Neural Methods for Answer Passage Retrieval over Sparse Collections

Daniel Cohen
University of Massachusetts Amherst

NEURAL METHODS FOR ANSWER PASSAGE RETRIEVAL OVER SPARSE COLLECTIONS

A Dissertation Presented

by

DANIEL COHEN

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 2021

College of Information and Computer Sciences

NEURAL METHODS FOR ANSWER PASSAGE RETRIEVAL OVER SPARSE COLLECTIONS

A Dissertation Presented

by

DANIEL COHEN

ACKNOWLEDGMENTS

I would like to begin by thanking my advisor, Bruce Croft. Having entered with some basic research experience in GPU optimization, Bruce consistently pushed me to become a better researcher while never overwhelming me. After my first semester, he casually asked what I was planning on submitting to SIGIR, which was a shock as I had always viewed the process as a significant task and achievement. It was this early exposure and normalization to hurdles in my career that has allowed me to excel.

Next, I would like to thank my thesis committee: James Allan, who provided feedback which made me reconsider a long held belief in the definition of *ad-hoc* retrieval. Brendan O'Connor, who was also on my portfolio, and always available for discussion. Patrick Flaherty, who helped provide a grounded direction towards overcoming a shortcoming in my work.

I also owe my thanks to those kind enough to mentor me outside of the UMass community, as each one has had a profound impact on the way I view research. Katja Hofmann, who's guidance resulted in me retooling my research process after returning from MSR. My other mentor at the time, Bhaskar Mitra, introduced me to the benefit of impromptu research conversations even when the hypothesis had just been formed. I had such a positive experience at MSR that I came back another time under Fernando Diaz, whose conversations again left a lasting impact for my future career path. Lastly, I would like to thank my first mentor, Mohamed Zahran, who saw my interest in research before I knew I wanted to pursue a Ph.D. and gently introduced me to the process.

I would also like to thank those who have collaborated with me. Particularly Scott Jordan, who spent too much time teaching me about RL for someone not in his RL course. Without his input and discussion, I would not have been nearly as productive. Next, my peers at CICS who were a vital part of my Ph.D.: Qingyao Ai, Keping Bi, Hamed Bonab, John Foley, Helia Hashemi, Myung-ha Jang, Jiepu Jiang, Katie Keith, Youngwoo Kim, Yen-Chieh Lien, Blossom Metevier, Ali Montazeralghaem, Shahrzad Naseri, Chen Qu, Sheikh Muhammad Sarwar, Emma Tosch, Lakshmi Vikraman, Liu Yang, and Hamed Zamani.

Outside of academia, I would like to thank my parents Gloria and David Cohen for fostering my intellectual curiosity above all else, and fully supporting any endeavor I pursued as long as it was deliberate.

For everything not in these pages but who is responsible for their existence, I owe a tremendous debt of gratitude to my partner, Samantha Hill. Thank you.

This work was supported in part by the Center for Intelligent Information Retrieval and in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA) via AFRL contact #FA8650-17-C-9116 under subcontract #94671240 from the University of Southern California. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

ABSTRACT

NEURAL METHODS FOR ANSWER PASSAGE RETRIEVAL OVER SPARSE COLLECTIONS

FEBRUARY 2021

DANIEL COHEN

B.A., NEW YORK UNIVERSITY

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor W. Bruce Croft

Recent advances in machine learning have allowed information retrieval (IR) techniques to advance beyond the stage of handcrafting domain specific features. Specifically, deep neural models incorporate varying levels of features to learn whether a document answers the information need of a query. However, these neural models rely on a large number of parameters to successfully learn a relation between a query and a relevant document.

This reliance on a large number of parameters, combined with the current methods of optimization relying on small updates necessitates numerous samples to allow the neural model to converge on an effective relevance function. This presents a significant obstacle in the realm of IR as relevance judgements are often sparse or noisy and combined with a large class imbalance. This is especially true for short text retrieval where there is often only one relevant passage.

This problem is exacerbated when training these artificial neural networks, as excessive negative sampling can result in poor performance. Thus, we propose approaching this task through multiple avenues and examining their effectiveness on a non-factoid question answering (QA) task.

We first propose learning local embeddings specific to the relevance information of the collection to improve performance of an upstream neural model. In doing so, we find significantly improved results over standard pre-trained embeddings, despite only developing the embeddings on a small collection which would not be sufficient for a full language model. Leveraging this local representation, and inspired by recent work in machine translation, we introduce a hybrid embedding based model that incorporates both pre-trained embeddings while dynamically constructing local representations from character embeddings. The hybrid approach relies on pre-trained embeddings to achieve an effective retrieval model, and continually adjusts its character level abstraction to fit a local representation.

We next approach methods to adapt neural models to multiple IR collections, therefore reducing the collection specific training required and alleviating the need to retrain a neural model's parameters for a new subdomain of a collection. First, we propose an adversarial retrieval model which achieves state-of-the-art performance on out of subdomain queries while maintaining in-domain performance. Second, we establish an informed negative sampling approach using a reinforcement learning agent. The agent is trained to directly maximize the performance of a neural IR model using a predefined IR metric by choosing which ranking function from which to sample negative documents. This policy based sampling allows the neural model to be exposed to more of a collection and results in a more consistent neural retrieval model over multiple training instances.

Lastly, we move towards a universal retrieval function. We initially introduce a probe-based inspection of neural relevance models through the lens of standard natural language processing (NLP) tasks and establish that while seemingly similar QA collections require the same basic abstract information, the final layers that determine relevance differ significantly. We then introduce Universal Retrieval Functions, a method to incorporate new collections using a library of previously trained linear relevance models and a common neural representation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

The ability to infer whether a document is relevant such that it successfully answers a user's information need expressed through a query is a difficult task, which is exemplified by having entire career fields dedicated to retrieving desired information. While information retrieval models have been successful in this task, they often rely on having multiple relevant documents for a single training query or use carefully designed handcrafted text features to learn an effective model. Historically, certain query types, specifically ones where the information need is topical and commonly seen in general ad-hoc retrieval, perform well under these constraints. In such situations, standard features such as term statistics, page rank, or length provide a salient relevance signal while term order and context are of lesser importance [24, 20, 55].

However, as the information need of a query becomes increasingly specific, such as progressing from *penguins* to *how do penguins survive the winter without eating*, the majority of a document becomes non-relevant and only a segment of text, from a sentence to multiple paragraphs, remains useful to the user. This increase in desired specificity therefore produces a relevant subset from the original set of relevant documents from the general topical query [55]. While theoretically, the amount of relevant passages can be equal to the number of topically relevant passages or documents, in practice this subset is significantly smaller than the original topically relevant set. This reduction in candidate training data produces a sparsity in the collection where often there is only a single relevant passage within an entire collection to train on, motivating the need to develop alternative methods to sufficiently train a retrieval model to identify this relevant subset of text for these queries, referred to as answer passages.

Compounding this issue are the unique characteristics of these subsets of relevant passages, both in relation to the query as well as with respect to the original set of topically relevant documents. There exists a significant lexical gap between a query and a relevant "document" due to the reduced body of text. Traditional IR methods used for general topical ad-hoc retrieval which ignore word order or heavily rely on term frequency fail to effectively capture the relevance signal when semantics play an increasingly important role for answer passage retrieval [20]. Even in cases where there exists substantial text overlap, it is difficult to discern whether a passage belongs to the subset of relevant documents when only a topical retrieval model is used to determine relevance. While there exist handcrafted features that specifically improve short text retrieval [143], parallels in NLP, topical ad-hoc IR, and computer vision suggests that the features learned by neural networks can significantly improve performance.

With the advent of effective deep artificial neural networks [43], these models have significantly reduced the load on identifying salient features required to bridge the gap from ad-hoc retrieval to passage level retrieval models. However, these deep neural models require large amounts of training data and regularization to avoid overfitting [154]. For the task of answer passage retrieval, this issue is further exacerbated by the type of relevance signal contained in the training data. As mentioned, the increased specificity of the query results in only a small number of relevant training instances. While this paradigm is not unusual and is seen in document retrieval, the degree to which the number of relevant passages is reduced is unique. It is common for each query to only have a single relevant passage in an entire training set, which we refer to as a sparse collection. The reduced set of samples presents a significant challenge when trying to capture these non-linear relations as large amounts of parameters are required for the model that can quickly overfit on a collection if not enough samples are shown. Thus, we approach this problem via three core approaches: neural architecture selection, improved use of samples within a collection, and cross collection transfer learning. While deep neural models are universal function approximators [47], this flexibility relies on an unlimited amount of data to satisfy parity. We present methods to reduce this data need by leveraging the architectural bias of various neural structures, improve robustness to unseen passages, and propose the foundations of a novel method of implementing a set of small linear retrieval functions to construct a general neural retrieval function capable of quickly learning on new collections.

1.1 Outline and Contributions

In Chapter 2, we present a concise overview of information retrieval, passage retrieval, and their close relationship with sparse labels. A foundation of past work is used to contextualize the work proposed. In Chapter 3 we introduce neural methods for passage retrieval that specifically address the unique obstacles presented through multiple avenues and demonstrate the necessity for passage level approaches, the first of which leverages local relations between text at the passage level. In doing so, retrieval performance is significantly increased, demonstrating that there exists a significant shift in the mutual information between pairs of words comparing standard word embedding and collection specific representations. Next, expanding on the contribution of locally learned embeddings, we introduce a hybrid neural architecture that successfully transfers pretrained standard Wikipedia embeddings to the IR passage task, while dynamically constructing local embeddings for semantically different words as well as out of vocabulary text. This is done by using a character level convolutional sub-architecture. Concisely, the contributions for this part of the thesis are:

1 Contribution: *An end to end neural structure is developed where passage specific word embeddings are learned along with a network's parameters. This achieves a significant increase in performance correlated with the change in lexicon and definitions from the original word embedding corpus.*

2 Contribution: *A neural model is developed for noisy text, where a hybrid embedding is constructed from both character and word based embeddings. This further improves passage retrieval performance from the base end to end architecture.*

In Chapter 4, we incorporate additional information across passage collections by introducing an adversarial approach for subdomain transfer and informed negative sampling via reinforcement learning (RL) for neural models. Both these approaches focus on incorporating outside information, either from unseen subdomains or better covering non-relevant documents within a collection to achieve a more stable neural retrieval model. The contributions coming from this chapter are:

3 Contribution: *We introduce an adversarial framework to allow a single neural model to robustly handle distribution shifts between subsampling corpus and the full evaluation collection.*

4 Contribution: *We improve the sampling of an IR model during training through reinforcement learning, the first use of reinforcement learning to improve the training process of an IR model. In doing so, the retrieval model trained by this policy is more robust to the final re-ranking document distribution and random seeds.*

In Chapter 5, we show that neural retrieval models capture similar abstract features over different collections but attend to different information within their upper layers. We demonstrate this impact of neural representation for passage retrieval by expanding on the work of the previous chapter by viewing passage relevance through the lens of NLP tasks to understand what language structures correlate with passage level relevance via a probe based approach. Establishing common information across tasks, we propose a lifelong learning paradigm to facilitate a universal retrieval model.

By viewing retrieval from a reinforcement learning perspective, we adopt recent work in universal value functions to enable a neural retrieval model to effectively incorporate new collections or user specific relevance needs. The contributions from this chapter are:

5 Contribution: *A probe-based architecture is applied to question answer collections to demonstrate the information difference between collections with similar appearances via NLP auxiliary tasks.*

6 Contribution: *We introduce temporal difference updates for IR, showing that learning to rank is a specific case of an interpolated value function, to overcome sample inefficiency and high variance updates present in current RL for IR approaches.*

7 Contribution: *Collections are modeled as MDPs with over a common state representation. We introduce universal retrieval functions, where a conventional relevance model is linearly separated to create a shared neural representation component for multiple collections and individual linear relevance functions.*

In summary, these contributions result in the followings impacts to IR. First, we improve the effectiveness of neural models by understanding architectural biases and finding structures amenable to the characteristics of answer passage retrieval. Second, we improve sample efficiency by overcoming the severe class imbalance when training a neural model without access to industry data. Lastly, we provide the foundations to transfer relevance and representation functions to new collections to allow for additional training data. Through these key results, we introduce an alternative approach to BERT or other very large pre-trained language models [25, 90]: rather than relying on billions of parameters trained on massive data sets, our work proposes the idea of iteratively learning the necessary information. In this fashion, a model is able to effectively retrieve over multiple collections using a small number of parameters and efficient computation. This lightweight BERT surrogate is supported by Tang et al. [1], where a distilled single layer bidirectional LSTM neural model is able to achieve BERT level performance with a fraction of the parameters.

CHAPTER 2
RELATED WORK

2.1 Text Representation For Retrieval

The core of information retrieval involves successfully identifying the information need of the user expressed through a query. This relies on a function, referred to as a retrieval model, that scores the relevance of a document with respect to a query. A critical step in achieving an effective retrieval function is identifying what features should be used as input to the model, referred to as text representation. A common approach when creating these functions is to leverage descriptive statistics of a collection given a query such as term frequency and document length to rank relevant documents higher. Two classic methods that rely on this information are BM25 [52] and query likelihood (QL) [97]. BM25 relies on tunable parameters that weight the importance of document length and term frequency while QL models the probability of a document generating a query given the document's distribution of text. Incorporating additional term co-location information, *n-grams* leverage small windows of neighboring text to better capture the relation between a document and query. However, this results in a balancing act as too large of a window results in poor overlap between a query and document. A remedy to this approach is a backoff model [153], where if there are too few n sized grams, the model backs off and examines the overlap using $n-1$ grams [112]. Other approaches that rely on trained parameters such as linear models, support vector machines, or boosted trees[78, 18, 15], use additional features to represent the text of an input query-document pair such as various term frequency statistics, spam score, retrieval models like BM25, characteristics of body and title text, and other potentially salient information.

A common theme of these above representations is the use of handcrafted features to reduce the information that is passed into the retrieval model. While still a form of compression, this makes vocabulary based distributed representations such as word2vec, GloVE, and RANDWALK [80, 96, 7] a significant departure in text representations for retrieval. These word embedding approaches all represent a single term as a row in a factorized noisy pairwise mutual information matrix between all vocabulary terms present in a collection [7, 62], which allows a neural retrieval model generalize across samples [83]. Succinctly, they condense the co-occurrence information of terms into a much smaller vector such that a single term's vector representation now resembles other terms commonly found near it.

As Arora et al. [7] show, common embedding approaches that leverage co-occurrence information can be decomposed into a variation of RANDWALK. Levy and Goldberg [62] proceed to show that these embeddings can attain comparable performance on a variety of tasks under reasonable hyperparameter tuning. Thus, for the remainder of this work, we use a single embedding approach for each set of experiments given their equivalency.

2.2 Non Neural Retrieval Models

For general retrieval, BM25 and query likelihood (QL) have shown significant robustness across collections without any sensitivity to data sparsity [52, 97]. These models rely on a form of handcrafted features as mentioned above: term occurrence and document length which are agnostic to the number of training points within a collection. BM25 heavily relies on a combination of term frequency, inverse document frequency, and length to determine the importance of query terms within a document, and QL uses a probabilistic view of relevance modeled by the document distribution generating the query.

Other approaches fall into the realm of learning to rank approaches, and are of particular note as they lay the groundwork for the recent work in deep neural models. These models, such as support vector machines [18], boosted approaches [15], and simple linear models [110], all rely on explicit term statistics such as sparse encodings of the vocabulary space $\mathbb{R}^V$ or other handcrafted features requiring domain knowledge which include the ratio of stop words to non-stop words, inverse document term frequency, or PageRank score [100]. As IR relevance metrics rely on listwise evaluations which are discrete and non-convex, these models are often optimized to minimize the error of a convex surrogate loss like hinge loss in order to form an effective retrieval model. However, the efficacy of these approaches is bounded by the information captured by the features used to model the document [22]. While including additional handcrafted features is feasible, the kernel choice in support vector machines and limited representational power of other approaches prevents the use of unstructured data such as all one hot encodings of terms within a document. As we discuss shortly, these properties are found in deep neural networks, and allow for a substantial paradigm shift in retrieval models by enabling the loss function to select important features via the optimization process.

2.2.1 Passage Retrieval Approaches

The concept of identifying the salient portion of a document, referred to as a passage, has been studied for a number of years. Earlier work introduced by O'Connor [91] investigated the retrieval of answer sentences and Al-Hawamdeh and P. Willett [5] proposed ranking full paragraphs within a document based on their relevance to the user's query using the frequency of query terms. However, in this dissertation we define passage retrieval as identifying a span of text from a single sentence to multiple paragraphs of a longer document that adequately addresses the information need of a non-factoid query while containing little extraneous information. This location and length ambiguity is not encountered in traditional fact based QA such as TREC-QA [131] and SQuAD [101], where the relevant information is contained within a span of a few tokens. Recent datasets such as MSMARCO [89], Yahoo Community QA [113], and others have been introduced to evaluate methods for this answer passage task.

Khalid and Verberne [56] examine two approaches for overcoming the difficulty of capturing a passage of unknown length within a longer text. Using two approaches of fixed and sliding windows, where a fixed window divides a document into disjoint documents and a slider window incrementally advances over sentences. They observe that a sliding window over sentences within a document provides a better segmentation of a document for passage retrieval, further establishing the approach introduced by Callan et al. [17]. In both approaches, BM25, term frequency inverse document frequency, and KL-divergence approaches are used to retrieve candidate passages.

Understanding the need to model passages differently than full text documents, Liu and Croft [70] propose adjusting a standard retrieval approach, a language model, to better fit text distributions over passages. This finding of passages possessing unique relevance signals when compared to topical ad-hoc retrieval is a driving motivation for incorporating neural approaches into answer passage retrieval. We establish in previous work [20] that neural methods effective for topical ad-hoc retrieval are not effective when applied to passage level text.

2.3 Neural Retrieval Models

While the previous learning to rank approaches use a small number of parameters to model relevance over handcrafted features, neural models use a large number of parameters over minimally processed features, and thus are suited to the answer passage retrieval task given their large number of non-linear transformations and parameters to manipulate the initial input data. Inspired by the neuronal activity in a brain in the form of a perceptron [102], these models consist of multiple levels that incrementally map initial raw input into incremental more refined, or abstract, representations through perceptron-like modules. The non-linear transformations between each layer allow multiple distinct inputs to map to the same region in the next layer.[86]. In some constructions, these models can even act as universal function approximators [47]. These properties result in neural models which are able to take in a large volume of information and iteratively learn what is important by adjusting their internal layers, removing the need to identify salient input features or effective kernels for other learning to rank approaches. While there are a number of different architectures each with associated biases [108, 36, 126], they all demonstrate the flexibility discussed above.

With the demonstrated effectiveness of the word embeddings produced from RANDWALK or GloVE [96, 7], neural models are able to leverage this wealth of input data to achieve state of the art results on a variety of tasks overlapping with answer passage retrieval. Iyyer et al. [50] leverage this for a quiz bowl task by using a recurrent neural network that captures temporal information. However, this setting does not fully represent the IR scenario due to the structure of quizbowl retrieval: passage length queries and documents no longer than a few tokens. Severyn and Moschitti [108] adapt this approach by implementing a convolutional neural network (CNN) for traditional factoid QA over term word embeddings and term frequency features. Shortly thereafter, several neural architectures for retrieval were introduced, and Guo et al. [39] propose a paradigm to view these architectures. They are a combination of *interaction* based, where query and documents are combined at an early layer to model relevance, and *representation* based that model the query and document independently before combining them for a final relevance score. This paradigm incorporates any base building layer, be it recurrent models such as CNNs, Long Short Term Memory models (LSTM), or multilayer perceptrons (MLP). There also exist models that contain both of these paradigms such as the Duet model introduced by Mitra et al. [84]. This architecture has two subnetworks that are interaction and representation based respectively.

