.4	loss margin	0.4
kernel size	[1-5] × 300	bidirectional	true
pooling	average	pooling	average

6.2 Experiment 2

To answer **E2**, we measure the performance of domain adaptation on the *Android Stack Exchange* dataset, having learnt a model on the *Ask Ubuntu* dataset.

Evaluation: The evaluation metric for this task is changed to instead evaluate a binary classification task - is a query similar to a given candidate question present in the corpus or not. As a consequence, the robustness of the system is measured using a $AUC(0.05)$ score. This designers of this

problem statement chose this different metric compared to those used in **E1**. This makes comparing results between the two experiments hard.

Baselines: We establish multiple baselines - some stronger than the other. To begin with, we train a TF-IDF model on the source domain and evaluate on the target domain. As a slightly stronger baseline, we directly apply the model learnt in **E1** onto the target domain.

Implementation specifics: We train the encoder network from scratch to train the two objectives detailed in Section 4. We maintain the constraint of evaluating on 40 negative samples for the source label classifier but do not vary the validation/test set for the target domain. We maintain the same constraints on the length of the title and body as in **E1**. Stop-words and word-casing are handled for the target domain. We evaluate the system on both, pre-trained embeddings used in **E1** and off-the-shelf GloVe embeddings [19]. All hyperparameters are tuned on the dev set. The model is trained on XX samples and tested on YY samples. All experiments were carried out on a XXX machine. Each model was run for XX epochs to train. Each epoch took XX minutes, and we were able to train the network on the full training-set in under 180 minutes.

Results: Table 5 tabulates the performance of neural encoders on the task of domain adaptation for question retrieval. Models with the suffix $xx - \mathsf{direct}$ in column 1, Table 5 correspond to the direct-transfer baseline models. The adversarial models are described with the suffix $xx - \mathsf{adv}$. The evaluation metric is AUC(0.05), which is reported on a scale of 0-1. An AUC(0.05) score of 1 implies that the model has a false positive rate of $<0.05\%$. Results are reported for both, the dev and test sets. We analyze and discuss these results in Section 7.

CNN		LSTM	
Parameter	**Value**	**Parameter**	**Value**
# of sentences	40	# of sentences	50
# of stop words	50	# of sentences	90
mean word length - title	299	# of sentences	300
mean word length - body	299	# of sentences	300
Batch size	299	# of sentences	300
Number of epochs	299	# of sentences	300
mean word length - body	299	# of sentences	300

Table 3: Parameters of the encoder and adversarial networks for domain adaptation

6.3 Experiment 3

TODO To answer **E1**, we measure the performance of our neural encoders against a baseline of BM25scores provided for each query in the test set of the corpus. These scores are pre-computed for each query-positive and query-negative pair. The ranked list of these scores per query is used as a baseline performance on the four metrics we choose (see Section 3 for details on the metrics).

Implementation specifics: In our implementation, we ensure that stop words are removed from the corpus. This reduces the average sentence length by 30%. Each input in a batch that we pass to our network consists of words in a query, one positive question corresponding to it and all its corresponding negative questions. We pad unseen words in the pre-trained embeddings with 0s. We also equate sentence length in a batch by padding with 0s. The 0-padding is taken into account while mean pooling the intermediate results in the network. We do not train the pre-trained embedding layer. In this work, to limit computational requirements, we consider only the first 10 words in the title and the first 100 words in the body of each sentence. Further, we score only 40 random negative questions per query instead of the full 100 provided in the dataset. All hyperparameters are tuned on the dev set. The number of trainable network parameters are constrained to a budget of 450K parameters so that the expressability of the models are comparable. Table XX describes the various hyperparameters of the encoder and adversarial network. The model is trained on XX samples and tested on YY samples. All experiments were carried out on a XXX machine. Each model was run for XX epochs to train. Each epoch took XX minutes, and we were able to train the network on the full training-set in under 180 minutes.

Siamese		doc2vec	
Parameter	**Value**	**Parameter**	**Value**
# of sentences	40	# of sentences	50
# of stop words	50	# of sentences	90
mean word length - title	299	# of sentences	300
mean word length - body	299	# of sentences	300

Table 4: Parameters of doc2vec and the Siamese network architectures

6.4 Experiment 4

TODO

7. DISCUSSION

7.1 Experiment 1

TODO: add adversarial training objectives, discuss stability of changing lambda, discuss differences with siamese networks etc. We observe that both, the LSTM and the CNN networks, beat the BM25 baseline by XX points on an average on each of the metrics. We notice that this accuracy is achieved with a very simple model architecture, without any effort invested in engineering features. Such a performance by off-the-shelf models suggests that neural models are an apt choice for modeling such tasks. Figures **??** describe the loss function over different epochs for the CNN and LSTM respectively. We note here though that we found the results to be sensitive to different hyperparameter choices. With the right choice of hyperparameters, we could gain 10 points on an average on all the metrics. For instance, we found tanh to produce better results over using ReLU as an activation function for the CNNs. We found the learning rates of *Adam*, the optimizer, when outside the range of XX, to have an adverse effect on the results. We found that mean pooling and max pooling produced comparable results on both networks. We found the choice of the margin in the maximum-margin separator to specially have an effect on the performance. Moving it by 0.5 points above or below the margin set for the final models (see Table **??**) decreased the final accuracy by 0.10 units.

