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 available wide variety of content 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 such 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. Given a question and answer pair present on an online discussion forum, such a retrieval task finds 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 *Ask Ubuntu*. 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 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; 17] as well as the performance on different architectures like Doc2Vec [16] and Siamese neural networks [19].

This study demonstrates the relevance of neural techniques in a critical NLP task like question-answer retrieval. We

*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.[21]

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. [17] 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 [17]. 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 [21]. Our work explores techniques in unsupervised domain adaptation, where the source and target domains ($\mathcal{D}$) are different, but share a common task ($\mathcal{T}$) 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 [20; 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 distribution means in the kernel-reproducing Hilbert space [3; 12], whereas [10; 8] map the principal axes associated

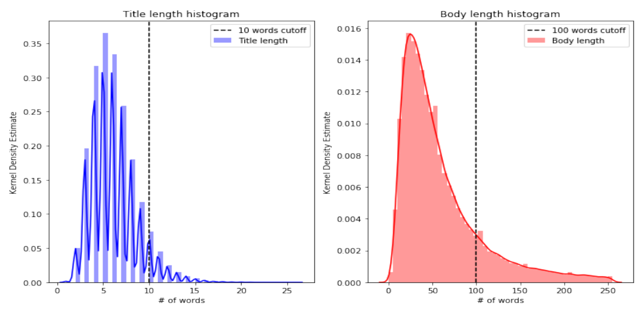


Figure 1: *Ask Ubuntu* dataset description

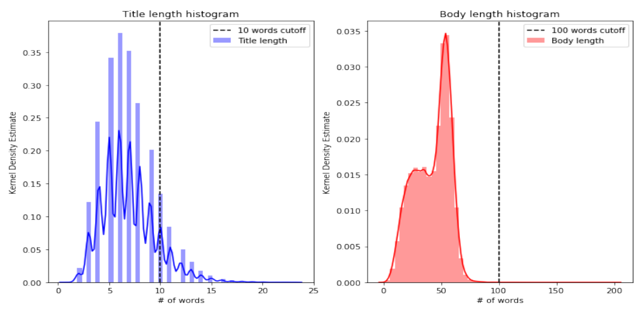


Figure 2: *Android* dataset description

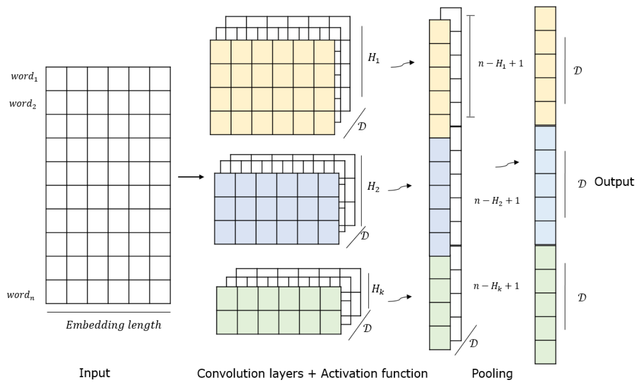


Figure 3: CNN architecture

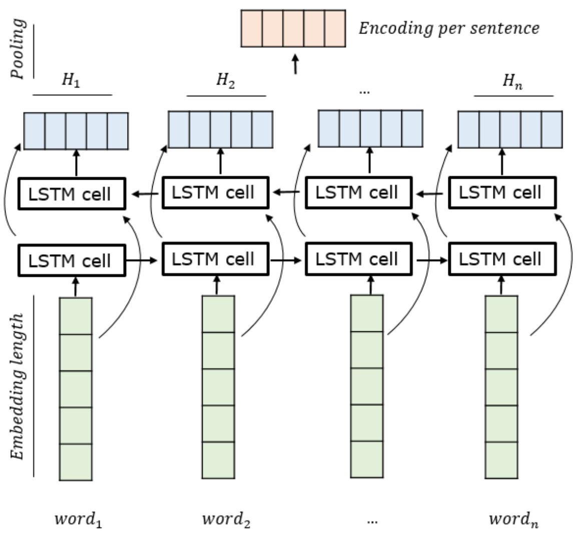


Figure 4: LSTM architecture

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 [17].