A recent development by Vaswani et al. [126] in neural machine translation introduced the transformer architecture. This architecture removes the need for convolutional or recurrent layers to model text by incorporating a large number of small attention heads to self attend over each layer. The ability of the transformer to incorporate large amounts of information within its parameters has allowed it to outperform previous neural baselines. One such version of this is Bidirectional Encoder Representations from Transformers (BERT) [25]. BERT's representations achieve state of the art performance on a variety of tasks by extensively pre-training a neural language model. Neural retrieval models have incorporated this pre-training with success by adjusting BERT to a retrieval purpose using the model as a black box phrase embedding and training an MLP as the final retrieval layer [73]. Furthermore, Yang et al. [144] incorporate this training paradigm and show that this setup is effective for both ad-hoc and passage retrieval. While effective, we do not include BERT based models in the experiments within this dissertation as the bulk of work was done prior to the introduction of the first evidence of the efficacy of this approach. Furthermore, the work in this dissertation presents an alternative approach to very large pre-trained language models, relying on efficient use of parameters to achieve the same end goal.

2.3.1 Training Deep Neural Models

As discussed, the data sparsity problem is exacerbated in passage retrieval due to the precise information needs of the query as not only must the model learn general topical relevance, it must also distinguish candidate passages that only contain partial information. Thus, transfer learning and domain adaptation approaches provide a promising avenue of leveraging a general relevance function for the precise relevance need of each collection. The core approach is to reduce the distance between some portion of a neural representation to allow the relevance function to effectively discriminate relevant and nonrelevant passages from multiple collections. A common component of this is adversarial networks, which surfaced shortly after they were introduced in the generative adversarial network (GAN) model. Goodfellow et al. [32] present a generative model that learns a distribution $p_G(x)$ that matches a true distribution $p_{data}(x)$. The generative model receives training updates through a joint loss function shared with an adversarial network, the discriminator, that learns whether a sample is from $p_G(x)$ or $p_{data}(x)$ as a binary classification problem. The generator is penalized when the discriminator can successfully classify the sample origin, framing the relationship as a minimax game. While initially proposed for generating continuous data, Donahue et al. [27] extend this work by learning an encoder that maps the data to the latent space $\mathbf{z}$. They show that this can learn useful features for image classification tasks without the need for supervised training.

Tzeng et al. [125] first propose a form of domain agnostic representation via *domain confusion*, where the maximum mean discrepancy between the final layers of two identical networks over different domains is directly minimized. With the introduction of adversarial agents, Ganin et al. [31] approach the same task of domain agnostic representation by using an adversarial discriminator. The representation of the main network is forced away from a domain specific representation by reversing the gradient updates outside of the adversarial discriminator.

As previous methods used shared weights for both domains, Rozantsev et al. [104] expand on this work showing that unpairing a portion of the classification model, with only a small number of parameters shared prior to input into the final layers, can lead to effective adaptation in supervised and unsupervised settings. Recently, Tzeng et al. [124] have represented a number of past domain adaptation works in a unified framework, referred to as *Adversarial Discriminative Domain Adaptation*, that captures previous approaches as special cases and encompasses a GAN loss into the training of the classifier and adversarial discriminator. This methodology achieves robust domain agnostic models over computer vision collections.

2.4 Reinforcement Learning Based Retrieval

The common evaluation metrics of an IR system often rely on a positive sensitive function such as precision, recall, or normalized discounted cumulative gain. As these approaches are non-differentiable, a neural network is not directly optimizable given standard smooth convex function requirements. Therefore, a common approach is to approximate the IR metric via softmax, hinge loss or other heuristic smooth convex functions [73, 83, 24]. While effective, this approach suffers an inherent flaw as these losses do not take into account the entire ranked list. In doing so, they suffer a fundamental calibration flaw such that minimizing the heuristic loss does not mean the IR metric has been optimized [16]. Reinforcement learning (RL) is a framework that allows a model to directly optimize a non-convex discrete IR metric based on its actions rather than indirectly through a loss function. In doing so, RL trained IR models, or policies, no longer suffer the calibration flaw. By modeling the retrieval process through a Markov Decision Process (MDP), the IR model now ranks documents via actions that then produce a score, or reward. Discussed below, this approach has been introduced within IR by a variety of works with success over handcrafted features.

Guan et al. [38] use RL in a session query reformulation environment while Zhang et al. [156] introduce a partially observable MDP framework to rerank documents based entirely on document ids and search log click information which is able to learn an effective reranking function despite no access to document content. Wei et al. introduce MDPRank [136], the framework most similar to our task, that treats each query and a list of documents as a single episode, and each action selects the most relevant document of the yet to be ranked documents within that list, which is competitive with traditional convex optimization methods.

Hu et al. [48] incorporate a variation of actor critic called deep deterministic policy gradient for the task of e-commerce sessions. This approach further reduces the variance by allowing the agent to operate only over the state space [67]. In addition, through a novel construction of an e-commerce session MDP, the critic component that relies on TD learning is able to perform a full backup.

However, these methods are used in a conventional learning to rank setup with highly salient domain specific features that include term overlap, document length, and BM25 scores among others. The closest work with respect to directly operating over terms is by Xia et al. [140] where the authors use an MDP to diversify search results. Rather than using learning to rank features, they independently model query embedding and document embeddings via doc2vec, a similar paradigm to word2vec [80] where a model tries to predict neighboring sentences as contexts. While allowing for a distributed representation, doc2vec based representations are not effective for IR purposes [42]. Furthermore, the policy is independent of these embeddings, and consists of two matrices,

$$\mathbf{Vq},\ \mathbf{V} \in \mathbb{R}^{K\times L}, \mathbf{Ud}, \mathbf{U} \in \mathbb{R}^{K\times L}$$

where $\mathbf{q}, \mathbf{d}$ represents query and document embeddings of dimension K, and the learned policy parameters are $\mathbf{V}, \mathbf{U}$. While Wang et al. [130] do incorporate RE-

INFORCE with a CNN based policy, the policy quickly diverges and is not able to retrieve effectively and instead rely on a paired model trained via a smooth convex loss.

2.4.1 Reinforcement Learning for Natural Language Processing

While not specifically addressing document retrieval, the common NLP RL environment of learning actions over text results in a significant overlap with IR methods. We therefore provide a short overview of RL based methods for NLP to contextualize our contribution and the current state of the field for other text based tasks.

For machine translation, RL has been used to overcome the non-differentiable objective function, BLEU. Rather than trying to optimize at an individual word level using a heuristic convex loss approximation, Wu et al. [138] propose a REINFORCE based optimization of a neural machine translation system. Subsequently, Choshen et al. [19] discuss the shortcomings of REINFORCE's high variance when applied to the machine translation task, and we observe this same issue when applying REINFORCE to deep neural IR models in Section 5.2.

In a similar situation to machine translation, summarization shared a non-differentiable objective function. Paulus et al. [95] introduce a REINFORCE based model that directly optimizes this objective. Li et al. [65] expand on this work and introduce an actor critic based reinforcement learning approach for abstractive summarization.

Addressing work in reading comprehension, Wang and Jin [132] incorporate an Actor-Critic agent to learn a policy for multi-step coarse to fine question answering where the largest state in the MDP is a single document. Shen et al. [109] introduce a policy trained via REINFORCE that can decide when to terminate the multi-hop reasoning process rather than relying on a fixed number of hops to answer a question.

CHAPTER 3
ANSWER PASSAGE TEXT REPRESENTATION

As discussed, answer passage retrieval requires a richer information representation than that of traditional ad-hoc retrieval that focuses on topical relevance. In this chapter, we present two approaches to increase the effectiveness of neural models. First, in Section 3.1 we demonstrate that using locally trained embeddings in an end-to-end approach significantly improves IR over a community question answer (CQA) passage collection than previous approaches using Word2Vec [80]. Subsequently in Section 3.2 we introduce a hybrid embedding architecture that dynamically constructs local embeddings while leveraging general knowledge via GloVE representations.

3.1 Local End to End Embeddings

As word embeddings are noisy factorized pairwise mutual information matrices, the information contained is entirely dependent on the type of prose used in the collection. While this representation is useful for relevance, it discards a significant amount of information that is useful when handcrafting features for determining relevance such as *idf*, term frequency information, and distributions between relevant and non-relevant passages [26]. As this type of information is useful even for the task of passage retrieval [129], we show that an end to end framework, tying in the representation of the text embeddings to that of an IR loss function, significantly improves the performance when compared to pre-trained embeddings created over a collection an order of magnitude larger.

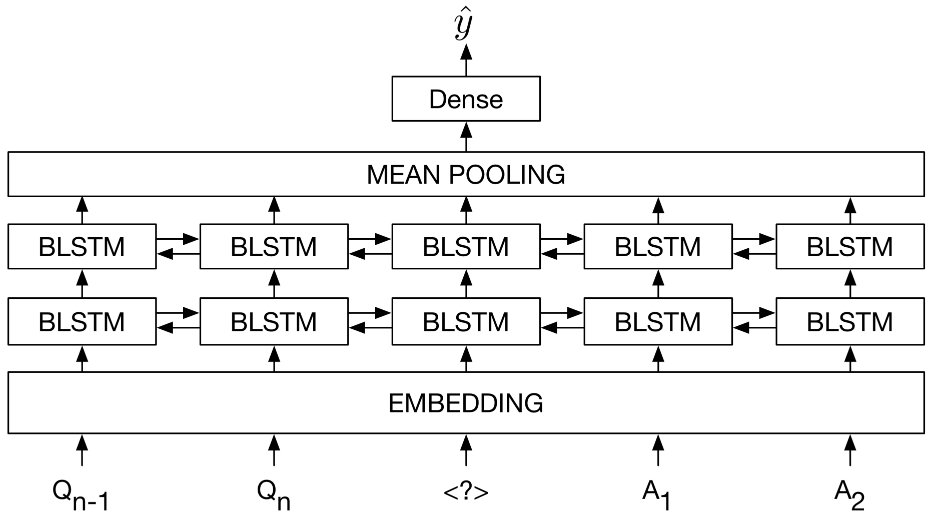


Figure 3.1: A simplified representation of the BiLSTM network with an n length question

3.1.1 A Neural Network for non factoid Retrieval

For this task, we use a variant of an LSTM network structure implemented in Wang and Nyberg [129] and Graves et al.[36]. The standard RNN architecture is constructed such that each layer not only receives input from the layer below it, but also its own output from the previous time step. LSTM units replace the standard neuron of a RNN with additional internal structures to manage vanishing and exploding gradients. These structures consist of input, forget, and output gates that manage the information flow of the cell's internal state.

We utilize a bidirectional neural network as in [36, 129], which can be viewed as inputting the sequence in reverse order to a second layer at the same level of the graph, and then merged either through concatenation or element-wise summation. The bidirectional layers for this paper were implemented via concatenation. A simplified representation of the network is shown in Figure 3.1 with the output of the network represented as $\hat{y}$.

3.1.2 Experimental Setup

3.1.2.1 Data

Tokens	Webscope L4	nfL6
Min	6	10
Max	350	79
μ	77.4	39.0
σ	62.0	13.2

Table 3.1: Statistical description of tokens per question-answer pair in nfL6 and Webscope L4 after preprocessing

The datasets used for our experiments were Yahoo's Webscope L4 and a filtered, lower quality non-factoid set created from Yahoo's general Webscope L6, named nfL6. The L4 set has been used previously [113] for non-factoid QA and is sometimes referred to as the "manner" collection. It consists of 142,627 questions, of which we select 138,340 questions that satisfy the condition of being under 351 words when combined with their corresponding answer and do not contain websites. The word limit was used as LSTM networks are not capable of capturing dependencies on arbitrary length sequences and would not be able to learn representations of greater length answers. All questions are of the manner "how {to|do|did|does|can|would|could|should}..." and are high quality. Each question contains a noun and verb, and each answer is well formed. All answers that were not the highest voted answer were removed for each question as multiple answers for a question could be correct. This was done so the network would learn to better differentiate between correct and incorrect answers and not try to learn which answers would receive the highest votes.

The nfL6 dataset, after processing, consists of 87,361 questions. Unlike L4, the questions in this dataset are more generic, such as "Why is the sky blue?" and "Why do people steal?". Furthermore, answers are not as high quality. This set was created from the lower quality L6 collection using a linear kernel with a support vector machine to remove poor question candidates. Initial training data is from UIUC's question dataset [66]. Fine grained classes of *description, manner*, and *reason* within the coarse grained class DESC were used as positive examples, with all others as negative examples. 3,500 additional training examples were attained from active training based on their distances to the hyperplane [122]. Additionally, to reduce noise, negative classifiers were trained on ENTITY, ABB, LOCATION, NUMERIC, and HUMAN classes to further reduce factoid questions in the collection.

Training, validation, and testing sets[1] for the BiLSTM implementation were created in a similar fashion to [50]. A small pool of candidate answers were collected for each question based on top results in a BM25 search.

3.1.2.2 Network Configuration

For input to the network, each question is concatenated with its answer and a `<?>` character is inserted between the two strings as shown in Figure 3.1. Incorrect answers were concatenated the same way with the question string to create negative training examples. The `<?>` character was used similar to the `<EOS>` and `<S>` mark in [114] and [129, 128] respectively. This mark signifies the transition between source and target sentences and is depicted in Figure 3.1.

The specific network configuration consisted of a 256 dimension embedding matrix initialized uniformly, which feeds directly into two 512 length BiLSTM layers with concatenated outputs. The cell activation function for the LSTM nodes is the sigmoid function, with internal gates using the *tanh* function. The output of the last BiLSTM layer is mean pooled across time steps and fed into a single dense node with a sigmoid activation function. As mentioned previously, the embedding layer is part of the network during training, and thus will change word representations to best fit the loss function.

[1]Available at https://ciir.cs.umass.edu/downloads/nfL6/

Optimization was done using the Adam algorithm [57] and trained to minimize the binary cross entropy weighted by how well the answers are separated with respect to the non-relevant training examples for each query as shown below.

$$BCE_q = \frac{1}{|q|} \sum_{\hat{y}_q \in q} (-y \log \hat{y}_q) - (1 - y_q) \log(1 - \hat{y}_q) \tag{3.1}$$

$$L = \sum_{q \in Q} (1 - (q_r - \mu_{q_{nr}})) BCE_q \tag{3.2}$$

$$BCE = \frac{1}{|N|} \sum_{\hat{y}} (-y \log \hat{y}) - (1 - y) \log(1 - \hat{y}) \tag{3.3}$$

$$\tag{3.4}$$

With Q as all questions, BCE_q as standard binary cross entropy for the question, q_r as the relevant answer score and $\mu_{q_{nr}}$ as the mean of all non relevant candidate answer scores for q. As the task cares about relative ranking over binary classifications, this scales the loss relative to the distance in scores such that weights will change based on the questions with the hardest to differentiate answers.

3.1.2.3 Evaluation

The evaluation metrics used are mean reciprocal rank (MRR) and precision at 1 (P@1) which are both common in IR and QA evaluations. Precision at 1 is a binary metric that is 1 if the correct answer is ranked highest, and 0 otherwise. The mean is then taken to evaluate performance over a collection of questions. The reciprocal rank is the multiplicative inverse of the highest ranking correct answer retrieved for a question. Thus the mean is $\frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i}$ with Q being the set of questions.

The test collection was created from pooling the top 10 results from a BM25 search for each question, and including the correct answer as the 10^{th} answer if it

Implementation	L4		nfL6	
	P@1	**MRR**	**P@1**	**MRR**
Okapi BM25	0.0783	0.1412	0.1312	0.2660
Severyn and Moschitti	0.0989	0.2434	0.1438	0.2842
Wang and Nyberg	0.4414	0.6152	0.1232	0.3271
BiLSTM	0.4752^{*}	0.6377^{*}	0.2002^{*}	0.4043^{*}
BiLSTM-Loss	$0.5157^{*\dagger}$	$0.6642^{*\dagger}$	$0.2375^{*\dagger}$	0.4219^{*}

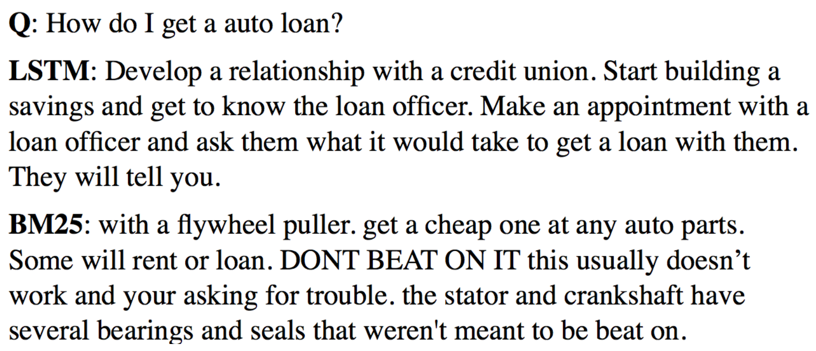
Table 3.2: Results on Webscope L4 and nfL6. Significant differences relative to Wang and Nyberg denoted by *, † denotes relative to BiLSTM (using two tailed t-test with p ¡ 0.05)

is not included in the list. These were then processed into sequences described in Section 4.2.

3.1.2.4 Results

The end to end LSTM is compared against previous deep learning implementations and the BM25 baseline in Table 3.2. BM25 was chosen for the baseline as Yih et al. [146] have shown that *tf.idf* models are a competitive benchmark. While the CNN of Severyn and Moschitti [108] fails to capture any dependency between questions and answers in the L4 data, the BiLSTM implementations are successful in learning a relation between them. Furthermore, the end to end local embeddings significantly improves results over using an independently trained word embedding matrix without the need of an additional model to incorporate term frequency information and BiLSTM layer as used in Wang and Nyberg [128]. This performance difference becomes more apparent when the language gap between training and testing of the embedding matrix grows. The nfL6 dataset contains slang and abbreviations not present in typical training text, which causes a hyperparameter tuned BiLSTM implemented to perform well below the modifications used in this paper.

The effect of the rank sensitive loss function results in significant improvement as well, referenced as BiLSTM-Loss in Table 3.2. In training and evaluation, the



Q: How do I get a auto loan?

LSTM: Develop a relationship with a credit union. Start building a savings and get to know the loan officer. Make an appointment with a loan officer and ask them what it would take to get a loan with them. They will tell you.

BM25: with a flywheel puller. get a cheap one at any auto parts. Some will rent or loan. DONT BEAT ON IT this usually doesn't work and your asking for trouble. the stator and crankshaft have several bearings and seals that weren't meant to be beat on.

Figure 3.2: Example of a question in which the BiLSTM implementation successfully returns the correct answer while BM25 does not.

network's range for $\hat{y}$ is dependent on the question rather than consistently centered around one point in [0,1]. As the task focuses on the relative rankings of candidate answers, embedding weights are updated based on the difference of non-relevant and relevant answers instead of solely based on their respective entropy.

Lastly, the performance of BM25 is included to further demonstrate the poor performance of BM25 on both answer passage collections. This reflects in the results of the Severyn and Moschitti model [108] as the use of term overlap features appended to the output of a hidden layer does not improve results over the sequence based approach of the LSTM. While RM3 is a higher performing method than BM25, it acts as a function over a function on the collection, and is not directly comparable to the baselines used in this evaluation.

3.2 Hybrid Architecture

While the local embeddings significantly improve the performance of the neural models for answer passage retrieval, this results in a significant cost as retraining embeddings to reflect a new collection can consistently improve performance; however, it is impractical to create new local embeddings at run time as recent methods [26, 80], including the end-to-end approach introduced in Section 3.1.1 rely on a time consuming optimization process requiring large amounts of data. Thus, we introduce a hybrid based embedding approach that leverages pre-trained embeddings, and further allows for dynamic construction of local representations. This is done via leveraging the same LSTM network to build phrase level representations of both standard word embeddings as well as with the flexibility of a character n-gram based approach as seen in the DSSM and the OCR degraded text task. We adapt the fixed window of the trigram hashing in DSSM by using varying length convolutional filters to aggregate multiple length character n-grams and then sequentially building sentence and passage embeddings using a recurrent network. This approach produces a network that (1) is robust to degradation in collection quality (2) maintains performance on high quality collections where standard character based approaches fail to perform, and (3) does not require the expensive process of retraining embeddings for each collection.