Model name	# of param in the n/w	Test				Dev			
		MRR	P@1	P@5	MAP	MRR	P@1	P@5	MAP
BM25 baseline	NA	0.68	0.54	0.42	0.56	0.66	0.52	0.42	0.52
CNN	337290	0.63	0.48	0.40	0.43	0.63	0.49	0.41	0.51
LSTM	337290	0.69	0.54	0.42	0.56	0.65	0.48	0.44	0.54

Table 5: Results for Experiment 1 - In-domain learning using neural models

Model name	Description	AUC(0.05) - Test	AUC(0.05) - Dev
TF-IDF	Baseline	0.58	0.58
CNN-direct	Direct transfer	0.62	0.62
LSTM-direct		0.62	0.63
CNN-adv	Adversarial domain transfer	0.70	0.70
LSTM-adv		0.70	0.70
Siamese networks	Other domain transfer methods	0.70	0.70
Doc2Vec		0.70	0.70

Table 6: Results from Experiment 2 - Domain adaptation

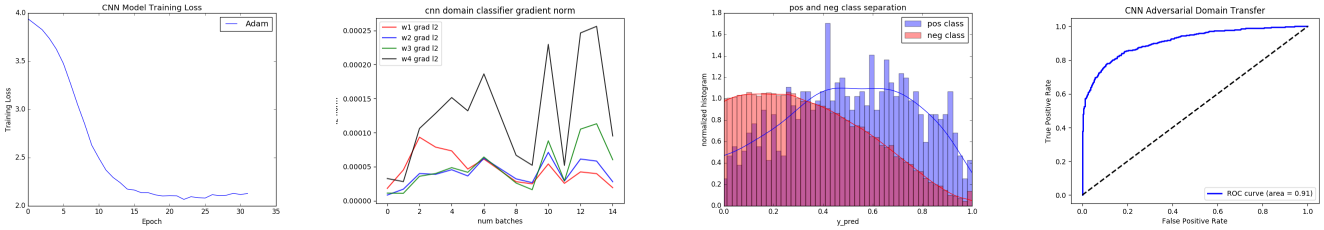

Figure 3: Results from Experiment 1

7.2 Experiment 2

The baseline is measured against an unsupervised (TF-IDF) model being learnt on the source domain. In comparison, we see the direct transfer model to perform slightly better by both, the CNN and LSTM. This is expected since these models learn , but relatively the

8. ACKNOWLEDGEMENTS

The authors thank the course staff of 6.864, Fall 2017, MIT. This was a fun project!

9. REFERENCES

[1] R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853, 2005.

[2] M. Baktashmotlagh, M. T. Harandi, B. C. Lovell, and M. Salzmann. Unsupervised domain adaptation by domain invariant projection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 769–776, 2013.

[3] K. M. Borgwardt, A. Gretton, M. J. Rasch, H.-P. Kriegel, B. Schölkopf, and A. J. Smola. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):e49–e57, 2006.

[4] B. Chen, W. Lam, I. Tsang, and T.-L. Wong. Extracting discriminative concepts for domain adaptation in text mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 179–188. ACM, 2009.

[5] H. Daumé III. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*, 2009.

[6] C. Dos Santos, L. Barbosa, D. Bogdanova, and B. Zadrozny. 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)*, volume 2, pages 694–699, 2015.

[7] M. Feng, B. Xiang, M. R. Glass, L. Wang, and B. Zhou. Applying deep learning to answer selection: A study and an open task. In *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*, pages 813–820. IEEE, 2015.

[8] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars. Unsupervised visual domain adaptation using

subspace alignment. In *Proceedings of the IEEE international conference on computer vision*, pages 2960–2967, 2013.

[9] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189, 2015.

[10] B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2066–2073. IEEE, 2012.

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

[12] J. Huang, A. Gretton, K. M. Borgwardt, B. Schölkopf, and A. J. Smola. Correcting sample selection bias by unlabeled data. In *Advances in neural information processing systems*, pages 601–608, 2007.

[13] Y.-B. Kim, K. Stratos, R. Sarikaya, and M. Jeong. New transfer learning techniques for disparate label sets. In *ACL (1)*, pages 473–482, 2015.

[14] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196, 2014.

[15] 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.

[16] P. Neculoiu, M. Versteegh, M. Rotaru, and T. B. Amsterdam. Learning text similarity with siamese recurrent networks. *ACL 2016*, page 148, 2016.

[17] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2011.

[18] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[19] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.