	source($\mathcal{D}_\mathcal{S}$)	target($\mathcal{D}_\mathcal{T}$)
Description	AskUbuntu	Android SE
Total # of questions	167763	42970
Avg words/title (std dev)	6 (2.5)	9 (3.9)
Avg words/body (std dev)	92 (210.6)	69 (24.1)
# unique queries in train-set	12724	NA
# unique queries in test-set	200	1148

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 6 words and the average body length is 92 words long after removing stop-words. Figure 1 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 a total of 167K questions. Of these, 12724 were used as queries to train on.
- Each query-positive and query-negative 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 in the following manner

- 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

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

  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 [17].
- 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

Figure 3 illustrates the CNN architecture implemented in this work. We detail the inputs to the different layers of the network here. 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** Looks up word embeddings of indices in **X**. Resulting dimension: (**B**, **W**, **D**) (**D**: # of embeddings per word)
- **Convolution, tanh** A convolution layer and then through tanh is applied to **X**. 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** Mean pooling of convolution layers. Resulting dimension: (**B**, **K**)
- The resulting encoded embedding of dimension (**B**, **K**) for **B** words in a given batch.

3.4 LSTM

Figure 4 illustrates the LSTM architecture implemented in this work. We detail the inputs to the different layers of the network here.

- **Input vector X** of dimensions: [**B**, **W**, **D**], where **B** is the batch size, **W** is max number of words, and **D** is the embedding dimension
- **Embedding layer** Looks up word embeddings of indices in **X**.
- **LSTM layer** returns the hidden state and LSTM cell state.
- **Pooling** Average pooling is used to average the outputs of the bi-directional LSTM producing a summary vector of dimension [**B**, $2 \times$ **H**], where **H** is the hidden size.

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.

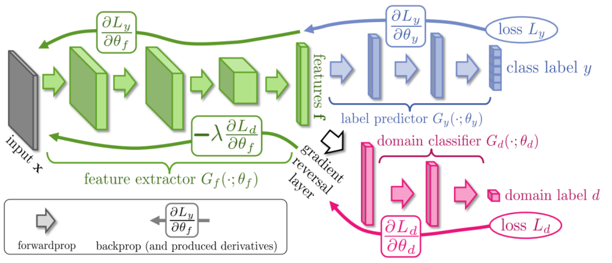
4.1 Dataset

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. No training set was available for this dataset - all questions were divided into validation and test sets. Each query was again associated with a few (< 10) positive and 100 negative questions. These were expert-annotated. The final model was tested on 1148 unique queries.

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



$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
loss L_y
class label y
features f
label predictor $G_y(\cdot;\theta_y)$
$-\lambda \frac{\partial L_d}{\partial \theta_f}$
input x
feature extractor $G_f(\cdot;\theta_f)$
gradient reversal layer
domain classifier $G_d(\cdot;\theta_d)$
domain label d
$\frac{\partial L_d}{\partial \theta_d}$
loss L_d
forwardprop
backprop (and produced derivatives)

Figure 5: Adversarial network for unsupervised domain adaptation investigated in this work. Image source: Ganin et al., 2015 [9]

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 propagate 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 5 depicts this arrangement.
- Specifically, let θ_f be the parameters of the question encoder (generator) G_f and θ_d be the parameters of the domain classifier (discriminator). The loss objective then is defined as:

$$\min E(\theta_f, \theta_d) = \sum_{i=1}^{n} L_y^i(\theta_f) - \lambda^t \sum_{i=1}^{n} L_d^i(\theta_f, \theta_d) \quad (2)$$

$$= \sum_{i=1}^{n} L_y(G_f(x_i;\theta_f), y_i) - \lambda^t \sum_{i=1}^{n} L_d(G_d(G_f(x_i;\theta_f), \theta_d), y_i) \quad (3)$$

where $L_y(\theta_f)$ is the multi-margin loss and $L_d(\theta_f, \theta_d)$ is the binary cross-entropy loss. During adversarial training, we jointly minimize the multi-margin loss and maximize cross-entropy loss. We accomplish this with a single back-propagation step by minimizing the total loss $E(\theta_f, \theta_d)$ and reversing the gradient of the domain classifier during the weight update, e.g. by setting a negative learning rate.