3.2.1 Model

We propose a hybrid CNN-LSTM model that not only constructs passage level representations from word embeddings, but simultaneously builds an identical representation from a separate character representation. This hybrid approach allows the network to leverage the information contained in pretrained word embeddings while simultaneously using the character subnetwork to construct collection specific representation in its hidden layers. A simplified representation of the model is shown in Figure 3.3 with the three key components illustrated. As each component plays a critical role in determining the relevance of a candidate passage, the remainder of this section explains in detail the construction and motivation for each layer's architecture within the model.

3.2.1.1 Character Embeddings

As opposed to previous work in IR with neural networks [79, 118, 108, 28], our model's input consists of an additional sequence of characters rather than words alone. The advantage of processing text from a character level representation is that it allows

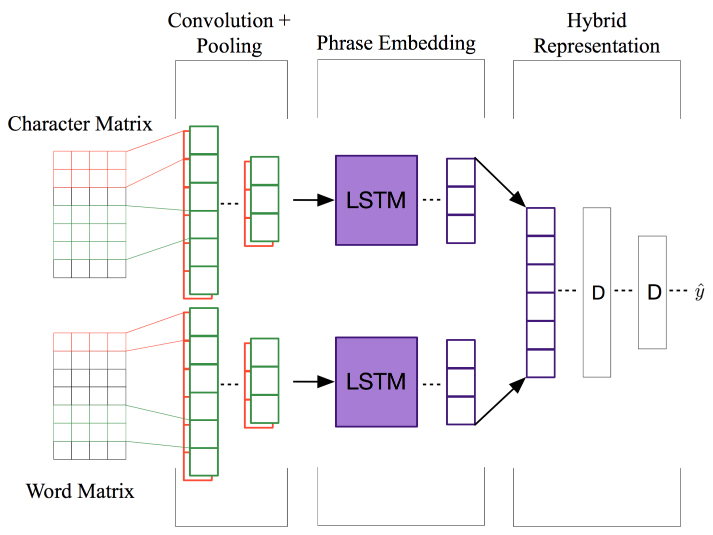


Figure 3.3: A compressed representation of the Hybrid architecture.

for the upper layers to learn a word representation tailored to the collection. Given a sequence of characters from a passage, we concatenate their embeddings into a $k \times l$ matrix where k is the dimension of the embedding and l is the length of the passage. The embeddings were created via the approach introduced by Mikolov et al. [80] with a skipgram window of 5 to create the character embeddings.

As in [158], the alphabet consists of 70 characters, 26 lowercase English letters, 10 digits, and 33 other characters. Characters not contained in the alphabet, including spaces, are represented as k dimensional zero vectors. Uppercase letters were converted to lowercase as they did not improve performance, and k was chosen to be 20. The 33 other characters are shown below:

```
-,;.!?:’/\|_@#$%^&*~{}`+-=<>()[]
```

3.2.1.2 Embedding-level Convolutional Layer

One can view a convolution as sliding a fixed width filter, **f**, over the sequence of character embeddings. The filter is constant as it slides over the text, and its weights are updated via backpropagation to identify specific features. This allows the model to transform the input of individual characters into words, or words into short phrases based on common, repeated patterns within the fixed width filter. In the model, the character or word sequence is converted into a passage matrix, $\mathbf{P} \in \mathbb{R}^{k \times l}$, where we convolve $\mathbf{P}$ with a filter $\mathbf{f} \in \mathbb{R}^{k \times w}$ with w as the width of the filter and has the same dimension as the embeddings. The model in this paper uses the activation function *tanh* which allows for faster convergence compared to the standard logistic function. In order for a convolutional layer to recognize a variety of features, each layer uses a number of filters within [100,1000] for typical IR and NLP applications. Thus, the output of a convolutional layer is a matrix, $\mathbf{F} \in \mathbb{R}^{c \times k \times l}$, where c is the number of filters chosen.

After performing the convolution, temporal max-pooling is performed to select the most salient features over a portion of $\mathbf{F}$. This eliminates non-maximal values and reduces the dimensionality and number of parameters needed for the network via non-linear down-sampling as in [108, 118].

These embedding level convolutions are then passed into the BLSTM structure introduced in Section 3.1.

3.2.1.3 Joint Representation

As the Hybrid model consists of two unique substructures, each processing word and character embeddings respectively, the output of the BiLSTM layers are mean pooled and concatenated across time steps to produce a single vector, $\mathbf{v} \in \mathbb{R}^{n}$ which can be viewed as the embedding of the entire phrase. The combination of character and word level embeddings allows this model to leverage two unique representations, the character subnetwork is directly tailored to the collection while the word subnetwork aids in generalization.

It is then fed into three dense layers to learn the interaction between word and character phrases. The final dense layer maps to a scalar value, $\hat{y}$ in Figure 3.3, that represents the relevance of the input.

3.2.1.4 Attention Mechanism

While LSTM networks are able to store internal states across sections of a sequence, they cannot capture arbitrary length dependencies that span across longer passages [36]. In order to persuade the hidden states of the model to focus on information relevant to the query contained in candidate passages, we use an attention mechanism by allowing the hidden layers of a network to compare query and document text when learning abstract representations. This reduces the information load on the network as the parameters are able to focus on modeling this interaction rather than each text individually.

With a variety of attention mechanisms available in previous work [147, 79, 107], we adopt a method that primes the network similar to machine translation [129] to aid in the LSTM capturing long term dependencies. Given a question-passage pair below,

$$q_1, q_2, \ldots, q_n \;\; <?> \;\; a_1, a_2, \ldots, a_n$$

The network iterates over the query until it reaches the $<?>$ token, at which point it receives a candidate answer. As discussed in Section 5.1, this method allows the network to imprint query specific terms and topics within the cell states that produce selective activations for related information in candidate passages. By priming the network, the recurrent layers learn to model intermediate representations of relevance rather than waiting to introduce query similarity within the final few layers [118, 77, 108].

3.2.2 Experiments

Following the local embedding experiments, we use the same two CQA collections, Yahoo's Webscope L4 and nfl6, along with more difficult web answer passage collection, called WebAP[2] that has a small number of queries. In contrast to the

[2]https://ciir.cs.umass.edu/downloads/nfL6

Layer		Char	Word	BiLSTM
Conv	w	[6,7]	[1,2]	
	c	[450,525]	[600,700]	
	σ	tanh	tanh	
Conv	w	[3,4]		
	c	[225,300]		
	σ	tanh		
Conv	w	[3,4]		
	c	[225,300]		
	σ	tanh		
BiLSTM	l	[350,350]	[550,550]	[600,600]
Dense	l	500	500	500
Dense	l	300	300	300
Dense	l	1	1	1

Table 3.3: Architecture of the three networks evaluated. Char and Word represent the components used for processing character and word embeddings respectively in the the Hybrid model; w = filter width, c = number of filters, σ = activation function, l = layer dimension.

previous CQA collections, the queries are more open ended and can have a variety of passages that are all relevant. An example of this is seen in the query "*Describe the history of the U.S. oil industry*". Non-relevant portions of each document are split into non-overlapping random length passages. This was done to avoid the network learning certain length passages as non-relevant. Candidate passages with a word count greater than 4000 were removed from the collection as they significantly increased the memory footprint of the models when training. This did not impact the results as they were labeled non-relevant and consistently ranked last during testing. Training, validation, and testing sets were created via a 64-16-20 split. Detailed statistics for each collection are shown in Table 3.4.

.

Tokens	Webscope L4	nfL6	WebAP
Min	6	10	2
Max	897	722	10885
μ	91.9	50.9	61.2
σ	99.7	25.6	58.1

Table 3.4: Statistical description of tokens per question-answer pair in nfL6, Webscope L4, and WebAP collections after preprocessing.

3.2.3 Baselines

We compare our Hybrid model to previous deep learning implementations and BM25. As little neural work has been done specifically on the answer passage retrieval task, we include additional networks used for factoid QA. We use Wang and Nyberg's [129] non-factoid BiLSTM model prior to boosting, Tan et al.'s [118] factoid QA CNN-LSTM model, and Severyn and Moschitti's Convolutional Deep Neural Network (CDNN) [108]. We also evaluate the individual word and char components of the Hybrid model to isolate the performance difference of word and character embeddings denoted as W-Hybrid and C-Hybrid respectively. Exact configurations of the BiLSTM, C-Hybrid, and W-Hybrid models are shown in Table 3.3. We also include DSSM [49] as a competitive character level baseline and DRMM [39] as a competitive neural architecture for document retrieval. All word embedding based neural models are evaluated both on embeddings training locally on the collection and pre-trained embeddings from Google's 300 dimension word2vec model[3]. Character embeddings are initialized from Wikipedia's 05-2015 data dump[4].

3.2.4 Evaluation

Mean reciprocal rank (MRR) and precision at 1 (P@1) are used for evaluation. Both metrics are common in IR, and reflect the small number of relevant answer

[3]https://code.google.com/archive/p/word2vec/

[4]https://dumps.wikimedia.org/enwiki/20160501/

Method	Embedding	L4	nfl6	WebAP
BM25	–	.1412	.2660	.4120
DSSM	–	.2477	.2576	.3127
DRMM	–	.3291	.3350	.4064
Tan et al.	–	.4217	.3934	.3612
Tan et al.	–	.2434	.2842	.3834
BiLSTM	Local	.6329	.4710	.4618
	Pretrained	.6129	.4287	.4502
W-Hybrid	Local	.6206	.5327	.4136
	Pretrained	.6190	.5236	.4411
C-Hybrid	Local	.5801	.4987	.5148
	Pretrained	.5798	.4983	.5150
Hybrid	Local	.6241	.5429	.4410
	Pretrained	.6407*†	.5433*†	.4716

Table 3.5: MRR performance of networks on the three test collections. Local and Pretrained refer to the embedding types used. * denotes significance with $p < .05$ with respect to baselines using two tailed t test. † denotes same significance against subnetworks (W/C-Hybrid)

passages as well as the importance on the first passage retrieved for mobile and audio search. Pooling was done in a similar way to the previous section, but increased to 100 for the WebAP collection to reflect the larger number of candidate passages available.

3.2.5 Setup and Training

Our CNN-LSTM based networks were optimized via RMSprop [120] over a binary cross entropy function. The networks were trained until the metrics over the validation set stopped improving.

3.2.6 Results and Discussion

In this section, we first evaluate the performance of the Hybrid model with respect to the baselines. In order to examine the impact of the additional character structure, we break apart the Hybrid model and evaluate the C-Hybrid and W-Hybrid subnetworks independently. Lastly, the comparative performance of local and pretrained embeddings are discussed in relation to the models. The results for each of these are shown in Table 3.5. Of particular note is the poor performance of the traditional factoid or sentence QA models, Severyn and Moschitti's CDNN [108] and Tan et al. cosine similarity based approach [118]. While both of these models perform close to state of the art on WikiQA and TREC QA, they achieve significantly worse results on the answer passage retrieval task. While not benchmarked, the architecture introduced by Meng et al. [77] possesses a similar structure to CDNN and thus would not perform well due to the shared structure and lack of temporal structure.

3.2.6.1 Hybrid Embedding Effect

The Hybrid model outperforms the baselines on all but the WebAP collection. The close performance on L4 when compared to the BiLSTM model can be attributed to the language contained in the L4 collection. Compared to nfl6, both queries and answer passages contain significantly less slang, improper syntax, and more consistent sentence structure. The lack of improvement suggests that the convolutional layers do not provide any additional benefit when the collection consists of well formed passages. This is reinforced by the drop in performance on all recurrent word embedding based models moving from L4 to nfl6. Both the Tan et al. and BiLSTM models are most impacted by the lower quality collection. However, the character based DSSM, as well all models with a convolutional component are more robust to this degradation in quality. In particular, the Hybrid model is shown to be the most adaptable to this, achieving 0.3516 P@1 and 0.5433 MRR utilizing pretrained embeddings on the noisier nfl6 collection.

The performance over WebAP highlights the weakness of neural models. Through the lens of BM25, the baseline DSSM, DRMM, Tan et al. and CDNN models all fail to outperform the *tf.idf* baseline. Although the Hybrid model has somewhat better performance than the BiLSTM model, there is little difference between their scores across local/pretrained embeddings. As the WebAP collection only has 82 queries, with an average of 97 graded passages per query, this prevents the network from seeing a large portion of the word embedding space as one cannot increase negative sampling without performance cost if the relevant and non-relevant passages are from different distributions [135]. Thus, at testing time the model often sees new vocabulary and passages unseen during training. Just as in the nfl6 dataset, character embeddings bridge this gap by allowing almost all characters to be seen during training, and the convolutional layers allows for small morphological differences to exist in the same area of the manifold. This is reflected in the C-Hybrid component of the Hybrid model outperforming both a standard BiLSTM and the W-Hybrid on the WebAP collection.

3.2.6.2 Compositional Impact

To view the additional information gained by including character embeddings that is omitted from the word level networks, we evaluate the individual components designated as W-Hybrid and C-Hybrid consisting of only word and character embedding inputs respectively. Examining W-Hybrid's metrics suggests that the convolutional filters learned are somewhat noisy, resulting in lower performance compared to the LSTM-only baseline as the attention component of the models are in the upper LSTM layers. However, the addition of the character component provides missing information to allow the Hybrid model to outperform all baselines, overcoming the reduced attention ability of the individual CNN-LSTM component interactions. This compounding effect is present in both the L4 and nfl6 collection but does not pertain to the WebAP collection. As mentioned in the previous section, the small amount of training examples allows the C-Hybrid component to achieve the highest metrics regardless of the embedding origin. This discrepancy can be attributed to the training process, where the weights associated with the word models converge much faster than the character based network. As such, the lexical gap between training and test sets in WebAP is exacerbated by the reliance on the quickly converging word network despite the addition of the character network.

3.2.6.3 Local vs Pre-trained Embeddings

Viewing the results from an embedding initialization perspective, conventional word based models drop in performance when using pretrained embeddings on all collections. As discussed in Section 3.1.2.4, updating word embeddings during training allows for a richer representation for the network to use in the hidden layers. The collection least effected by this drift in information from pretrained to local embedding is the WebAP collection. We attribute this to the lower volume of training examples seen compared to the L4 and nfl6 datasets.

Unlike the BiLSTM network, implementing a convolutional layer as input over the word embeddings causes the upper layers to become less sensitive towards the type of embedding used. However, only the Hybrid model has the most consistent performance on the pretrained embeddings, significantly outperforming models using local embeddings. The Hybrid model allows for this robustness to embedding source by dynamically leveraging the character embeddings to bridge the gap between word embedding initializations. This is exemplified on noisier collections such as nfl6, where even the W-Hybrid model suffers when moving from local to pretrained embeddings.

3.3 Discussion Of Negative Results

Model	**Trained**	**Transfer**	**P@1**	**MRR**
BM25	N/A	WebAP	.3000	.4120
BiLSTM [129]	L4+nfl6	WebAP	.0941	.2116
W-Hybrid	L4+nfl6	WebAP	.1176	.2718
C-Hybrid	L4+nfl6	WebAP	.1602*	.2937*
Hybrid	L4+nfl6	WebAP	**.1836***	**.3115***

Table 3.6: Performance of networks cross-trained on the yahoo CQA data and evaluated on the WebAP collection. BM25 score is included for reference. * denotes significance with $p < .05$ with respect to baselines using two-tailed t test.

As the Hybrid architecture discussed in Section 3.2 involves a substantial amount of components, we discuss alternative constructions or hyperparameters which failed to achieve the same level of performance. We initially constructed the architecture such that the character and word convolutions were pooled together prior to constructing the phrase embedding. While the model still converged, its performance on L4 was comparable to the base W-Hybrid in Table 3.5. We attribute this to the learning rate between the Character based representation and the Word components. As observed in Figure 3.4, the word component is able to quickly achieve an effective relevance function as opposed to the character component. This characteristic could then result in the character information being discarded in the upper LSTM components given the quick convergence of the word level information. Only after separating them to individual components did we observe the performance gained

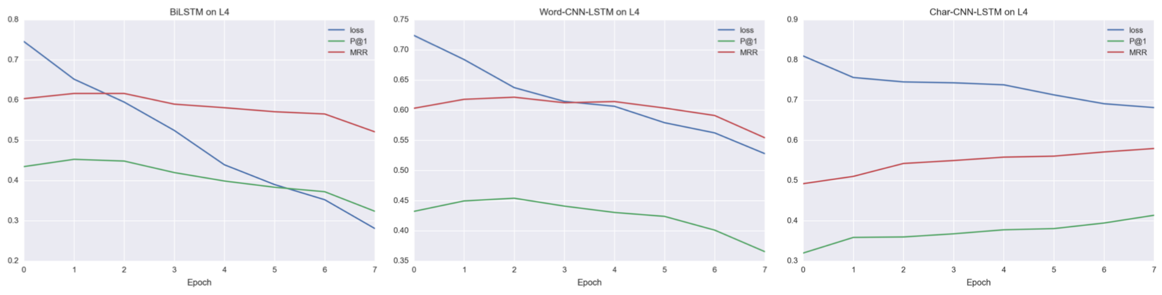


Figure 3.4: A compressed representation of the Hybrid architecture.

from the character level information. As the same learning rate was used for all parameters of the model, possibly setting individual learning rates for each component or pre-training the character subnetwork might also be a viable solution to the above architecture not performing.

CHAPTER 4

INCORPORATING ADDITIONAL INFORMATION FOR RETRIEVAL

The class imbalance in IR collections is one of the core challenges of training a successful neural retrieval model. This imbalance results in only a handful of relevant documents for each query, which results in a poor setting for neural models. As seen in all current neural retrieval models [73, 83, 40], as well as the work introduced in Chapter 2, training neural models involves heavily under-sampling negative documents and over-sampling relevant instances. While selective over-sampling results in an effective model for reranking, this biases the model towards performance on a certain area of the collection manifold and does not optimize performance over the whole collection [84]. As discussed in Wang et al. [133], this approach results in discarding potentially informative negative samples, and over-sampling relevant documents via increasing the presence of the minority class contributes redundant data at the risk of overfitting.

Therefore, in this Chapter we address two approaches to remedy this issue. In Section 4.1 we introduce a method that improves robustness when a neural model is exposed to a new subdomain of a collection unseen during training. Following that, Section 4.2 provides a policy based method of meta-learning that strives to expose a neural retrieval model to as much of the collection as possible while still maintaining IR performance.

4.1 Ensuring Subdomain Robustness

Neural IR models typically learn to distinguish between the input feature distributions corresponding to a relevant and a less relevant query-document pair by observing a large number of relevant and non-relevant samples during training. However, as Mitra et al. [83] discuss, the ability to learn new features may come at the cost of poor generalization and performance on subdomains not observed during training. The model, for example, may observe that certain pairs of phrases—, "Theresa May" and "Prime Minister"—co-occur together more often than others in the training corpus.

During the training of these models, as the neural model continually sees the same relevant document, the model may conclude that it is more important to learn a good representation for "Theresa May" than for "John Major" based on their relative frequency of occurrences in training queries and document. While these correlations and distributions are important if our goal is to achieve the best performance on only a portion of the collection, the model must learn to be more robust across subdomains if we instead care about "out of box" performance on new queries that cover a larger portion of the collection. In contrast, traditional term based retrieval models and LTR models based on aggregated count based features—that make fewer distributional assumptions—typically exhibit more robust cross subdomain performance.

We propose training deep neural ranking models using an adversarial component to intentionally ablate information local to only single domains that is not vital to determining relevance.

4.1.1 Cross subdomain regularization using adversarial learning

We train our neural ranking model on a small set of subdomains and evaluate its performance on held out subdomains. During training, we combine our ranking model with an adversarial discriminator that tries to predict the subdomain of the training sample based on the representations learned by the ranking model.

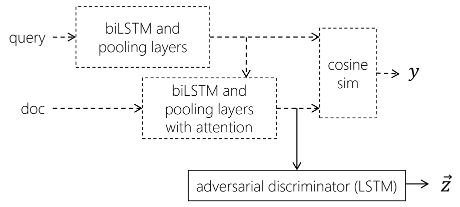

query
biLSTM and pooling layers
doc
biLSTM and pooling layers with attention
cosine sim
y
adversarial discriminator (LSTM)
$\vec{z}$

Figure 4.1: CosSim w/ adversarial discriminator

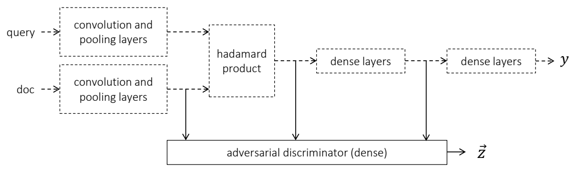

query
doc
convolution and pooling layers
convolution and pooling layers
hadamard product
dense layers
dense layers
y
adversarial discriminator (dense)
$\vec{z}$

Figure 4.2: Duet-distributed w/ adversarial discriminator

Figure 4.3: Cross subdomain regularization of the two baseline models—CosSim and Duet-distributed—using an adversarial discriminator. The discriminator inspects the learned representations of the ranking model and provides a negative feedback signal for any representation that aids subdomain discrimination.

The motivation of the adversarial discriminator is to force the neural model to learn subdomain independent features that are useful to estimate relevance. We propose using an adversarial agent to force the features learned by the ranking model to be subdomain agnostic by shifting the model parameters in the opposite direction to subdomain specific spaces on the manifold. This cross subdomain regularization via subdomain confusion [124] can be represented as a joint loss function:

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_{\text{rel}}(q, doc_r, d_{nr}, \theta_D, \theta_{\text{rel}}) \\ &\quad + \lambda \cdot \Big(\mathcal{L}_{\text{adv}}(q, doc_r, \theta_D) + \mathcal{L}_{\text{adv}}(q, doc_{nr}, \theta_D)\Big)\end{aligned} \tag{4.1}$$

where $\mathcal{L}_{\text{rel}}$ is a relevance based loss function and L_{adv} is the adversarial discriminator loss. q, doc_r, and doc_{nr} are the query, the relevant document, and the non-relevant documents, respectively. Finally, θ_{rel} and θ_D are the parameters for the relevance and the adversarial models, respectively. λ determines how strongly the subdomain confusion loss should impact the optimization process. We treat it as a hyper-parameter in our training regime. The ranking model is trained on a set of subdomains $D_{\text{train}} = \{d_1, \dots, d_k\}$ separate from the set of held out subdomains $D_{\text{test}} = \{d_{k+1}, \dots, d_n\}$ on which it is evaluated.

The discriminator is a classifier that inspects the outputs of the hidden layers of the ranking model, and tries to predict the subdomain $d_{\text{true}} \in D_{\text{train}}$ of the training sample. The discriminator is trained using a standard cross-entropy loss.

$$\mathcal{L}_{\text{adv}}(q, doc, \theta_D) = -\log\big(p(d_{\text{true}}|q, doc, \theta_D)\big) \tag{4.2}$$

$$p(d_{\text{true}}|q, doc, \theta_D) = \frac{exp(z_{\text{true}})}{\sum_{j \in D_{\text{train}}} exp(z_j)} \tag{4.3}$$

Gradient updates are performed via backpropagation through all subsequent layers, including those belonging to the ranking model. However, as proposed by Ganin et al. [31], we utilize a gradient reversal layer. This layer transforms the standard gradient, $\frac{\delta L_{\text{adv}}}{\delta \theta}$ to its additive inverse, $-\frac{\delta \mathcal{L}_{\text{adv}}}{\delta \theta_{\text{rel}}}$. This results in θ_{rel} maximizing the subdomain identification loss, while still allowing θ_{D} to learn to discriminate subdomains. While not directly optimized, this can be viewed as modifying (1) via a sign change for L_{adv}.

4.1.1.0.1 Passage Retrieval Models We evaluate our adversarial learning approach on the passage retrieval task. We employ the neural ranking model proposed by [118]—referred to as CosSim in the remaining sections—and the Duet model [84] as our baselines. Our focus in this paper is on learning subdomain agnostic text representations. Therefore, similar to Zamani et al. [150], we only consider the distributed sub-network of the Duet model.

The CosSim model is an LSTM-based interaction focused architecture. We train the CosSim model in the same manner as [118], with a margin of 0.2 over a hinge

		CosSim		Duet-Dist.	
source → *target*	Size	Original	Adv	Original	Adv
All→All	142627	0.6188	**0.6214**(+.4%)	**0.6136**	0.6061(-1%)†
All*→Sports	139000	0.5194	**0.5925**(+12%)†	0.4567	**0.5011**(+10%)†
All*→Home	133372	0.5275	**0.5433**(+3%)†	0.5285	**0.5457**(+3%)†
All*→Politics	138739	0.5101	**0.5507**(+8%)†	0.5291	**0.5342**(+3%)†
All*→Travel	140150	0.4486	**0.4723**(+5%)†	0.4196	**0.4532**(+8%)†

Table 4.1: Performance across L4 topics under MRR. All* is the entire L4 collection with target topic removed. † represents significance against non adversarial model ($p < 0.05$, Wilcoxon test)

loss function. The Duet-distributed is trained by maximizing the log likelihood of the correct passage, as originally proposed in [84]. Similar to [87], we adapt the hyper-parameters of the Duet model for passage retrieval. The output of the Hadamard product is significantly reduced by taking the max pooled representation, the query length is expanded to 20 from 8 tokens, and the max document length is reduced to 300 from the original 1000 tokens.

As opposed to past uses of adversarial approaches [31, 45, 124], ranking requires modeling an interaction between the query and the document. As shown in Figure 4.1, the adversarial discriminator in our setting, therefore, inspects the joint query-document representation learned by the neural ranking models. For deeper architectures, such as the Duet-distributed, we allow the discriminator to inspect additional layers within the ranking model, as shown in Figure 4.2.

4.1.2 Experiments

4.1.2.1 Data

L4 We use Yahoo's Webscope L4 high quality "Manner" collection [113]. For evaluation and training, all answers that were not the highest voted were removed from the collection to reduce label noise during training and provide a better judgment of performance during evaluation. Training, development, and test sets were created

	CosSim		Duet-Dist.	
source → *target*	Original	Adv	Original	Adv
(InsuranceQA, L4)→ WebAP	0.2410	**0.3873**	0.1250	0.4567
(InsuranceQA, WebAP)→ L4	0.2957	**0.4335**†	0.0758	0.193
(L4, WebAP)→ InsuranceQA	0.4267	**0.4717**†	0.0489	0.1473

Table 4.2: Performance across collections, where metrics under each collections represents the performance of the model trained on the opposing two collections. † represents significance against non adversarial model ($p < 0.05$, Wilcoxon test)

from a 80-10-10 split. Telescoping is used to create answer pools for evaluation from the top 10 BM25 retrieved answers as in [21].

InsuranceQA In the InsuranceQA dataset, questions are created from real user submissions and the high quality answers come from insurance professionals. The dataset consists of 12,887 QA pairs for training, 1,000 pairs for validation, and two tests sets containing 1,800 pairs. For testing, each of the 1,800 QA pairs is evaluated with 499 randomly sampled candidate answers.

WebAP As both L4 and InsuranceQA are based on isolated passage retrieval for a directed question, we include the WebAP collection from Keikha et al. [55] to examine how well a model trained on isolated passages with specific questions can generalize to a more general passage retrieval task. The format of this collection consists of 82 TREC queries with a total of 8,027 answer passages in total. As only relevant answer passages are annotated in this collection, we create non-relevant documents by using a sliding window of random size. Evaluation is done over a telescoped list of top 100 BM25 retrieved documents.

4.1.2.2 Training

We experimented with two different training settings—updating the ranking model and the discriminator parameters alternately as proposed by [32], and simultaneously. We also tried different values for λ. Based on our validation results, we choose to train the CosSim model with alternate updates and $\lambda = 1$. For the Duet-distributed model, we see best performance with simultaneous updates and $\lambda = 0.25$. All models were trained with PyTorch [1] and we implement early stopping based on the validation set.

4.1.2.3 Evaluation

We evaluate our proposed adversarial approach to cross subdomain regularization under two settings. Under the *cross topic* setup, we consider the 25 topics in the L4 dataset. We evaluate separately on four of these topics—Sports, Home, Politics, and Travel—each time training the corresponding models on the remaining 24 topics. For the *cross collection* setup, we consider all three collections introduced in Section 4.1.2.1. Similar to the cross topic setting, we evaluate our models on each collection individually while training on the remaining two. However, due to more pronounced differences in both size and distributions between these collections—as compared to the differences between the L4 topics—our basic adversarial approach had limited success on the cross collection task. Thus, we adopt two additional changes to our training regime:

(i) we sample the training data from the training collections equally to avoid overfitting to any single collection, and

(ii) we feed training samples from the evaluation collection to the adversarial discriminator.

We make sure that the training samples from the evaluation collection have no overlap with the test samples. In addition, we clarify that the ranking model receives no parameter updates from these training samples with respect to relevance judgments. These samples are only used to train the discriminator model's loss. This training

[1] https://github.com/pytorch/pytorch

setup may be appropriate when we want to train on some collections and evaluate on a different collection, where we can leverage the unlabeled documents from the target collection to at least guide the training of the adversarial component.

4.1.3 Results and Discussion

Cross Topic Table 4.1 show the poor performance of the CosSim and Duet-distributed models on the four target topics when trained on the remaining collection. Notably, training on the topic specific data alone also performs poorly likely because of inadequate training data. However, in the presence of the adversarial discriminator both the models show significant improvement in performance on all held out topics. The improvements are somewhat bigger on the Duet-distributed baseline. We posit this is because the Duet-distributed model—with a deeper architecture—fits the training subdomain better at the cost of further loss in performance on the held out subdomains. Therefore, the adversarial learning has a stronger regularization opportunity on the Duet-distributed model.

Cross Collection In a similar vein as the cross topic evaluation, the incorporation of the adversarial signal significantly increases performance on the held out collections in Table 4.2. However, the difference in both size and distributional properties between these collections are far greater. Therefore, while the addition of the adversarial discriminator results in significant improvements—the absolute performance on the held out collections are still modest, even with adversarial regularization. We interpret these results as a reminder of the challenges in adapting these models to unseen subdomains.

4.1.4 Negative Results

As observed in subsequent work [2], the gradient reversal process can significantly disrupt the training process as it is actively ablating useful transformations to minimize $\mathcal{L}_{rel}$. This ablation then has the possibility to shift key parameters that upper layers rely on, destabilizing the model. We observe that the negative gradient learning rate is very sensitive, potentially causing the model to diverge even when used with learning rate values that are amenable to standard training regimes. Furthermore, the reversal process cannot occur on every step, nor too soon in the training, as the forward relevance model needs to achieve a stable representation that is robust to the noise introduced via the adversarial regularization approach.

Furthermore, the adversarial component is sensitive to the input location to the original IR architecture. While the LSTM based model demonstrated robustness across layers, the Duet architecture required multiple gradient reversal inputs to achieve the desired effect.

4.2 Learning a Negative Sampling Policy for Full Collection Coverage

We now approach fully covering the collection from a sampling perspective as opposed to regularization in the previous section. For most work, heuristic sampling approaches, or policies, are created based off of domain experts, such as choosing samples with high BM25 scores or a random process over candidate documents to handle the class imbalance problem to avoid overfitting. However, these sampling approaches are done with the test distribution in mind. In this work, we demonstrate that the method chosen to sample negative documents during training plays a critical role in both the stability of training, as well as overall performance. Furthermore, we establish that using reinforcement learning to optimize a policy over a set of sampling functions can significantly reduce performance variance over standard training practices with respect to IR metrics and is robust to initial parameter values.

We decompose the training of a neural IR model into two components: an environment, which optimizes the IR model, and an agent, which learns to control the optimization process by selecting documents for the IR model to rank. Within this paradigm, past approaches of choosing a negative document randomly or from a retrieved BM25 list are represented as handcrafted sampling policies.

4.2.1 Markov Decision Process

We demonstrate that the negative sampling problem can be formalized as a Markov Decision Process [115] via the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, P, R, d_0, \gamma)$. Here, $\mathcal{S}$ represents the set of possible states the agent can be in, $\mathcal{A}$ is the set of possible actions the agent can select, and $\mathcal{R} \subset (r_{min}, r_{max})$ such that $r_{min} > -\infty, r_{max} < \infty$ is the set of possible rewards that the agent can receive. P is the transition function, $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$, such that $P(s, a, s') \coloneqq \Pr(S_{t+1} = s'|S_t = s, A_t = a)$ is the transition function that characterizes the distribution over states at time $t+1$ given the state S_t and action A_t at time t. R then represents the reward function that characterizes the distribution $R(s, a, s', r) \coloneqq \Pr(R_t = r|S_t = s, A_t = a, S_{t+1} = s')$ such that it maps the agent's action between states s, s' to a real value. Lastly, $d_0 \coloneqq \Pr(S_0 = s)$ represents the initial state distribution, and $\gamma \in [0, 1]$ is the reward discount parameter.

We call the method an agent uses to select an action a policy. A policy, $\pi \colon \mathcal{S} \times \mathcal{A} \times \mathbb{R}^n \rightarrow [0, 1]$, is a function parameterized by a weight vector, $\theta \in \mathbb{R}^n$ where $\pi(s, a, \theta) \coloneqq \Pr(A_t = a|S_t = s, \theta)$. An agent's goal is to approximate a policy that maximizes the expected sum of discounted rewards. This goal is denoted with the objective function, $J(\theta) \coloneqq \mathbf{E}[G|\theta]$, where $G = \sum_{t=0}^{\infty} \gamma^t R_t$ is called the return and conditioning on θ means actions will be selected according to the policy π using the weights θ. We assume at some finite number of time-steps, T, the agent enters a special state called a terminal absorbing state where all the actions transitions back into this state with probability one and all rewards are zero. The interval of time $t \in [0, T]$ is called an episode and when $t = T$ the episode ends and time is reset to $t = 0$. In the environment used in this paper, an episode represents the training of a neural IR model over multiple epochs until an early stopping condition is met.

4.2.1.1 Action

$\mathcal{A}$ is a set of retrieval functions, $f : Q \times C \rightarrow C_r$ over the retrieval collection C given queries Q and produces a ranked set C_r. Thus, the agent selects an action within the functional space of the document collection rather than choosing individual documents to sample. In this paper, we restrict this space to two functions, BM25 and a random distribution $d \sim U(C)$. Once the action is selected, an independent process then samples from the set of documents retrieved from this function.

4.2.1.2 State

$\mathcal{S}$ is a combination of information regarding the IR-Model and the Training Data as shown in Figure 4.4. Specifically, $s \in \mathcal{S}$ contains two parts: (1) information about the incoming batch with respect to queries and positive documents. (2) information regarding the state of the IR model and training process.

We represent the state set $\mathcal{S}$ as a combination of the current batch and the features of the neural model. The neural retrieval model is represented as the vector

$$< \mathcal{L}(b_{t-1}, \eta_{t-1}), \beta||\nabla\mathcal{L}(b_{t-1}, \eta_{t-1})||_2, \frac{t}{T_e}, e >$$

where $\mathcal{L}(b_{t-1}, \eta_{t-1})$ is the loss of the network from the previous batch given the network's parameters η_{t-1}, $||\nabla\mathcal{L}(b_{t-1}, \eta_{t-1})||_2$ is the ℓ_2 norm of the gradient in the top n layers of the neural model multiplied by a constant size β. The β parameter is introduced as a scaling factor due to the use of a deep reinforcement learning (RL) agent to bound feature ranges [116]. While neural networks perform best with normalized inputs, this plays a critical role for RL where significant changes in state can result in the distribution over actions collapsing to a single point. As the magnitude of the gradient grows with the number of parameters, β acts as a normalization constant to prevent the agent from collapsing. Lastly, the current step in the epoch, t is normalized with the total number of steps T_e in each epoch, and the current epoch number, e, is included.

As seen in Figure 4.4, in addition to the above features, the agent also receives a compressed representation of the incoming minibatch containing information regarding the query and relevant document pairs as well as similarity statistics over candidate documents. The compressed representation is done at the query-passage level. For a given sequence $w_1, \ldots, w_n$, we introduce the compression function that represents a weighted term frequency based embedding. Formally, this function, ϕ, leverages an embedding function f_e and term frequency information:

$$\phi(w_1^n) = \sum_i^n f_e(w_i)$$

Each query is represented alongside a relevant document in the matrix $\mathbf{B} \in \mathbb{R}^{2|b| \times d}$ such that

$$\mathbf{B} = [\phi(Q_1), \phi(D_{1_r}), \ldots, \phi(Q_{|b|}), \phi(D_{|b|_r})]^\intercal$$

where $|b|$ is the number of queries in the minibatch and d_e is the embedding size of ϕ. The advantage of structuring the batch in this representation allows for a series of wide convolutions to be pulled to learn a distributed representation of the state with respect to the query and positive examples. A feedforward approach was experimented with, but failed to yield performance better than random.

4.2.1.3 Reward

As discussed by Ng et al. [88], seemingly intuitive reward functions result in drastically different behaviour than what was expected. While reward shaping is still an active field of research, the authors suggest only rewarding for the event you want the agent to learn to maximize. We want the agent to learn to maximize the IR

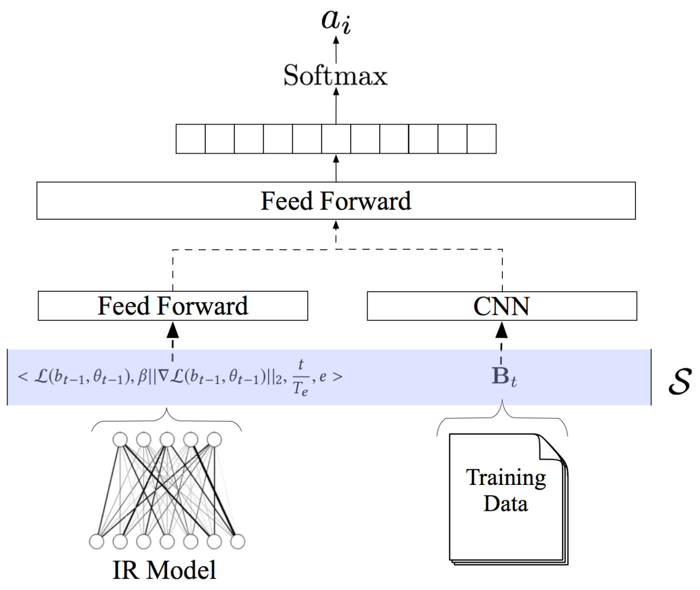

a_i
Softmax
Feed Forward
Feed Forward
CNN
$< \mathcal{L}(b_{t-1}, \theta_{t-1}), \beta||\nabla\mathcal{L}(b_{t-1}, \theta_{t-1})||_2, \frac{t}{T_e}, e >$
$\mathbf{B}_t$
$\mathcal{S}$
IR Model
Training Data

Figure 4.4: Break down of state information and agent architecture.

model's performance on the validation set during training. To this end, we consider the following three reward functions: $\mathcal{R}_{\text{raw}}, \mathcal{R}_{\text{ceil}}$ and $\mathcal{R}_{\text{diff}}$. The function R_{raw} rewards the agent using the mean average precision (MAP) score on held out query set, i.e., $R_t = MAP(q, \eta_t)$, where MAP is a function that computes the MAP score on a set of queries q, using an IR model with weights η_t. For this reward function the agent needs to maximize the sum of MAP scores the IR models can achieve. The function R_{ceil} rewards the agent using the difference of the maximum possible MAP score on the collection, $m_{\max}$, and the current MAP score, i.e., $R_t = MAP(q, \eta_t) - m_{\max}$. With this reward function the agent tries to maximize the sum of rewards, which is non-positive. The hope is the agent will learn to minimize the time it takes to converge the IR model. The last reward function we define, R_{diff}, is computed from the change in the IR model's performance, i.e., $R_t = MAP(q, \eta_{t+1}) - MAP(q, \eta_t)$. With this reward function the agent has to learn to maximize the change in the IR model's performance, $MAP(q, \eta_T) - MAP(q, \eta_0)$. Consider the following,

$$R_{t+1} + R_t = MAP(q, \eta_{t+2}) - MAP(q, \eta_{t+1}) \tag{4.4}$$

$$+ MAP(q, \eta_{t+1}) - MAP(q, \eta_t) \tag{4.5}$$

$$= MAP(q, \eta_{t+2}) - MAP(q, \eta_t), \tag{4.6}$$

for $t \in [0, T-1]$. Therefore, when $\gamma = 1$ the agent maximizes the return,

$$\sum_{t=0}^{T-1} \gamma^t R_t = \sum_{t=0}^{T-1} \gamma^t (MAP(q, \eta_{t+1}) - MAP(q, \eta_t) \tag{4.7}$$

$$= \sum_{t=0}^{T-3} \gamma^t (MAP(q, \eta_{t+1})) - MAP(q, \eta_t) \tag{4.8}$$

$$+ MAP(q, \eta_T) - MAP(q, \eta_{T-2}) \tag{4.9}$$

$$= MAP(q, \eta_T) - MAP(q, \eta_0). \tag{4.10}$$

This property can be made to hold true for all $\gamma \in [0, 1]$ if the reward function is modified to be $R_t = \gamma MAP(q, \eta_{t+1}) - MAP(q, \eta_t)$. This reward function is a potential based reward function similar to that used in reward shaping [88][2]. However, when $\gamma < 1$ the agent is tasked with maximizing $\gamma^T MAP(q, \eta_T) - MAP(q, \eta_0)$, which does not reflect our intended objective. In our evaluation of agent performance using R_{diff} we set $\gamma = 1$.