5. EXPLORING DOMAIN ADAPTATION

We explored two other techniques to carry out unsupervised domain transfer - doc2vec [16] and Siamese recurrent networks [19]. We motivate why we chose these techniques and thereafter briefly describe each of the techniques.

5.1 Motivation

The larger theme behind our exploration is finding simple structures which could achieve domain adaptation. We wanted to understand what basic blocks one could change in a neural architecture which would aid in better domain adaptation. In this spirit, we explored two components - dropouts and embeddings. We explored the effect of dropout on Siamese RNNs and explored using an expressive embedding resulting from a doc2vec model.

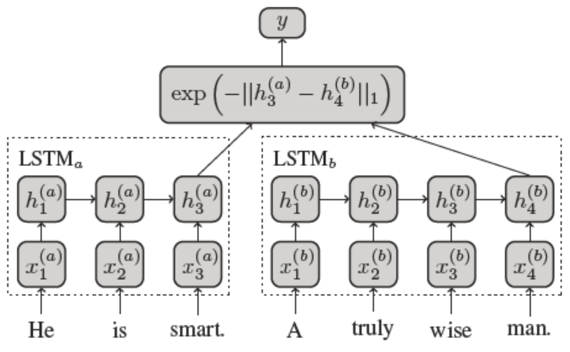


Figure 6: Siamese RNN. Image source: Neculoiu et al. [19]

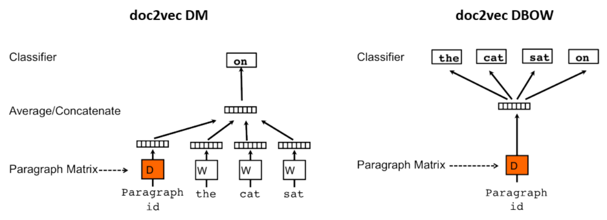


Figure 7: Doc2Vec. Image source: Le et al. [16]

Siamese RNN for domain adaptation Siamese RNNs in recent literature have shown to outperform the state of the art techniques in sentence similarity tasks. We wanted to explore if embeddings generated by such an architecture would be better suited for our task. Further, very recent literature[18] has also explored using such an architecture to achieve domain adaptation. In addition to what we learned from [18], we wanted to understand the simple effect of adding dropout layers to it. Recent work has shown that dropout can effectively achieve domain adaptation [24]. We wanted to investigate if adding dropouts to Siamese networks could achieve domain adaptation as well. We did not expect it to produce models as expressive as the neural adversarial techniques explored in this work, but we expected it to at least provide a competitive baseline for such similarity tasks. We also chose this technique to learn and implement a contrastive loss function, and in doing so, understand its effect on such a task.

Doc2Vec for domain adaptation We wanted to understand the role of expressive embeddings in a domain adaptation task. First, we wanted to understand how they would perform in a direct transfer setting and further, whether replacing them with the encoders in the current advarsarial setting had an effect. To study this effect, we investigate Doc2Vec - a recent, specialized document-level embedding.

5.2 Techniques

- **Siamese RNN** Siamese neural networks [19] are characterized by shared weights and are trained with a *contrastive loss function* that differentiates the inputs.