4.2.1.4 Gamma

We select $\gamma = 0.995$ for L4 and $\gamma = 0.999$ for Robust04. While $\gamma = 1$ follows from the reward shaping formulation, its role in the MDP heavily influences the difficulty of training the agent. The selection of γ reflects a balance between ease of learning a policy with a smaller γ and the performance of the policy using a larger γ.

[2]To make R_{diff} a potential based reward that preserves optimal when added to another reward function the following reward function can be used: $\gamma\varphi(\eta_{t+1}) - \varphi(\eta_t)$, where $\varphi(\eta_k) = \gamma MAP(q, \eta_k) - MAP(q, \eta_0)$

4.2.2 Agent

In this section, we discuss the agent and the approach used to learn a policy.

Algorithm 1: Approach for learning a policy based control method.

```
Input: Episode limit $L$, IR Model $f_{IR}$, Training Data, $D_{tr}$, and Validation
  Data ,$D_{tr}^{val}$, Reward function $R$, Stop Condition $stop$, and initial policy $\pi$
for episode $l = 1$ to $L$ do
    t = 0
    initialize state $s_0$ from $f_{IR}, D_{tr_t}$
    while not $stop(f_{IR}, D_{tr}^{val})$ do
        $a_t \sim \pi(s_t)$
        Take action $a_t$ and generate negative set $D^-$
        Update $f_{IR}$ with $\{D_{tr_t}, D^-\}$
        Observe $s' = s_{t+1}, r \leftarrow R(f_{IR}, D_{tr}^{val})$
        Update Agent via Eq. 4.12,4.13,4.14
        $t \leftarrow t + 1$
```

4.2.2.1 Agent Architecture

Given the distributed representation of $\mathcal{S}$, neural models are prime candidates for this approach due to their ability to transform the input space into a linearly separable decision boundary. However, the depth of the network is a critical component as non linear function approximations significantly contribute to divergence when learning a policy. Thus, we choose a shallow network to reduce stability issues shown in Figure 4.4.

As each state is generated from a standard minibatch with only relative locations playing an important role, we adopt a convolutional perspective. This approach suits the $\mathcal{S}$ distribution as there are two stages of processing. First, a small two dimensional convolutional kernel is used to match neighboring $\phi(q_i), \phi(d_{i_r})$ embeddings in $\mathbf{B}_i$. This is convolved with a large $\frac{|b|}{2}$ filter size with zero padding to learn a general batch difficulty representation. This representation is then max-pooled and passed to three affine transformations of dimensions [500,—$\mathcal{A}|, |\mathcal{A}|$]. As discussed in the following section, a softmax is then taken and sampled to determine the action a_i. This model architecture is used for both the actor and critic with the exception of the final layer, which maps to a scalar value of the state for the critic.

4.2.2.2 Policy Gradient

Many RL methods exist to optimize the agent's policy, π. In this work, we focus on the family of algorithms known as policy gradient methods. That is, we employ algorithms which approximately optimize $J(\theta)$ via gradient ascent with respect to policy parameters θ, *i.e.*, $\nabla J(\theta)$. An expression for this gradient is given by the policy gradient theorem [116], which, states that

$$\nabla J(\theta) = \sum_{s \in \mathcal{S}} d^{\pi}(s) \sum_{a \in \mathcal{A}} (\pi(s,a,\theta)(q^{\pi}(s,a) - b(s)) \frac{\partial \ln \pi(s,a,\theta)}{\partial \theta}, \tag{4.11}$$

where $d^{\pi}(s) := \sum_{t=0}^{\infty} \gamma^t \Pr(S_t = s)$ and $b(s)$ is a state dependent baseline, i.e, an estimate of the value function $v^{\pi}(s)$. In practice, one can use the REINFORCE algorithm [137] to estimate the gradient using Monte-Carlo simulation, replacing $q^{\pi}(s,a)$ with the observed return from state s and taking action a. However, this method has high variance and updates after an episode which makes it a poor fit for this setting.

The Actor-Critic algorithm [115] is a policy gradient method that performs online updates and estimates $q^{\pi}(s,a)$ with a lower variance, but biased approximation. An actor-critic algorithm is composed of two parts — an actor (the policy) and a critic that evaluates the quality of the actor's choices. The critic component in this work uses a neural network to approximate the state value function and is trained with temporal difference learning (TD). The TD-error, δ_t, is the difference of predicted value of the state S_t and the prediction after observing the reward R_t and next state S_{t+1}, i.e., $\delta_t = R_t + \gamma f_v(S_{t+1}) - f_v(S_t)$, where f_v is a function that estimate $v^{\pi}(s)$

with weights $v \in \mathbb{R}^n$. This prediction error is used to update both the policy, π, and function approximator f_v. The following updates define the actor critic algorithm:

$$\delta_t = R_t + \gamma f_v(S_{t+1}) - f_v(S_t) \tag{4.12}$$

$$v = v + \alpha_v \delta_t \frac{\partial}{\partial v} f_v(S_t) \tag{4.13}$$

$$\theta = \theta + \alpha_\theta \delta_t \frac{\partial}{\partial \theta} \ln \pi(S_t, A_t, \theta) \tag{4.14}$$

where α_v and α_θ are positive scalar learning rates.

In addition to the actor-critic algorithm we also experiment with the Proximal Policy Optimization (PPO) [106], an off-policy actor critic algorithm. Off-policy algorithms optimize the policy π using data collected from some other policy π_{old}. These algorithms introduce an additional optimization challenge as the distribution of data collected using π_{old} does not match the data distribution of π. This distribution mismatch can be corrected using importance sampling [99], but increases the variance of the gradient estimates. In our experiments below, actor-critic outperforms PPO.

4.2.3 Experiments

In this section, we describe the baseline approaches, data used, and IR models evaluated. Succinctly, we train and evaluate the learned policy on two different IR models over two different collections. Once the agent has converged as evidenced by performance of the IR model on the validation set, we evaluate the IR model's performance.

4.2.4 Collections

We use two diverse collections, each representing a different negative sampling choice. The first collection, Yahoo's Webscope L4 [3] represents an answer passage

[3]https://webscope.sandbox.yahoo.com

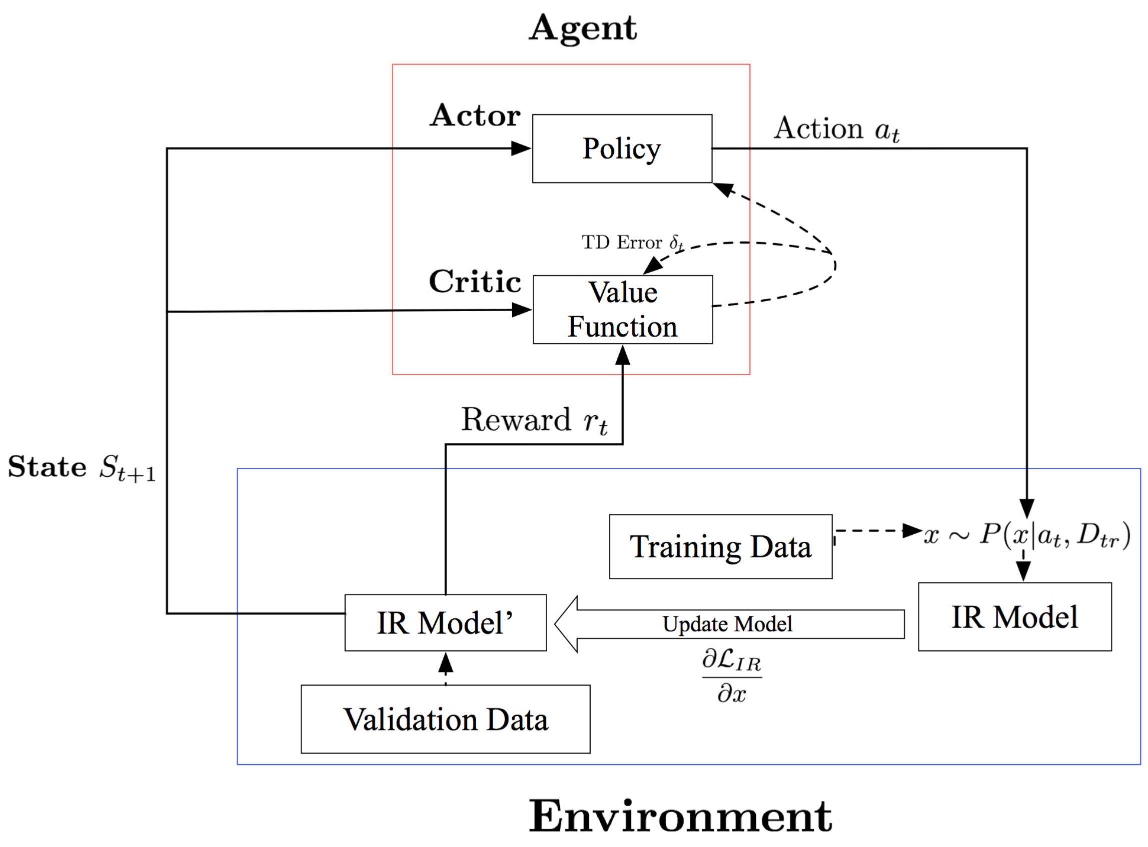


Figure 4.5: Actor-Critic Setup with IR Environment

retrieval problem, where there is only one relevant answer in the entire collection. These questions are filtered from Yahoo Answers that meet the criteria of manner questions, as discussed in [129] and has approximately 120,000 query-answer pairs. This collection represents the situation when BM25 is not a strong baseline for relevance [129, 21] and includes the task of identifying effective negative passages for training as there is only one positive passage with all others acting as negatives.

The second collection, Robust04, consists of 500k news documents from TREC disks 4 and 5, and the query set consists of 250 title queries (TREC topics 301-450, 601-600). Here, each query has judged negative documents along with multiple relevant documents. Furthermore, these judged documents were selected from pooled retrieval runs that are heavily influenced by term frequency information. Thus, this

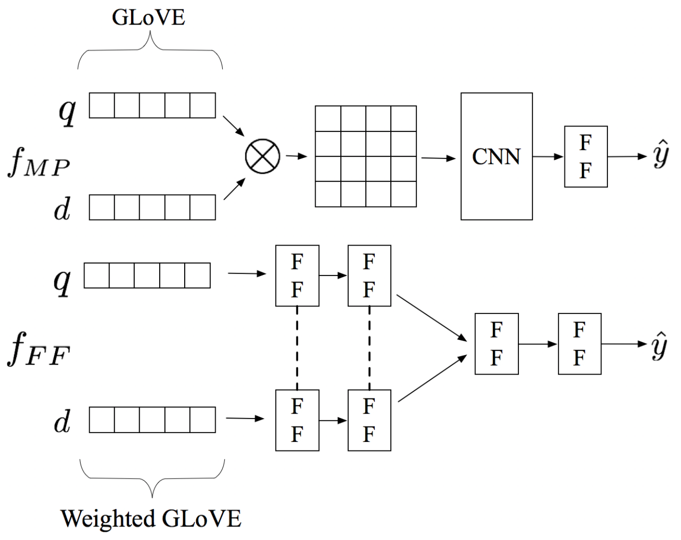


Figure 4.6: Compressed architecture of two neural models used. FF represents a feedforward layer, f_{MP} is MatchPyramid while f_{FF} is a siamese network with dotted lines representing shared weights. GloVE weighting layer is excluded in diagram for f_{FF}

represent the case where labeled information is significantly richer, but there exists an underlying bias as well as much smaller query set for training.

4.2.4.1 Baselines

We examine naive sampling over all functions in $\mathcal{A}$. Once each function retrieves a list of candidate negative documents, three approaches are examined. First, a naive random sampling approach is used over the list. Second, we adapt an uncertainty sampling based approach, expected error reduction (EER), to select the best negative document to train on [103]. As EER is performed for active learning where $P(y|D) >> 0$ for some class y, this is not true in search, where the majority of a collection is not relevant to a given query. Thus, given a random document, we assume its correct label is non relevant and arrive at the below representation of EER adapted to a known class:

$$d_{neg} = \underset{d \in C \setminus R_q}{\arg\max} P_\eta(r|q,d) \log P_\eta(r|q,d)$$

where $C \setminus R_q$ represents the set of documents not labeled relevant for query q, and $P_\eta(r|q, d)$ represents the probability given parameters η that document d is relevant to query q. As the third sampling baseline, we implement an approach leveraging a distribution based view of information gain [157] referred to as Dynamic-λ.

In addition, as the policy will have access to a larger sampling space than some of the baselines, we include a random agent to ensure that the policy has learned something besides a random sampling approach. This is a relatively high baseline given the curated $\mathcal{A}$ function space as well as a random policy acting as a competitive baseline in [37].

Lastly, we include the performance of IRGAN [130] on only the discriminator. While the authors state that IRGAN consists of two types of retrieval model and often one outperforms the other, it can be viewed as learning the most difficult sampling policy via REINFORCE for the discriminator. Thus, we ignore the generator to properly compare frameworks. We adopt code provided by the authors and tune for performance on each model and collection.

4.2.4.2 Neural Retrieval Models

We evaluate the efficacy of sampling methods with two deep neural methods that provide a challenging control problem for the policy due to the large numbers of parameters. We introduce two neural models of varying complexity as seen in Figure 4.6.

First, we introduce a feedforward model, referred to as f_{FF} that is typical of a distributed neural retrieval model [84]. Using a GloVE initiated embedding, f_{FF} treats the document as a weighted bag of words similar to [24] with a learned weight for each term. This is then averaged into a vector representing each query and candidate document. The lower layers, referred to as f_l, are of dimension [2048,1024,512] and process the query and document independently prior which are then concatenated to form a query-document vector $< f_l(Q), f_l(D) >$. This is then passed into an upper feedforward network of dimension [512,300,1]. f_{FF} is trained by maximizing the log likelihood of the correct document over the sampled set, C_s, via taking the softmax.

As f_{FF} leverages an independent query-document representation, we implement MatchPyramid (MP) using a cosine similarity function over input embeddings to demonstrate an interaction based model under policy control [94]. The model is trained as in the original work by using a cross entropy loss function to learn parameter weights and is referred to as f_{MP}.

4.2.4.3 Evaluation

We evaluate the policy by examining performance on the held out validation set during each episode. We examine MAP of the top ranked 100 documents for Webscope L4 and the top 1000 for Robust04. The smaller scope of L4 was selected due to the computation costs as there are roughly 12,000 queries in the test set. Significance between methods are determined via two tailed t-test. However, as we are examining not only total performance, but consistency across different random seeds and hyperparameters, we include the Kolmogorov-Smirnov test that measures the probability of two empirical distribution functions belonging to the same distribution. Both measurements are evaluated with a significance value of $p < 0.05$.

4.2.5 Results

In this section, we evaluate the method, AC-IR, over two different retrieval tasks and neural models. We report distribution information by including mean and standard deviation. As discussed in previous work [11, 53] and bolstered by the lottery ticket hypothesis [29], examining the max run from multiple experiments leads to an ineffective evaluation of a stochastic process. After discussing indicators of performance, we provide analysis into the impact of the reward shaping approach, hyper-parameter stability, and lastly, the convergence and stability of the agent over multiple runs and during training.

Method	Webscope L4	
	f_{FF}	f_{MP}
BM25$_{rand}$	0.0706±.029	0.1631±.064
BM25$_{Dynamic-\lambda}$	0.0905±.032	**0.2083±.040**
BM25$_{EER}$	0.0919±.031	0.2050±.039
Random$_{rand}$	0.0679±.004	0.0727±.011
Random$_{Dynamic-\lambda}$	0.0621±.073	0.0915±.005
Random$_{EER}$	0.0642±.065	0.0899±.005
IRGAN$_{policy}$	0.0557±.038	0.0807±.005
AC-IR	**0.1239±.011***†	0.1975±.008†
Random AC-IR	0.0603±.003	0.1026±.040
	Robust04	
	f_{FF}	f_{MP}
BM25$_{rand}$	0.0320±.012	0.056±.015
BM25$_{Dynamic-\lambda}$	0.0408±.014	0.0549±.012
BM25$_{EER}$	0.0409±.012	0.0558±.011
Random$_{rand}$	0.0390±.001	0.0518±.015
Random$_{Dynamic-\lambda}$	0.0401±.002	0.0597±.022
Random$_{EER}$	0.0386±.003	0.0600±.021
IRGAN$_{policy}$	0.0394±.006	0.0538±.019
AC-IR	**0.0496±007***†	**0.107±.046**
Random AC-IR	0.0455±.003	0.102±0.048

Table 4.3: Performance of sampling methods with respect to mean average precision. Mean performance is included with standard deviation. *,† refer to significance to $p < 0.05$ compared to highest baseline using Student's t-test and the Kolmogorov-Smirnov test respectively.

4.2.6 IR Impact

Looking at the runs in Table 4.3, we observe that AC-IR significantly improves the consistency of performance on Webscope L4 for both f_{FF} and f_{MP} over multiple random seeds. In the case of f_{FF}, AC-IR is able to outperform that of the EER and Dynamic-λ approaches on both collections without explicit access to the model's uncertainty on a new batch. The agent is able to capture this internally using only $\mathcal{S}$ and the reward to infer this information. As an example, we plot performance over many random seeds in Figure 4.7 on L4, and only AC-IR is capable of consistently achieving the upper bound of performance when compared to other methods. Furthermore, we identify a bimodal distribution on the BM25 baselines, where the model either successfully converges to values near the max for the given mode or fails to learn based off of the initial parameter distribution.

For f_{MP}, AC-IR performs slightly worse than Dynamic-λ on L4 with respect to both mean and distribution characteristics. However, on the case of Robust04, the policy significantly outperforms all baselines, and reaches parity with the random agent. This behaviour is learned, as non-linear RL has a tendency to collapse to a single action [116], and AC-IR is significantly different than the random policy on all other collections. Furthermore, the BM25 methods have a greater increase in reward during initial minibatches, and without an effective policy to converge on, AC-IR would most likely collapse to BM25.

The result of AC-IR not drastically improving the max reported score is particularly interesting, as unlike standard supervised training collections like CIFAR [59], the information space over IR collections is significantly larger with respect to Shannon entropy. This suggests that the neural models are possibly limited by the number of linear regions the parameters can operate over as discussed by Montufar et al. [86]. Thus, viewing the functions in $\mathcal{A}$ as a set of linear regions, the neural IR models are exposed to a well defined but narrow area of the manifold via BM25, and a much larger area via the random process with the possibility that the gradient descent update might not be informative due to multiple linear regions within a minibatch. Therefore the upper bound of each neural model is not significantly improved by AC-IR. However, controlling the type of regions exposed to the model during training significantly improves mean performance, as well as the number of runs that fall near the upper bound of performance.

Lastly, the relatively low performance of IRGAN can be attributed to three issues. First, REINFORCE is high variance given the static state value $b(s)$ in Equation 9 and the fact that the reward can suffer large changes in certain states, such as if the IR neural model begins overfitting on the training set. This is further exacerbated by the depth of the neural retrieval models being used in this experiment. Second, the authors state that the generator applies a hierarchical softmax, but this is a non trivial structuring of the sample space [33]. Third, we do not use the generator as a ranking model as it represents the sampling policy to train the discriminator.

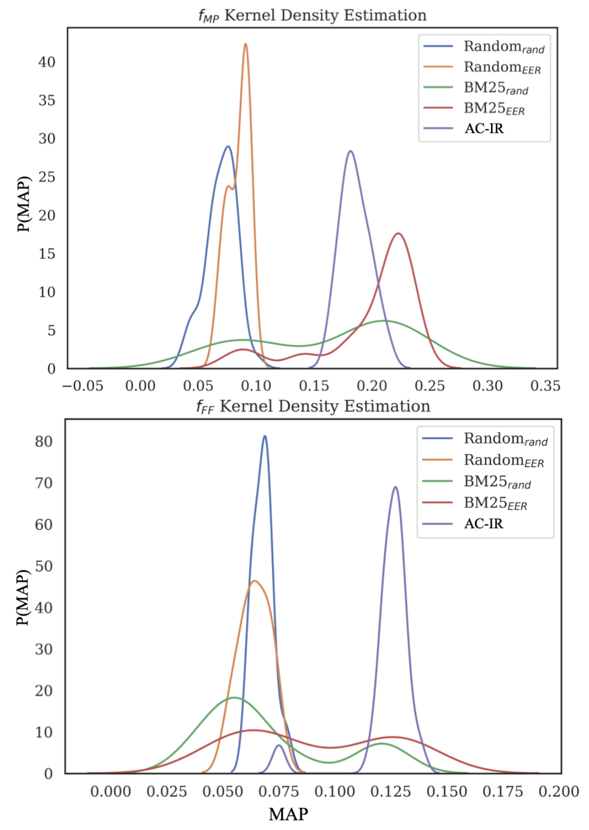


Figure 4.7: Distribution of policy performance using kernel density estimation over Webscope L4. AC-IR demonstrates performance during convergence.

Succinctly, the learned functional policy of AC-IR is able to take advantage of the strong performance of the static policies while ensuring the poor performance regions of their distributions are not reached.

4.2.6.1 Document Level Actions

While the AC-IR acts over sampling functions, we investigate the capability of the AC agent to learn a policy to select individual documents. We convert all documents in the collection to a *tf.idf* weighted mean embedding, and for each Q, D in a batch, we create a candidate list of size s_d from the *tf.idf* weighted embedding of Q via cosine similarity. We use a new action space, $\mathcal{A}_{\mathcal{D}} = \mathbb{N}_{<s_d}$, where the i^{th} action represents selecting the i^{th} closest document in cosine space. We modify $\mathcal{S}$ to include additional information about the candidate list for each query by a matrix $\mathbf{M} in \mathbb{R}^{s_d \times 5}$, where each tf, document length, unique terms, cosine similarity, and BM25 ranking. As over 91% of queries have a top 100 ranked BM25 document within the top s_d documents in cosine space, the agent should be able to at least collapse to a BM25 sampling policy which we empirically determined to be a more effective policy to cosine similarity. However, the agent fails to converge on this new MDP despite extensive hyperparameter tuning. We investigate this behaviour further by incorporating imitation learning to identify what kind of signal is required to learn a BM25 sampling policy in cosine space. Following work by [13], we pretrain the AC agent using a supervised signal rather than directly with a reward function. In our case, the signal consists of a binary label for each document indicating whether the document is also within the top n retrieved BM25 ranked documents for the query. In theory, the new state space includes enough information to determine rough BM25 rankings indicated by past work in weak supervision [24]. However, even with the benefit imitation learning the agent still fails to perform better than random.

Examining the generalization properties of the model trained via imitation learning, we observe that the agent is only capable of memorizing the rankings BM25 documents until its parameters are saturated. This suggests that for the case of document selection in an MDP, actions, and thus the corresponding updates to the policy, should be ranked rather than treated in isolation as potential future work.

4.2.6.2 Agent Training

Convergence and Stability: Previous work has discussed the instability of both RL and neural networks using the same hyperparameters over training data [53, 116]. In our case, we observe instability during training of both the neural IR model and the agent. As shown in Figure 4.7, simply changing the random seed significantly impacts the performance of a neural model on the majority of sampling methods. This presents a challenging problem for the agent as it must not only identify how well the neural model is converging based on the agent's knowledge from past episodes, but determine the position of this neural model within its performance distribution. The way we defined the state space cannot fully capture the underlying mechanisms of the neural IR models. That is the state of the IR model is only visible to the agent and can be modeled as a partially observable Markov decision process, using the observation function $O : \mathcal{S} \times \mathcal{O} \to \mathbb{R}_{\geq 0}$, describes the distributions over observations (features), ψ, given a state S. The state as we have formulated it are only observations and the actual state remains unknown to the agent. Currently, function approximation is used to overcome the partial observability and learn a sampling strategy without knowing exactly on which queries and documents the IR model can rank correctly. If instead a set of features that provide this missing information were used, the agent could then learn a policy to that adapts the sampling strategy to explicitly exploit the current state of the IR model.

4.2.6.3 Hyper-parameter Case Study

Reward Shaping: We report AC-IR under $\mathcal{R}_{\text{raw}}, \mathcal{R}_{\text{ceil}}, \mathcal{R}_{\text{diff}}$ over the L4 dataset shown in Table 4.4. The performance of the policy with respect to the IR task of training an effective model is clearly shown to be captured most by $\mathcal{R}_{\text{diff}}$, while $\mathcal{R}_{ceil}$ is able to achieve an effective policy for this task albeit in less than 12% of all runs for only a small number of episodes. However, the optimal policy for $\mathcal{R}_{\text{raw}}$ is drastically different than what we expected and was similar to findings in [88]. In this case, as the future discounted return $\mathcal{R}_{\text{raw}}$ is monotonic with time due to the inability for MAP to be negative, the agent learns to prolong training as long as possible. Rather than maximizing the performance, it avoids triggering the early stopping condition by ensuring small but consistent gains in performance. Given a long enough training period, this policy produces a greater return than attempting to maximize the performance of the neural model at a single epoch. While modifying γ can reduce the total return and make this process easier to learn for the agent, it becomes another critical hyperparameter that is tied to collection and neural model characteristics rather than the overall control problem.

Reward Method	MAP	Convergence Rate
$\mathcal{R}_{\text{diff}}$	0.1239	31%
$\mathcal{R}_{\text{raw}}$	0.0603	0%
$\mathcal{R}_{\text{ceil}}$	0.0671	12%

Table 4.4: Performance of agent trained to optimize different reward functions. MAP on Webscope L4 with convergence rates greater than random are reported using f_{FF}.

Limiting the Amount of Epochs: As each episode is defined by training the neural model until the stopping condition is met, we experiment with setting a limit on the amount of epochs that can occur in each episode to facilitate faster training of the agent. We set the maximum number of epochs to four, and identify that this results in better performance for the agent learning from $\mathcal{R}_{\text{ceil}}$, but prevents an effective policy to be learned under $\mathcal{R}_{\text{diff}}$. This result supports the reward shaping motivation discussed in Section 4.2.1.3, as the hard limit on four epochs relieves the agent operating under $\mathcal{R}_{\text{ceil}}$ from stopping the training as early as possible and instead focuses on maximizing performance during this time.

Early Stopping: As the IR model will eventually overfit on the test data regardless of the sampling method, the reward on the held out validation set will start decreasing. This undesirable tail end behaviour can then result in non optimal updates to the policy with respect to the performance of the neural IR model prior to overfitting. Furthermore, the choice of $\gamma < 1$ biases the expected return by discounting the earlier rewards achieved when the IR model was able to generalize to data outside of the training set. We experiment by increasing the early stopping criteria to a patience of ten epochs that fail to improve over the validation set. In doing so, we observe that no agent is able to successfully converge to an effective sampling policy that's significantly different than random. This behaviour suggests that the agent is not capturing the environment fully, as the critic should be able to learn $b(s)$ as shown in Equation 4.11.

4.2.7 Negative Results

Properly capturing a neural model's training process over a large number of samples is a challenging task. As such, there were a number of hypotheses which failed to outperform the competitive random baseline. The first of which attempted to include a survey of the collection to inform the agent of its next actions and choose candidate documents individually. This approach failed to perform better than random, and highlights some of the shortcomings of the approach introduced above.

Shown in Figure 4.8, MDP_2 redefines the problem from selection retrieval functions to individual documents. This constitutes a new MDP, as we redefine $\mathcal{S}$ and $\mathcal{A}$ to include the ranked list of top n documents through BM25, cosine similarity to the

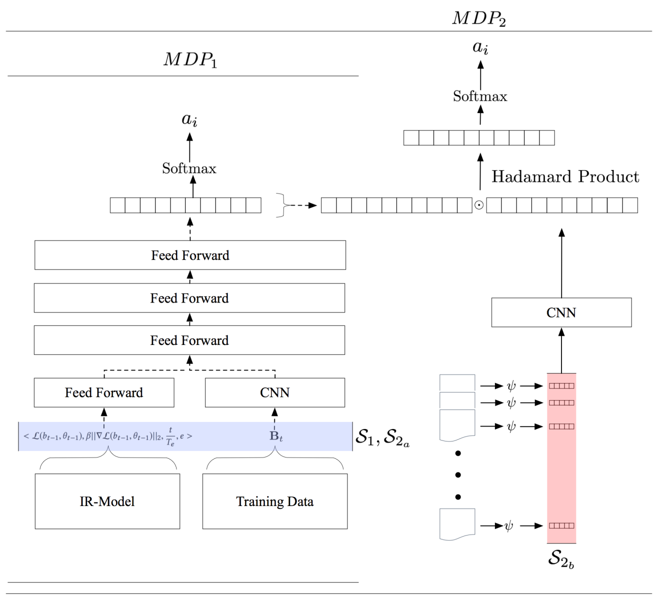


Figure 4.8: Individual document selection framework.

relevant positive document in doc2vec space [61], or random. The documents are encoded via ψ, which is similar to the weighted embedding representation ϕ used in Section 4.2.1.2, but with tf.idf weight information included. When trained, AC-IR over MDP$_2$ did not perform significantly better than random, and the agent would often fail to converge to a stable sampling policy. Recently, Tang and Yang [119] introduced a way to view large collections as a single state space for effective RL, possibly remedying the issues experienced in MDP$_2$.

CHAPTER 5
TOWARDS A UNIVERSAL RETRIEVAL MODEL

As demonstrated in Sections 4.1 and 3.2, neural retrieval models are extremely sensitive to small perturbations in data. In this chapter, we propose a novel framework to allow for an effective retrieval model over multiple collections with minimal training.

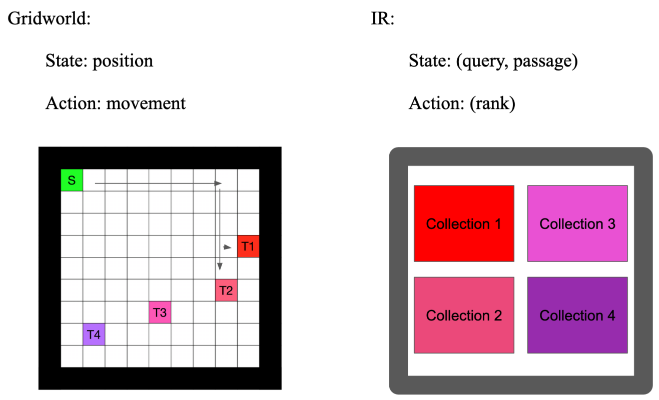


Figure 5.1: Parallel between multitask RL and URF.

We first introduce an inspection of the information content needed to determine relevance between two similar QA collections in Section 5.1. We then incorporate recent advances in RL to cover the specific training regimes often seen in IR training where a retrieval model individually scores documents which seemingly contradicts the paradigm seen in RL where the model operates over the entire state space. Lastly, we extend this work to cover multiple collections, providing the foundations of universal retrieval functions(URF) which parallels multitask goals in RL environments. Each collection's relevance need can be viewed as an individual task over the combined body of passages and documents from multiple collections. This setting is commonly seen in RL, exemplified in gridworld where the task is the goal location, and the general environment is the structure of the maze as shown in Figure 5.1. Task transfer and general value functions within RL literature attempt to reduce the cost of learning a new task given a similar or fixed environment. We extend this work to IR and move towards a universal retrieval function (URF) such that, when trained, this URF is capable of effectively performing on a variety of collections. As RL relies on a formal MDP definition, we model the retrieval process within this framework, and demonstrate that related prior works in RL naturally extend to conventional training paradigms of neural information retrieval models.

5.1 Leveraging Linguistic Structures for Retrieval

Identifying the required information to determine relevance plays an integral role in selecting more effective models and sheds light on what information is safe to discard. A motivating advantage of neural models over traditional learning to rank approaches is the ability to learn increasingly abstract features while ignoring a significant amount of detail. Leveraging how information is stratified and discarded in this process provides insight into whether a portion of a document would be useful for closer inspection of a higher quality neural IR model along with its increased computation cost. Within the field of NLP, there has been a large body of work investigating what information is useful for these tasks through attention mechanisms [72], gradient investigation [63], or cell activation [54]. However, direct application of these techniques for IR tasks results in identifying traditional features [93, 84] such as term overlap and related words we have demonstrated to be insufficient for the passage retrieval task.

IR and NLP have close ties given similar neural structure work across the domains, and with similar models finding success in both domains [158, 108, 50, 25]. In light of this advantage of NLP pre-training, we propose viewing IR neural models as transformations on traditional linguistic structures defined through auxiliary NLP tasks rather than at the traditional term level. This approach portrays relevance features as a function of distributed NLP features, and while this is not innately interpretable, the rigid definition of these auxiliary tasks provide a reference to examine how the IR model leverages these non traditional features based on an IR model's ability to transfer relevance information to these tasks.

To demonstrate the degree to which relevance is composed of linguistic information, we provide a novel technique leveraging Alain and Bengio's [6] probe based methodology and apply it to measure the information loss with respect to related tasks. As NLP objectives are often highly structured, they provide a more concrete environment to understand what information is being captured in a network trained for retrieval.

5.1.1 Method

We investigate the information captured by an IR answer passage deep neural model by utilizing the intermediate representations of deep neural networks. The foundation of this work comes from the data processing inequality [22], where given a Markov chain of successive representations, $X \rightarrow Y \rightarrow Z$, then

$$I(X;Z) \leq I(X;Y)$$

where $I(X;Y)$ is the mutual information function. Tishby and Zaslvsky [121] show that layered neural networks form a Markov chain of representations. Thus from the perspective of deep neural models, each transformation at best can preserve the information from an earlier layer and the most information available at any point in a neural model is contained in the lower layers. The advantage of additional layers

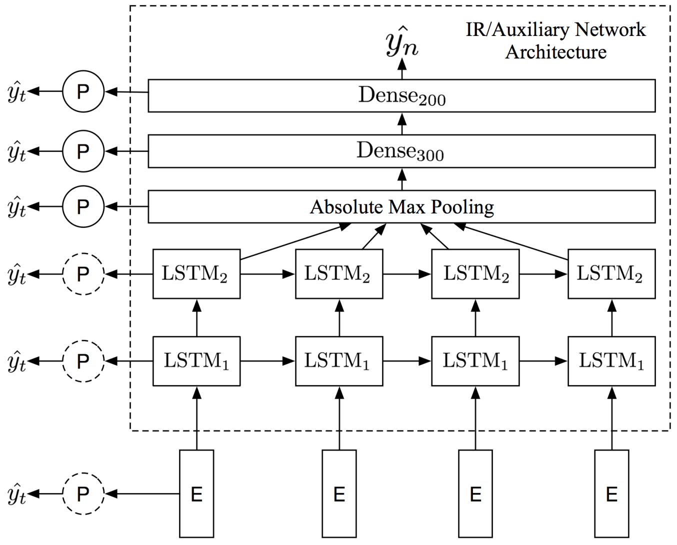


Figure 5.2: An overview of the probe (P) insertion model. The main LSTM model attempts to evaluate the input for its main task in embedding (E) format, $\hat{y}_n$ while the probes use each layer's intermediate representation to predict an auxiliary task $\hat{y}_t$.

is not to add information, but to identify the salient information with respect to the target, and transform the representation into a linearly separable space.

We apply this inequality to identify what information is discarded by a passage retrieval model from the perspective of NLP tasks. Specifically, after training an IR model for answer passage retrieval, we freeze the weights and pass it over POS, NER, sentiment, and textual entailment datasets. Each hidden state of the IR model then goes into separate neural networks, or probes, to predict the input's true NLP label. The efficacy of the probes' ability to learn and predict the various NLP tasks illustrates what linguistic and semantic properties are being discarded by the network from the base embeddings.

5.1.1.1 Model

In this paper, we implement a multilayer LSTM network similar to the one used in Chapter 2 that feeds into a series of dense feed forward layers (Figure 5.2). This model

Table 5.1: Hyperparameters of Main and Probe models with K as the final classes for a task

Main Model		
Layer	**Dimension**	**Activation**
LSTM 1	512	internal: sigmoid, output: *tanh*
LSTM 2	512	internal: sigmoid, output: *tanh*
Dense	300	ReLU
Dense	200	ReLU
Dense	K	-
Probe		
Layer	**Dimension**	**Activation**
Dense	300	ReLU
Dense	200	ReLU
Dense	K	-

has the advantage of being effective for both NLP and non-factoid passage retrieval while possessing a simple structure lending itself to easier interpretation [118, 74]. The final layer is a dense layer with the number of nodes equal to the number of classes where a softmax function is taken to create a probability distribution over the labels. Additional hyperparameters are provided in Table 5.1. We use a dropout rate of 0.2 over both LSTM outputs during training of the main network for all tasks as it has been shown to improve the generalizability of deep neural models to unseen data [151, 154].

5.1.1.2 Auxiliary Networks

In order to provide an approximate upper bound for the internal representation and information contained in the IR network described in the previous section, we train additional LSTM networks identical to the IR network dedicated to each auxiliary task. The sole difference between these auxiliary LSTM models and the IR model is the training data and dimension of the final dense layer. The same hyperparameters and training methods were used across all LSTM networks. The difference in probe performance between those inserted in the IR model and those in the auxiliary network can then be viewed as the discarded information with respect to the auxiliary signal.

5.1.1.3 Probes

We use small multilayer perceptions which accept as input each intermediate layer of the main LSTM networks. These probes are trained to predict target labels of the input using only the current hidden layer of the network they are monitoring. Changing the input of the main IR network to reflect an auxiliary task allows the probes to effectively become a measurement tool in how much information the main network is retaining with respect to these auxiliary labels. An illustrative representation of the setup is shown in Figure 5.2. Any task that requires individual labeling of tokens, denoted by the dashed probe symbol P, are fed the LSTM and embedding sequences in temporal order. Text classification tasks receive (1) a max pooled representation across time of the recurrent layers, (2) a sum of embeddings or (3) direct output from a dense layer as shown below,

$$x_i = \underset{t}{}(|\mathbf{h}_{i,t}|) \tag{5.1}$$

$$\mathbf{x} = \sum_{w \in S} \text{Embedding}(w) \tag{5.2}$$

$$\mathbf{x} = \sigma(W\mathbf{h}_{l-1} + b) \tag{5.3}$$

where $\mathbf{x}$ is the vector input into the probe, $\mathbf{h}_{i,t}$ is the hidden LSTM layer at dimension i at time t, w represents the words contained in the sample S, and $\mathbf{h}_{l-1}$ is input into the dense layer of the main network and the output is passed to the probe.

5.1.1.4 Tasks

We also evaluate the vocabulary overlap between the IR collection and the auxiliary task to ensure that the majority of the new input into the IR network has been

Table 5.2: Vocabulary overlap measured by $\frac{A_i \cap B_j}{B_j}$ between auxiliary collections (B_j) and the two IR collections (A_i).