They are thus well-suited for computing similarity between a query and a candidate question. We use a shared bi-directional LSTM as an input encoder and add several fully connected layers for domain adaptation. The architecture is shown in Figure 6. The contrastive loss function $L(x_1, x_2, y)$ is defined as

$$L = \frac{1}{2}(1-y)D_w^2 + \frac{1}{2}y\max\{0, -D_w + \Delta\}^2 \quad (4)$$

$$D_w = \sqrt{[RNN(x_1) - RNN(x_2)]^2} \quad (5)$$

where y is a binary label equal to 0 for similar inputs (in which case we want to minimize the distance D_w) and equal to 1 for dissimilar inputs (in which case we want to maintain a margin Δ). The distance D_w is a pairwise Euclidean distance between RNN encodings of the inputs x_1 and x_2. In order for the Siamese RNN to generalize to domain adaptation, we include dropout[23] in between the dense layers with keep probability equal to 0.5. Alternatively, *variational dropout*[14] can be used to tune the keep probability to improve generalization performance.

- **doc2vec** The doc2vec model was proposed by Le and Mikolov [16] as an extension of word2vec to learn fixed-length feature representations from variable-length pieces of texts, such as sentences, paragraphs, and documents. Thus, each document is represented by a dense vector which is trained to predict words in the document. While the word vectors represent the concept of a word, the document vector intends to represent the concept of a document. There are two doc2vec architectures: Distributed Memory (DM) and Continuous Bag of Words (CBOW) as shown in Figure 7. The DM model acts as memory that remembers what is missing from the current context. The CBOW model samples a text window, then samples a random word from the text window and forms a classification task of predicting the random word given the text window. Such a document vector representation can likely model both domains in a domain-adaptation task.

6. EXPERIMENTS AND RESULTS

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

- **E1:** How well do deep neural encoders such as LSTM and CNN solve the task of in-domain question retrieval. How do they 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 do they compare to baselines and the state of the art? What are the limitations?
- **E3:** How well can a Siamese neural network architecture and doc2vec model question-similarity tasks and whether they aid in domain adaptation.

CNN direct		**LSTM direct**	
Parameter	**Value**	**Parameter**	**Value**
Kernel size	[1-5] × 200	Hidden size	128 × 2
Output channels	5×200	Loss margin	0.4
Weight decay	1e-5	Weight decay	1e-5
Learning rate	1e-3	Learning rate	1e-3
Loss margin	0.4	Bidirectional	True
Pooling	average	Pooling	average
Batch size	32	Batch size	32

Table 3: Optimal hyperparameters of the direct transfer CNN and LSTM networks - Experiment 1

CNN adversarial		**LSTM adversarial**	
Parameter	**Value**	**Parameter**	**Value**
λ	1e-6	λ	1e-7
Learning rate	1e-3	Learning rate	1e-3
Discriminator	MLP	Discriminator	MLP
Output channels	1250	Hidden size	128
Kernel size	[2-6] x 250	Bidirectional	False
Embed dim	300	Embed dim	300
Pooling	average	Pooling	average
Weight decay	1e-5	Weight decay	1e-5
Batch size	32	Batch size	32

Table 4: Optimal hyperparameters of the adversarial CNN and LSTM networks - Experiment 2.

Siamese RNN		**doc2vec DM/DBOW**	
Parameter	**Value**	**Parameter**	**Value**
Hidden size	128	doc2vec dim	256
Bi-directional	True	Context window	10
Dropout prob	0.5	Min count	2
Loss margin	0.5	Learning rate	0.025

Table 5: Optimal hyperparameters of Siamese RNN and doc2vec models - Experiment 3

6.1 Experiment 1

To answer **E1**, we measure the performance of our neural encoders against a baseline of BM25 scores 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 represent unseen words in the pre-trained embeddings with an all-zero vector. We equate sentence length in a batch by padding with 0s till maximum length of the batch. 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 3 lists the various hyperparameters of the encoder network. The model is trained on 12K queries and tested on 2K queries.