Task	**L4+nfl6**		**WikiQA**	
	Unique	**Total**	**Unique**	**Total**
CoNLL 2003	.595	.649	.424	.565
PTB II	.769	.815	.584	.673
IMDB	.293	.972	.110	.916
SNLI	.720	.994	.383	.952

seen during training. As shown in Table 5.2, there is a significant overlap between IR training and task evaluation vocabulary with the exception of IMDB, which is most likely due to the movie and actor entities present in the collection.

Core Task: Answer Passage Retrieval: The core IR task being studied is answer passage retrieval. As mentioned, this task represents a unique challenge when compared to ad-hoc retrieval and factoid QA. While factoid retrieval often encounters questions such as "*When did James Dean Die*" or "*How high is Everest?*" that require only one or two tokens to successfully fulfill the information need of the query, passage retrieval requires information that spans multiple sentences. This integral difference results in factoid QA networks failing to beat standard *tf-idf* baselines on answer passage retrieval tasks as seen in Section 3.1.2.4.

Auxiliary Task: Part of Speech Tagging: Part of speech (POS) tagging is the task of labeling each word with its syntactic part of speech, e.g. noun, verb, adjective, based on its use in a sentence. As shown in past work by Bjerva et al. [12], networks trained on semantic tagging tasks independently capture part of speech information. As passage retrieval requires semantic processing to bridge the information across sentences, we investigate the extent to which an answer passage neural model also captures POS tags.

Auxiliary Task: Named Entity Recognition: While related to POS tagging, named entity recognition (NER) requires higher level features which often consist of POS information, whether latent or explicit, due to the dependencies across a sentence and additional information required for accurate entity tagging [60, 64]. We evaluate a NER auxiliary task to see if the core answer passage network dedicates some of its parameters to capture information pertaining to named entities. Recent work in deep neural QA [108] has shown that adding named entity overlap between question and answer significantly improves performance with respect to IR metrics. The increase in IR metrics suggests that entity information plays an integral role for modeling relevance.

Auxiliary Task: Sentiment Classification: As a significantly higher level task compared to POS tagging and NER due to the need to process and compress an entire sequence, we implement a sentiment classification task. Here, the objective is to correctly identify whether a sentence denotes a positive or negative view of the topic. Li et al.'s [63] work in visualizing LSTM networks for sentiment classification provides insight on what features are important to predict sentiment. The most critical components are the ability to capture local context around a word, recognize negation, qualitative adjectives and key verbs.

Auxiliary Task: Textual Entailment: We use a textual entailment task to evaluate whether information retrieval at the passage level could be viewed as whether the query provides evidence for a passage to be considered relevant. The goal of this task is to determine whether two sentences (1) are contradicting each other, (2) are unrelated, or (3) that the first sentence (the evidence) entails the hypothesis. The performance of the probes on the core IR model will help disentangle the semantic information related to entailment over that which relates a query to its relevant passage. As each example is an ordered pair of sentences, we concatenate the evidence-hypothesis sentences the same way as query-passage pairs for the answer passage retrieval task. The evidence serves as the query and the passage represents the hypothesis. The auxiliary network and probes for this task have a three node dense final layer to classify entailment, contradiction, and neutral classes.

5.1.1.5 Multitask Inspection

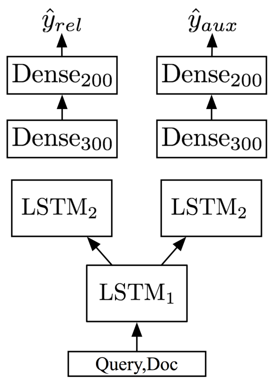


Figure 5.3: Simplified representation of multitask architecture with LSTM$_1$ acting as shared layer.

As information retained in each layer has some benefit towards determining relevance, we examine the impact of explicitly reinforcing this signal through a multitask environment using a similar neural structure as Long and Wang [71] where gradients are passed through task specific sub-networks into larger main model. Thus the probe remains task dependent while the layer of the IR network it connects to, and those below it, become shared layers for the multitask objective. This approach retains the probe inspection method while simultaneously adopting a competitive neural multitask framework. As the IR collections do not have gold NLP labels for training, we use the trained auxiliary NLP networks to create pseudo labels for training.

The structure of the multitask architecture consists of the main model hyperparameters described in Table 5.1. The corresponding task-specific substructures are mirrored. Thus if the shared layer is LSTM 1, then LSTM 2 and the subsequent feedforward hyperparameters are used for both tasks with no weight sharing. A depiction of this setup is exemplified in Figure 5.3. The multitask model is optimized via the joint loss function

$$L = L_{aux} + L_{IR} \tag{5.4}$$

where L_{aux} and L_{IR} are the respective loss functions used for single task training discussed in the following section.

5.1.1.6 Training

We use Adam for optimizing both the main models as well as the probes with a cross-entropy loss function and a learning rate of 10^{-3} , which provides a robust value for training [57]. Each main model was trained via PyTorch[1] over a 80-10-10 train, development, and test partition and was stopped after the best validation loss did not improve for four epochs as a form of early stopping. Each probe was trained, validated, and tested on the same data to measure the amount of information captured by the main model rather than the probe's ability to generalize.

Input into the IR network is done in a similar manner to past work [128] by concatenating question and passage text with an end-of-sentence (EOS) token as shown below.

$$< q_1, \ldots, q_n > + < \text{EOS} > + < a_1, \ldots, a_m >$$

This allows for query passage interaction while still being easily adaptable to processing input from auxiliary tasks. In the case of the NLP tasks that do not have text pairs to partition with <EOS>, we feed the text in directly to simulate the query stage of an IR task, and then we train another set of probes on samples where <EOS> is prepended to the same the sample. The IR network views this as

[1]https://github.com/pytorch/pytorch

an empty query and a candidate passage, which enables us to identify how captured information differs for the same text as query and passage in the IR main network.

All tokens are expressed as GLOVE 300D embeddings [2] [96]. In order to provide a consistent text representation across all tasks, we do not update the initial embeddings during training at any point. This represents a common baseline across all models.

5.1.1.7 Datasets

Answer Passage Retrieval: We combine the Yahoo's Webscope L4 and the noisier nfl6 collections and refer to it as the CQA collection for the remainder of this framework. In addition, to determine whether there is a distribution of information specific to that determining the relevance of answer passages, we include a shorter factoid retrieval collection: WikiQA [145]. The average length of the L4 and nfl6 answer passages are 92 and 60 words respectively, while WikiQA sentences have an average length of 25 words.

Part of Speech Tagging: The collection used for evaluating this auxiliary task is the Wall Street Journal set from Penn Treebank III [76]. As mentioned, POS probes are inserted only into the temporal (LSTM and embedding) layers of the core network. The POS auxiliary main network was trained over 46 POS using the standard train (0-20), validation (21, 22), and test (23, 24) splits as seen in past work [81].

Named Entity Recognition: We use the CoNLL-2003 NER for training and evaluation [105] and use *MISC, LOC, ORG, PERS, O* tags over the standard BIO annotation (*Begin, Inside, Outside*). This was done to investigate whether the IR main model is able to identify and differentiate among the classes over the more detailed task of determining whether a token is the beginning of a phrase or inside it.

Sentiment Classification: Like past work [63], we use the Internet Movie Database (IMDB) review collection [75] where a movie review is either positive or

[2]http://nlp.stanford.edu/data/glove.840B.300d.zip

negative with 25k samples for each label. We use binary cross entropy to evaluate this task.

Textual Entailment: We use Stanford's Natural Language Inference (SNLI) corpus [14]. This entailment set is a collection of 570k human-written English sentence pairs with labels of *entailment* (183,416), *contradiction* (183,187), and *neutral* (182,764). We discard the 785 samples that do not fall under one of these three labels. Each sample is an ordered pair of sentences, one that serves as the evidence and the following that is a hypothesis. The auxiliary network and probes for this task have a three node dense final layer to classify entailment, contradiction, and neutral classes.

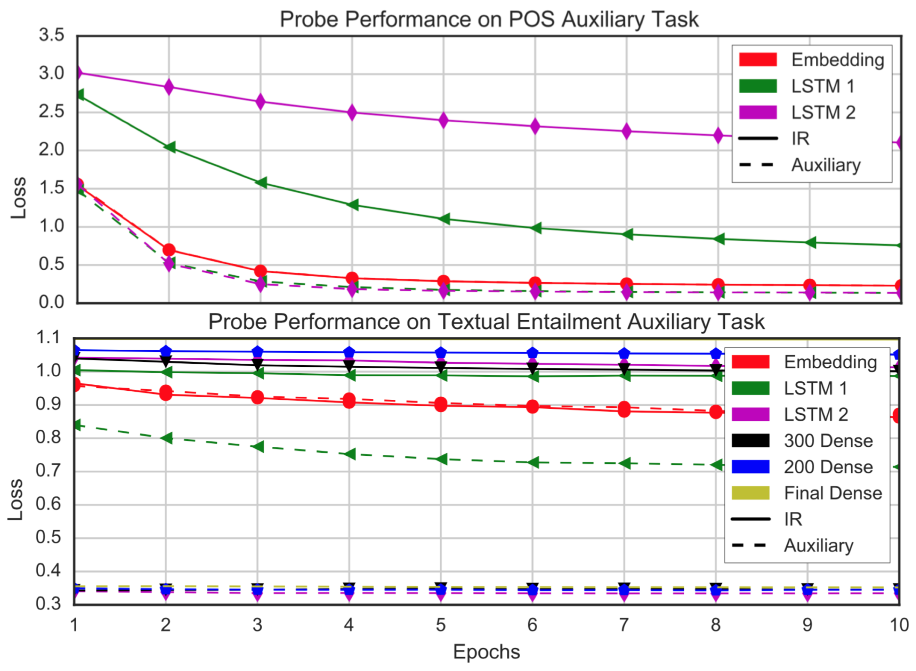


Figure 5.4: Performance of probes over each layer on all auxiliary tasks as queries. IR represents the probes inserted into the answer passage network and Auxiliary represents probes inserted into the identical network trained for the auxiliary task.

5.1.2 Results and Discussion

As shown in Table 5.3, there is a steady decline in information loss as the initial embeddings flow up through the layers. Following work found in computer vision [148] where each layer captures increasingly abstract representations, the answer passage model also reflects this tendency. Lower level POS and NER information is captured consistently in the first LSTM layer and discarded in the upper layers, while the abstract entailment information persists into the model's upper layers, even sharing some of the transformations needed to determine passage relevance. This reinforces the analysis done by Søogaard and Goldberg [111], where they had greater success with a neural architecture that supervised POS information at the lower layers for multitask learning. Lastly, we show that two seemingly similar IR tasks that are considered closely related have significantly different information needs.

Part of Speech Tagging: The performance during training (Figure 5.4) highlights the large degree of stratification of information that the IR network is undergoing when learning relevance. Reflected in the loss function, the initial embeddings retain the most POS information while the subsequent LSTM layers suffer a decrease in F1 within the IR model. However, moving from the first to second LSTM layer in the core IR model receives a much greater 50% loss in performance. This large degradation suggests that as a somewhat low level feature, POS information is still captured in the hidden representation of the higher LSTM layer albeit in a much weaker representation. The slower slope of the loss function on LSTM_2 and significantly degraded F1 score, combined with Palangi et al.'s [93] work on LSTM networks learning a rough topical model, suggests that the probe is learning to recognize more abstract topical representations and mapping them to POS labels. The difference in performance across query and passage representations indicates that the IR network attends to POS information equally.

Table 5.3: F1 score for NER and POS, and Accuracy for Sentiment and Entailment tasks of each layer of the IR network over auxiliary NLP tasks with input treated as the query. *Aux* in the second column represents the probes inserted into an identical LSTM network trained directly on the auxiliary task. Parenthesis indicates performance difference when placing <EOS> prior to sample input, bold shows best layer on each task, and '-' is a space placeholder as the values cannot be computed given the mean pooling after the LSTM component.

CQA								
Layer	**NER**		**POS**		**Sentiment**		**Entailment**	
	IR	**Aux**	**IR**	**Aux**	**IR**	**Aux**	**IR**	**Aux**
Random	.200		.022		.500		.333	
Embedding	.963		.917		.844		.590	
LSTM_1	**.927(-.001)**	**.987**	**.751(-.001)**	.951	**.721(-.004)**	.900	.522(+.036)	.715
LSTM_2	.810(-.002)	**.987**	.305(-.002)	**.954**	.666(-.001)	.900	.518(+.040)	.873
Dense_{300}	-	-	-	-	.689(-.005)	.926	**.527(+.039)**	.877
Dense_{200}	-	-	-	-	.668(-.007)	.932	.454(+.031)	.881
Dense_y	-	-	-	-	.498(+.006)	**.934**	.366(-.008)	**.885**

WikiQA								
Layer	**NER**		**POS**		**Sentiment**		**Entailment**	
	IR	**Aux**	**IR**	**Aux**	**IR**	**Aux**	**IR**	**Aux**
Random	.200		.022		.500		.333	
Embedding	.963		.917		.844		.590	
LSTM_1	**.934(-.001)**	.987	**.794(-.000)**	.951	**.638(+.001)**	.900	**.464(-.010)**	.715
LSTM_2	.845(-.001)	**.987**	.386(+.051)	**.954**	.593(-.001)	.900	.425(+.020)	.873
Dense_{300}	-	-	-	-	.572(-.003)	.926	.400(+.018)	.877
Dense_{200}	-	-	-	-	.557(-.001)	.932	.377(+.021)	.881
Dense_y	-	-	-	-	.503(+.003)	**.934**	.355(-.002)	**.885**

Named Entity Recognition: Closely related to POS tagging, we analyze the probes' performance on the NER auxiliary task. Probe performance on the auxiliary network shows a greater need for capturing abstract and contextual information than POS tagging due to the separation in performance of the embeddings and LSTM layers in both loss and F1 over epochs on the auxiliary NER network.

Examining the probes within the IR network reinforces the evidence that the second LSTM layer is learning a more topical representation. However, as the second LSTM layer discards a significant amount of POS information, the sustained performance on the NER task across LSTM layers suggests that the IR model uses named entities for passage length relevance judgements either through explicit capturing at the cell level, or in a latent representation in the hidden layer independent of POS information. The F1 drop when processing the same samples from a passage perspective indicates that the IR network focuses on capturing more information related to named entities when processing text at the query stage. However, the drop in performance could also be due to the lack of relevant query text priming the network to focus on named entity information.

Sentiment Classification: Moving to a more abstract task requiring an entire sentence, probes trained to label sentiment result in a significantly different outcome than NER and POS. Each layer remains a close neighbor to its subsequent one when viewed from the probes' perspectives. The small decrease in sentiment classification performance, accompanied with the large loss of POS information suggests that some form of more abstract sentiment information is captured in each layer. However, while sentiment information is used for establishing relevance, there is no signal present in the actual relevance label, as shown by Dense$_y$'s result of 0.49 and the random model receiving a 0.50 accuracy score. In addition, the IR network does not seem to process sentiment information differently across query and passage text as seen by the relatively stable performance in Table 5.3.

Entailment: Confirming the results in the previous paragraphs, the most abstract task of capturing entailment suffers the least across layers. Additionally, contrary to the other auxiliary tasks, higher layers significantly outperform the lower ones as seen in the difference between LSTM$_1$ and Dense$_{300}$ in Table 5.3. Accounting for the accuracy across other tasks, the increased performance of the third layer, Dense$_{300}$, suggests that the transformations used to determine relevance at this point also act to move entailment classes into a more linearly separable space.

Lastly, the performance of the probe on the relevance score, Dense$_y$, shows that the relevance of a query passage pair has some information with respect to logical entail-

Table 5.4: Per label accuracy performance over SNLI entailment collection on CQA model. E, N, C represent the classes *Entailment, Neutral*, and *Contradiction.*

Layer	**E**	**N**	**C**
Random	.333	.333	.333
Embedding	.593	.531	.624
LSTM_1	.569	.466	**.517**
LSTM_2	.596	.473	.470
Dense_{300}	**.610**	**.476**	.480
Dense_{200}	.529	.403	.422
Dense_y	.424	.372	.296

ment. We expand this insight and investigate individual label performance as shown in Table 5.4. The individual label evaluations show that each of the three classes requires unique information. In addition, the relevance model retains information for detecting entailment, while information for neutral and contradictory labels is iteratively discarded at each layer. The following dip in performance in Dense_{200} indicates that the upper layers put less emphasis on entailment. Finally, looking at the relation between the scalar relevance value, Dense_y, and the individual label metrics shows that positive entailment information is related to the relevance of a passage, although non relevant documents provide no indication that the query and passage pair do not contain some type of entailment.

5.1.2.1 Multitask Inspection

Examining the impact of the auxiliary loss signal for IR, the same trend as seen in Table 5.3 occurs in Figure 5.5, where the layer that captures the most information with respect to the auxiliary task is also the most effective layer within the multitask environment for retrieval. Of particular interest is the NER performance on WikiQA. This task significantly improves performance when using LSTM_1 as the shared layer, and subsequently suffers the greatest performance decrease across all tasks when moving upward. This suggests that for retrieval on this collection, the use of named

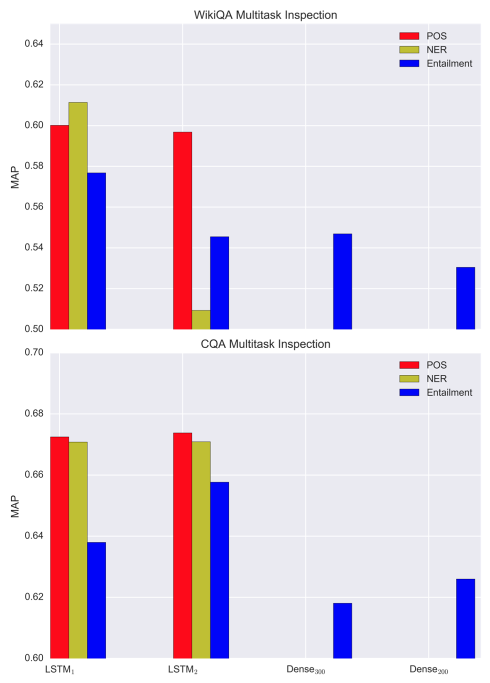


Figure 5.5: Per-layer performance of NER, POS, and Entailment tasks measured by MAP on WikiQA and CQA collections.

entities within a neural model is heavily biased towards the first layer, not only from an information perspective, but also from a performance view as well. Lastly, following the trend in Table 5.4, the multitask model over CQA benefits from using LSTM$_2$ as the shared layer, where the most information used for entailment is captured. This also demonstrates that the optimal shared multitask layer for retrieval is not the lowest by default, as it is for POS and NER auxiliary tasks.

5.1.2.2 Dataset Comparison

WikiQA vs CQA: As mentioned in section 5.1.1.6, we perform the same NLP auxiliary analysis on an additional factoid QA dataset. Shown in Table 5.3, there exists a consistent decline in performance from the lower to upper layers. However, due to the greater amount of factoid type queries, the WikiQA model retains more information with respect to NER and POS information at the cost of reduced performance on sentiment and entailment tasks. Not only does the WikiQA model perform worse than the CQA model on these tasks, but the hidden transformations used for determining relevance fail to provide any assistance regarding separating entailment unlike the CQA model. While the WikiQA dataset shares significantly less vocabulary overlap than the CQA collection as seen in Table 5.2, examining the impact of missing vocabulary on the incorrectly classified auxiliary samples reveals a Pearson's correlation of 0.194, and restricting the CQA collection to the same as the WikiQA training set, 12,888 random samples, does not significantly reduce performance on the auxiliary tasks. This provides insight in why some past models that perform successfully on shorter QA tasks struggle on passage retrieval [20].