Model	# params	Test				Dev			
		MRR	P@1	P@5	MAP	MRR	P@1	P@5	MAP
BM25 baseline	N/A	0.68	0.54	0.42	0.56	0.66	0.52	0.42	0.52
CNN	420,600	0.70	0.56	0.42	0.57	0.70	**0.53**	0.42	0.53
LSTM	337,920	**0.71**	**0.57**	**0.43**	**0.57**	**0.72**	0.49	**0.44**	**0.54**

Table 6: Results for Experiment 1: in-domain learning using neural models

Model	AUC - Test	AUC(0.05) - Test
TF-IDF	0.94	0.62
CNN-direct	0.88	0.53
LSTM-direct	0.90	0.59
CNN-adversarial	**0.96**	**0.76**
LSTM-adversarial	0.96	0.74
Siamese	0.74	0.20
doc2vec-DM	0.83	0.36
doc2vec-DBOW	0.82	0.29

Table 7: Results for Experiments 2 and 3: Domain Adaptation

Words not seen during training are treated as 0-vectors. All experiments were carried out on a CUDA-enabled GTX Titan GPU. Each model was run for 16 training epochs, averaging to 15 min/epoch for LSTM and 10 min/epoch for CNN.
Results: Table 6 tabulates the performance of neural encoders on the task of question retrieval and similiarity. We compare ranking performance of CNN and LSTM models against BM25 based ranking by Apache Lucene. Higher number indicates higher performance. We also record the number of trainable parameters used. We analyze and discuss these results in Section 7.

6.2 Experiment 2

To answer **E2**, we measure the performance of domain adaptation on the *Android Stack Exchange* dataset, having learned a model on the *AskUbuntu* dataset.
Evaluation: The evaluation metric for this task is Area Under ROC Curve (AUC), since this is modeled as a binary classification task of whether a query is similar or not to a candidate question. As a consequence, the robustness of the system is measured using a AUC(0.05) score, which corresponds to AUC with FPR ≤ 0.05.
Baselines: We compare our techniques against two baselines: the count based TF-IDF model trained on source domain and evaluated on the target domain serves as one baseline. Additionally, models learned in **E1** and directly applied onto the target domain serve as another.
Implementation specifics: We train the adversarial LSTM and CNN models from scratch as detailed in Section 4. We maintain the constraint of 40 negative samples for the source domain but do not vary the dev/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 6B tokens, 300-dimensional GloVe embeddings [22]. All hyperparameters are tuned on the dev set. Optimal parameters for the final model are listed in Table 4. The model is trained on the dataset from **E1** and tested on 1.1K unique queries belonging to the target domain.
Results: Table 7 tabulates the performance of LSTM and CNN neural encoders on the task of domain adaptation for question retrieval. Higher number indicates higher performance. AUC(0.05) refers to ROC AUC for which FPR $\leq$ 0.05. We analyze and discuss these results in Section 7.

6.3 Experiment 3

To answer **E3**, we explore a new architecture based on Siamese RNN and a new representation of the question title and body based on doc2vec as shown in Figure 7
Implementation specifics: We use the same set-up as the direct transfer experiments, in which we train on source and evaluate on target. In the case of Siamese RNNs, we feed a batch of (query, candidate) question pairs and a label 0 for a pair of similar questions 1 for a pair of dissimilar questions. We use 6B tokens, 300-dimensional GloVe embeddings to represent the word vectors. In addition to sister Siamese networks with shared weights, we add several dense layers to increase the degrees of freedom required for domain transfer. To aid generalization, a dropout layer is used between the dense layers with keep probability of 0.5.
In the case of doc2vec, we consider two variations: the distributed memory (DM) and the distributed bag of words (DBOW). In both cases, the title and the body are concatenated into one document for every query. The doc2vec models are then trained on source dataset for 64 epochs with a linear learning rate decay schedule starting at 0.025. The word embeddings and the document vectors are learned during this training process.
Results: Table 7 tabulates the performance of Siamese RNN and doc2vec models on the task of domain adaptation for question retrieval. Higher numbers indicate better performance. AUC(0.05) refers to ROC AUC for which FPR $\leq$ 0.05. We discuss these results in Section 7.

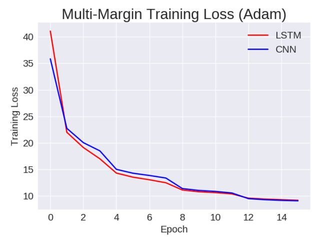


Figure 8: In-domain LSTM and CNN training loss (Experiment 1)

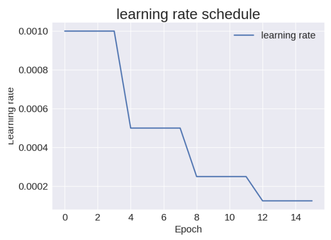


Figure 9: Learning rate schedule (All experiments)

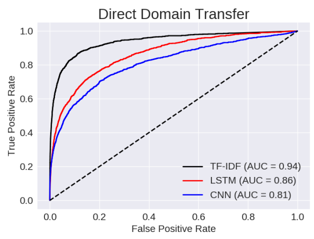


Figure 10: ROC direct transfer (Experiment 2)

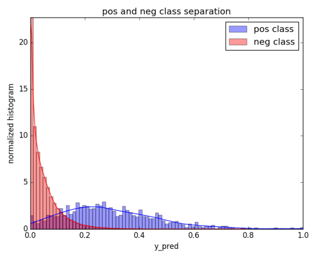


Figure 11: TF-IDF class separation (Experiment 2)

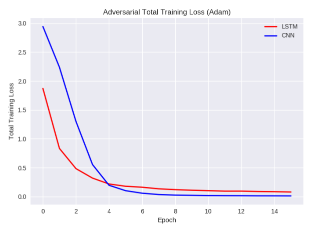


Figure 12: LSTM and CNN total adversarial training loss (Experiment 2)

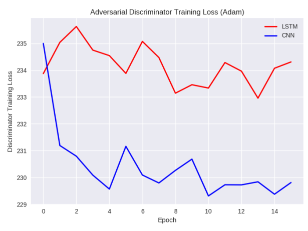


Figure 13: Discriminator training loss (Experiment 2)

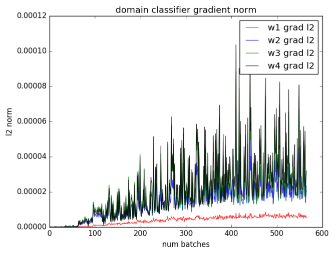


Figure 14: Gradient norms - Domain classifier (Experiment 2)

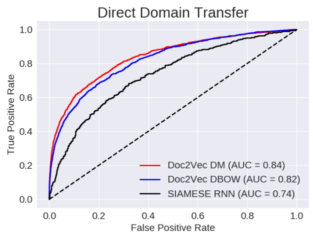


Figure 15: ROC direct transfer (Experiment 3)

7. DISCUSSION

7.1 Experiment 1

Table 6 summarizes the performance of our models on four IR-metrics. We see that both the neural models clearly beat the baseline BM25 information retrieval approach on all metrics on the test-set by an average of 0.03 points. We find that LSTM achieves a slightly higher performance than CNN. However, due to the recurrent nature of LSTM, it takes longer to train. In the process of hyper-parameter optimization, we found average pooling to perform better. Bi-directional LSTM performed better than forward LSTM. For the CNN, we found that increasing the number of output channels and using *tanh* activation performed better. Figure 8 shows the loss for both the networks when training with a learning rate decay schedule shown in Figure 9. The initial learning rate was set to 1e-3 and was halved every 4 epochs. We found the performance to be sensitive to the choice of the loss margin. We achieved best loss reduction over iterations at margin 0.4. These results suggest that LSTMs and CNNs are useful encoding techniques for representing a query. Additional improvement can be attained using a gated CNN architecture as described in [17].

7.2 Experiment 2

Table 6 summarizes the performance of our models on the AUC(0.05) metric. The TF-IDF baseline transfer method has an AUC(0.05) score of 0.62. We see that the Direct-transfer CNN and LSTM baseline models are comparable to the TF-IDF baselines, but do not perform as well. The neural models for domain adaptation clearly beat these two baselines by 0.13 points on an average. These are encouraging results and demonstrate the robustness of the approach to domain adaptation. Figure 10 shows direct domain transfer ROC AUC performance for the three baseline models - TF-IDF, LSTM and CNN. TF-IDF achieves highest direct transfer AUC. Figure 11 shows the positive and negative class separation attained by the TF-IDF model. This histogram is normalized since the number of negative examples in the dataset far exceeds the number of positive (similar) question pairs. By varying the threshold on the predicted label, we attain the ROC curve shown on the left. Notice that the two classes are well-separated by the TF-IDF model leading to a high AUC score. Figure 12 shows the total adversarial training loss for the LSTM and CNN models. We see the difference in initialization of the two models by comparing the loss at epoch zero. The losses plateau after 8 epochs resulting in early stopping. Figure 13 shows the stochastic nature of the domain classifier loss. Its magnitude is much larger in comparison to the question encoder loss and therefore a small value of λ ($1e$-6) is used to attenuate it. We found that AUC performance was very sensitive to the choice of λ. We selected the best value of lambda by sweeping between $1e$-3 to $1e$-7. We also examined the gradient norms of the domain classifier as shown in Figure 14. We found the gradient of the last among the 4 dense layers of the MLP to be the highest and most stochastic.

7.3 Experiment 3

Figure 15 shows the ROC AUC performance of the Siamese RNN and doc2vec models. From Table 7, we see that these two techniques perform less favorably in comparison to adversarial domain transfer. We tuned the hyperparameters on the validation set, however, we were unable to match performance of CNN and LSTM direct transfer models. We hypothesize that a deeper Siamese RNN architecture will be more effective in domain adaptation. However, in order to confirm our hypothesis, we would have to explore this further. The doc2vec models perform slightly more favorably as compared to the Siamese network. However, the doc2vec models were unable to match the performance of direct transfer LSTM and CNN models. The community has seen mixed results in using doc2vec. This very recent work explores empirically its utility [15] and recommends training doc2vec models on large, external corpora, preferably using pre-trained embeddings, for best results. We did not deal with the datasets in the order of magnitude mentioned in [15]. We do wish to continue exploring its utility. Among the two doc2vec models, the DM architecture performs slightly better than DBOW. This is expected and is also reported in the original paper [16].

8. CONCLUSION

We explore the task of question retrieval in online discussion forums. Given a question-answer pair, we employ neural techniques to detect similar question pairs on Ubuntu and Android Stack Exchange forums. We find that neural techniques model this task well and outperform traditional IR-based approaches in finding similar content. Another area explored in this work is that of unsupervised domain transfer. Given a labeled dataset of question-answer pairs from a discussion forum on one topic, we evaluate how well neural approaches are able to transfer this knowledge of similarity detection onto question-answer pairs on another topic. No labeled information is available for the pairs on the new topic. We find that neural adversarial approaches to domain transfer perform better than baseline domain transfer approaches. We also explore other architectures for domain adaptation. We find that Siamese RNN, although suited for domain-adaptation-like tasks, does not perform as well as the adversarial setup. Further, doc2vec is unable to capture the feature-space which the adversarial network trains, rendering them unsuitable for this domain adaptation task. The simplicity in the architecture, the lack of any feature engineering required, and relatively low training time make adversarial neural models worthy of further study and possible adoption in industrial systems for such tasks.

9. ACKNOWLEDGEMENTS

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

10. 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] D. P. Kingma, T. Salimans, and M. Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583, 2015.

[15] J. H. Lau and T. Baldwin. An empirical evaluation of doc2vec with practical insights into document embedding generation. *arXiv preprint arXiv:1607.05368*, 2016.

[16] 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.

[17] 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.

[18] S. Motiian, M. Piccirilli, D. A. Adjeroh, and G. Doretto. Unified deep supervised domain adaptation and generalization. In *The IEEE International Conference on Computer Vision (ICCV)*, volume 2, 2017.

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

[20] 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.

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

[22] 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.

[23] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

[24] Y. Yang and J. Eisenstein. Fast easy unsupervised domain adaptation with marginalized structured dropout. In *ACL (2)*, pages 538–544, 2014.