5.2 Towards a Universal Retrieval Function - Temporal Difference Updates for Information Retrieval

Having identified substantial information overlap across collections in the previous section, we introduce a modification of an established reinforcement learning method to facilitate widespread use of a branch of reinforcement learning for IR, temporal difference (TD) learning needed for recent work in task transfer with neural architectures over sparse collections [10, 23]. While reinforcement learning methods have shown success in document ranking, these contributions have relied on a specific policy gradient method: REINFORCE [136, 140, 142]. This brings associated issues like high variance gradient estimations and sample inefficiency, which is a pronounced problem in answer passage retrieval over noisy collections with limited training data. Within the reinforcement learning community, there exists a substantial body of work on alternative methods of training which revolve around temporal difference updates, such as Q-learning, Actor-Critic or SARSA, and resolve some of the issues seen in REINFORCE [58, 115]. However, TD methods require the full size of the state to be modeled internally within a single ranking instance, which is unrealistic for neural text retrieval as a single model would have to process $|C|$ documents simultaneously for each query where C is the set of documents in the collection. We therefore propose interpolated sub-state temporal difference (ISSTD), operating on the sub-state, or individual documents in the case of matching models, and interpolating the temporal difference updates to the rest of the state, or candidate documents. We further demonstrate that traditional IR neural models that score documents independently are a valid approach under ISSTD, allowing us to use universal retrieval functions (URF) with any current ranking architecture.

5.2.1 Reinforcement Learning for Information Retrieval

Reinforcement learning presents a framework for an agent to learn from its actions within an environment. This can range from canonical examples like robotic control [141] and game playing [44] to even machine translation [139]. The agent can be a neural model, program, or any process that is able to improve its future actions based on the input it receives from the environment. In the case of game playing, the environment could be Atari, while for machine translation it could be represented as the space of possible translations and training examples. This environment partially determines the reward to convey how well an agent's decisions satisfy some unknown objective. The last core principle of RL is the sequential nature of decision making.

Previously introduced in Section 4.2, we provide a brief overview of RL required for the following work. We defined the MDP as the tuple $(\mathcal{S}, \mathcal{A}, P, R, d_0, \gamma)$ representing the state space, action space, transition function $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, initial state distribution d_0 and decay parameter γ.

The objective is then to find a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes J, the expected total discounted reward $R_t \sim r(s_t, a_t)$ that the agent can obtain,

$$J(\pi) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t R_t \Big| \pi\Big].$$

Closely related to this goal is the value function,

$$V^{\pi}(s) = \mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^t R_{t+k}\Big| s_t = s, \pi\Big] = \mathbb{E}\Big[G_t \Big| s_t = s, \pi\Big], \tag{5.5}$$

which is the expected discounted return if the agent follows policy π from state s. The goal is then to find some π^* that achieves

$$V^*(s) = \max_{\pi \in \Pi} \mathbb{E}\Big[G_t \Big| s_t = s, \pi\Big] \ \forall s \in \mathcal{S}. \tag{5.6}$$

As discussed, there exists a large body of work on finding π^*, with policy gradient as the main approach used for learning to rank [136, 140, 142, 152, 130]. The alternative approach, which we use for URF, is temporal difference (TD) learning. In this framework, π^* is achieved via bootstrapping under the Bellman equation:

$$TV(s) = r(s,a) + \gamma \mathbb{E}[V(s')] \tag{5.7}$$

where s' is the next state visited and will converge under certain conditions [115]. It is straightforward to see where the concept of TD comes from, T updates V based on the expected next state's discounted cumulative reward. Q-learning, SARSA, and other TD based methods update via the function $Q : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ shown below

$$Q^{\pi}(s,a) = \mathbb{E}\Big[G_t \Big| s_t = s, a_t = a, \pi\Big] \tag{5.8}$$

under the same contraction operator T. While V represents the cumulative reward from state s, Q captures the cumulative reward from state s having taken action a. The advantage with Q-learning or SARSA is that one does not need to model the the environment directly, and so these approaches are therefore referred to as model free methods. This is advantageous in large or challenging environments where the transition or reward functions P and R are difficult to capture.

We highlight a key difference between Q-learning and SARSA: the update for Q-learning will always be greedy, meaning that the update under T will take Q

value of whatever action is best in the next state s'. However, SARSA will follow the underlying π and take an ϵ-greedy approach. While this difference only occurs with probability ϵ, Q-learning is off-policy and suffers from divergence under function approximation whereas SARSA will converge [115]. This plays an integral role for interpolated sub state learning later in this work.

Lastly, there or no limits on what defines Q or V in terms of a function. These can be modeled via a table and trained under dynamic programming, a linear function approximator $Q(s, a) = \theta^\intercal \phi(s)$ where ϕ is some state representation, or a non linear function approximator like a neural net. Currently non-linear function approximations of Q or V discard all convergence guarantees when using TD; nonetheless, these non-linear approaches achieve remarkable results in challenging environments [44, 141, 30].

5.2.2 A Markov Decision Process for Deep Neural Retrieval

MDP construction directly impacts what the agent learns, and is a non trivial task when translating real world environments (IR) into this formulation. As such, we adopt the well defined MDP proposed by Wei et al. [136].

State We construct the state $s \in \mathcal{S}$ as

$$[Qry, D_{ur}, t] \tag{5.9}$$

where query of length L, $Qry = [q_1, q_2, ..., q_L]$, D_{ur} is a list of unranked documents where $d_i \in D_{ur} = [d_1, d_2, ..., d_n]$, and t represents the current timestep in an episode. The terminal state occurs when $|D_{ur}| = 1$. A sub-state z_i can then be viewed as $[Qry, d_i, t]$ where $d_i \in D_{ur}$.

Action: At each step, the policy chooses a document to rank next from the set of unranked documents, D_{ur}. Each candidate represents an element in the set of actions $\mathcal{A}$.

Reward: We use reciprocal rank as defined below, where D_r is the set of all relevant documents with respect to Qry and D_{nr} are all other non-relevant documents.

$$r(s,a) = \begin{cases} 0 & a_t \in D_{nr} \\ \frac{1}{t} & a_t \in D_r \end{cases}$$

Transition: As the transition function $P : \mathcal{S} \times \mathcal{A} \rightarrow S$ models the dynamics of the environment and maps a state, action pair to a new state. We structure the transition function as

$$P(s_t, a_t) = [Qry, D_{ur} \setminus d_{a_t}, t+1],$$

where d_{a_t} represents the document chosen at step t by action a_t.

We set D_{ur} in the initial state d_0 to be n documents from the collection generated via BM25 [52] from a random query in the training set. We select $n = 10$ for ease of analysis in our evaluation of our findings.

In this section, we discuss why TD methods fail in the current IR regime, and then formally introduce the ISSTD to gracefully handle these conditions. A typical ranking model scores each document independently and a ranking is produced from these independent scores. For each query, one can view individual documents d_i of a collection as a sub-state $z_i \in s$. In this representation, z_i is a partition of s such that $\bigcup z_i = s, \ \bigcap z_i = \varnothing$. However, a state $s \in \mathcal{S}$ consists of all documents to be ranked for a query or session. As discussed, policy gradient based methods are a drop in optimization method to directly optimize a non-convex or discrete metric. These methods accomplish this by creating a distribution over all possible documents via the softmax function when selecting the next document:

$$\pi(s) = \frac{\exp(f_{IR}(q, d_i))}{\sum_{j \in |D_{ur}|} \exp(f_{IR}(q, d_j))} = \frac{\exp(f_{IR}(z_i))}{\sum_{j \in |Z|} \exp(f_{IR}(z_j))}.$$

This representation enables independent computation of $f(q,d)$ regardless of the amount of candidate documents in s. In the case of TD learning, this isn't necessarily true. For example, in Q-learning [134] where the TD update occurs via

$$Q(s,a) = Q(s,a) + \alpha(s,a)(r(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a)), \tag{5.10}$$

where s', a' is the next state, action in the environment. The update, in either table lookup, linear, or non-linear function approximation is reducing the error for the entire state's representation. However, in the case of IR situations, we are unable to model a single s entirely, and substitute $Q(s,a)$ on the right hand side of Equation 5.10 for the IR model operating over a single document. Thus the actual update that would occur is

$$Q(s,a) = f_{IR}(z) + \alpha(s,a)(r(s,a) + \gamma \max_{z'} f_{IR}(z') - f_{IR}(z)). \tag{5.11}$$

This now computes the value of a sub-state trajectory through the MDP, removing information about actions except for the reward and learning rate and assigns it to the entire state. In order for TD to be used in IR, it would require careful construction of an MDP to avoid any sub-state computation or a significantly large compute cluster to handle the entire s efficiently in the case of deep learning, allowing the use of Equation 5.10. Hu et al. [48] demonstrate one such construction, but this relegates TD based methods to niche problems despite their powerful abilities [44, 30]. In addition, competitive TD methods rely on large memory replay buffers, which compounds this issue due to the need to recompute $2|b_m||s|$ documents for each update where $|b_m|$ is the batch size of the buffer. In the remainder of this section, we introduce machinery such that Equation 5.11, with a small modification, is a valid substitute for Equation 5.10 without loss of guarantees and facilitates the drop in use for any typical IR task by showing that a standard learning to rank process is a special case of an interpolated value function maintaining convergence properties.

5.2.3 Interpolated Sub-State Temporal Difference Learning

Consider $\theta \in \mathbb{R}^b$ as the paramaterization of some $Q_\theta : B(\mathcal{S} \times \mathcal{A}) \to \mathbb{R}$ which acts as a function approximator for Q. Then shown in Szepesvári and Smart [117], a TD update under the Bellman operator can be done via

$$\Delta\theta_{ti} = \alpha_{ti}\beta(s_i, a_i, s)(R_t + \gamma \max_{a'} Q_{\theta_t}(s', a') - \theta_{ti}), \tag{5.12}$$

where $s_i = \{(s_1, v_1), \ldots, (s_n, v_n)\}$ is a set of basis points of $\mathcal{S}$, Q_{θ_t} is the Q function paramaterized by θ_t and is a barycentric interpolator over $\mathcal{S}$, α_{ti} is the learning rate for dimension i of θ at time t, and β is a bounded measurable smoothing function. A key note is that in this formulation, β allows the potential updating of multiple components of θ_t for each basis point. We introduce two definitions necessary to show equivalence to Equation 5.11, enabling the full use of TD methods for IR, along with potential modification to improve sample efficiency.

Definition 5.2.1. $\mathcal{P} : B(\mathcal{S}) \to \mathbb{R}^b$ is a composite pointwise evaluation operator with respect to a fixed set of basis points $Z = \{(z_1, v_1), \ldots, (z_n, v_n)\}$ if $(\mathcal{P}V)_i = V(s_i)$.

Definition 5.2.2. Let $F : \mathbb{R}^b \to B(\mathcal{S})$ be a mapping from parameters to functions. Then F is interpolative with respect to the set of basis points S if for all $V \in B(\mathcal{S})$, $\mathcal{P}F\mathcal{P} = \mathcal{P}$.

Lemma 5.2.1. The zero order spline interpolation F, $Fu = \sum_{i=1}^{b} u_i \mathbf{1}_{A_i}$is a measurable non-expansion in the sup-norm over some basis set Z and $\mathbf{1}_{A_i}$ as the measurable indicator function of A such that

$$\mathbf{1}_{A_i}(x) = \begin{cases} 1 & x \in A_i \\ 0 & x \notin A_i \end{cases}$$

Proof. We define A_i to be a partition of Z such that A_i covers the k-nearest neighborhood around z_i with $k = 1$. This results in the a piecewise continuous evaluation over basis Z where $Fu_i = u_i$ for the entire space space A_i around point z_i. Then we observe $||Fu||_\infty = ||u||_\infty$. For any u, g in the same Banach space,

$$||Fu - Fg||_\infty = ||u - g||_\infty \tag{5.13}$$

which satisfies the requirement of a non-expansion. F is also piecewise continuous and measurable as the sum of measurable functions, i.e.$\sum_{i=1}^{b} u_i \mathbf{1}_{A_i}$ is also measurable. $\square$

Let $V : B(\mathcal{S}) \to \mathbb{R}$ be a linear function approximator defined by Equation 5.5, then the sub-state update

$$V(s) = f_{IR}(z) + \alpha(s)(r(s,a) + \gamma \mathbb{E}[f_{IR}(z')] - f_{IR}(z)) \tag{5.14}$$

converges to a $\hat{V}^*$ if (i) $\mathcal{P}$V exists, (ii) F is a non-expansive interpolation such that $||Ff_1 - Ff_2|| \leq \gamma ||f_1 - f_2||$, and (iii) $Z = \{(z_i, v_i)\}_1^n$ is the set of basis points of s such that Z creates a partition of s with $\bigcup z_i = s, \ \bigcap z_i = \varnothing$.

Proof. In order to do so, we use the algorithm

$$\theta_{t+1} = \mathcal{P}TF\theta_t, \tag{5.15}$$

where T is the Bellman operator, and $\mathcal{P}$ is composite pointwise evaluation operator, and $F : \mathbb{R}^b \to B(\mathcal{S})$ [34].

Then we can see that value iteration can be modeled via

$$V_{t+1} = TF\mathcal{P}V_t. \tag{5.16}$$

This provides the critical first step of the drop in placement of TD updates for learning to rank. In this representation, $\mathcal{P}$ acts as a decomposition of V to the point wise evaluation of the expected return of that basis point. As $F\mathcal{P}$ is a non-expansion, then $F\theta_t$ converges to $\hat{\theta}^*$ [34]. We observe a direct mapping of a satisfactory $\mathcal{P}$ to the case seen in Equation 5.14 where f_{IR} is a composite pointwise evaluation of V such that each basis point consists of d_i. Formally,

$$\theta_{IR}(z_i) = (\mathcal{P}V)_i \tag{5.17}$$

deconstructs some larger retrieval model that fully captures all possible documents within a reranking list or a collection as its input and decomposes it to individual functions over each z_i.

Next, we select an F that can interpolate the mapping from the parameter θ to a $V \in B(\mathcal{S})$ with respect to S. Intuitively, Definition 5.2.2 means that $F(\theta)(z_i) = v_i$ at each z_i in S. Then the operation $\mathcal{P}$ acts as an initial decomposition of V and F interpolates its values across the parameter space. Szepesvári and Smart [117] prove that if $F\mathcal{P}$ is a interpolative non-expansion, then $V_{t+1} = F\mathcal{P}TV_t$ converges to V^*. By Lemma 5.2.1, a zero order spline is a non-expansion. As the smoothing is 0 everywhere except where the characteristic function is active, $\beta(s_i, a, s)$ in Equation 5.12 becomes an indicator variable. This results in $\beta_{ti} = 0$ for everywhere except for basis s_i, or z_i in the case of the initial IR learning to rank TD based Equation 5.11.

Thus, as long as

$$\sum_{i=0}^{\infty} \alpha = \infty, \qquad \sum_{i=0}^{\infty} \alpha^2 < \infty \tag{5.18}$$

meaning all states are visited infinitely often, then $TV(z) \rightarrow \hat{V}^*$ and $TQ \rightarrow \hat{Q}^*$ even when individual sub states are updated independently for value iteration and Q-learning respectively [117]. $\square$

While we show that typical learning to rank updates are a special case of interpolation, observe that more suitable smoothing functions such as radial basis function kernels or Gaussian processes satisfy the non-expansive properties of F while potentially improving sample efficiency. We leave the evaluation of alternative smoothing methods to future work.

Corollary 5.2.1.1. Let $|| \max_z f_{IR}(z) - Q(s,a)|| \leq \epsilon$ such that

$$\mathbb{E}[|| \max_z f_{IR}(z) - Q(s,a)||] = 0$$

under any MDP such that every state is visitable, $\lim_{t\to\infty} P(s_t = s_{\text{terminal}}) = 1$, and $\sum_{i=0}^{\infty} \alpha = \infty, \quad \sum_{i=0}^{\infty} \alpha^2 < \infty$, then

$$Q_{ISS}(s,a) = f_{IR}(z) + \alpha(s,a)(r(s,a) + \gamma \max_{z'} f_{IR}(z') - f_{IR}(z)) \tag{5.19}$$

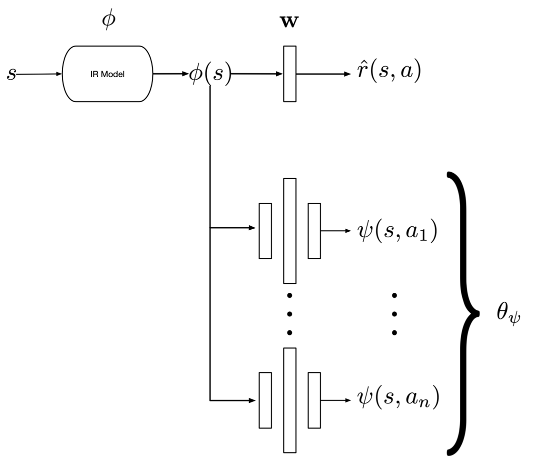
oscillates within a region C of the fixed a point $\hat{Q}^*$ defined by

$$Q(s,a) = Q(s,a) + \alpha(s,a)(r(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a)) \tag{5.20}$$

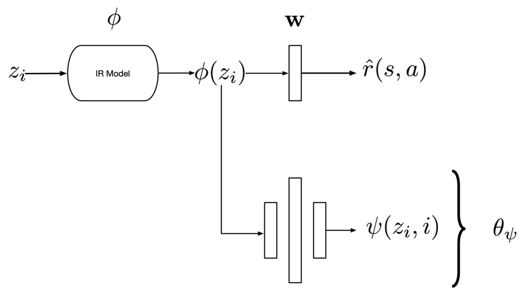
Proof. If we treat a π that operates on an entire s as an epsilon greedy $\pi(s) = \text{argmax}_a Q(s,a)$, then Q_{ISS} follows the exact policy as the underlying generating policy Q and becomes SARSA in this instance. We then represent Q_{ISS} to be a linear function approximator of Q. If we define $Q(s,\cdot)$ as

$$Q(s,\cdot) =< f_{IR}(z_1), \ldots, f_{IR}(z_n) > \tag{5.21}$$

then, we observe that Q_{ISS} is the operation $\theta_{ISS}{}^{\intercal} Q(s,\cdot)$ and acts as a linear function approximator over Q such that $||\theta_{ISS}||_\infty = 1, ||\theta_{ISS}||_2 = 1$. Then this is a linear function approximator of the π generated by following Q-learning, .i.e. SARSA.



(a) SF Deep Learning Architecture



(b) ISS-SF Deep Learning Architecture

Figure 5.6: Caption

We leverage the result from Gordon [35] that states that SARSA will converge to region around the generating π under linear function approximation. Thus Q_{ISS} will converge to a region C around the fixed point $\hat{Q}^*$. □

Having established that operating on individual documents is a valid extension of TD methods over the entire candidate document set without the need to develop specialized retrieval models, we now introduce the TD method used for URF with the ISS modification.

5.2.4 Successor Features for IR

With the machinery developed above, we are now able to incorporate successor feature (SF) learning [9], a powerful TD technique with unique properties which pair well with IR tasks. We introduce SF learning in the standard s, a notation for clarity, and apply $F\mathcal{P}$ discussed above during training.

This framework offers two significant advantages over conventional Q-learning: (1) it formally separates representation and relevance judgements, and (2) it better captures uncertainty of a current state due to the *successive* state representations [51, 9]. We provide a brief outline of this approach.

The SF representation is based on the concept that the reward function $r(s,a)$ can be decomposed into an inner product of a state representation $\phi : \mathcal{S} \to \mathbb{R}^K$ and reward vector $\mathbf{w} \in \mathbb{R}^K$ such that

$$r(s,a) = \phi(s,a)^\intercal \mathbf{w}. \tag{5.22}$$

This representation is not restrictive to any environment as it can be trivially deconstructed to recover any reward function. We derive SF, ψ^π ,by incorporating ϕ and $\mathbf{w}$ into the standard Q-function to produce SF ψ^π.

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}[\Sigma_{i=0}^{\infty} \gamma^i r(s_i,a_i) | S_0 = s, A_0 = a, \pi] & (5.23)\\ &= \mathbb{E}[\Sigma_{i=0}^{\infty} \gamma^i \phi(s_i,a_i)^\intercal \mathbf{w} | S_0 = s, A_0 = a, \pi] & (5.24)\\ &= \mathbb{E}[\Sigma_{i=0}^{\infty} \gamma^i \phi(s_i,a_i) | S_0 = s, A_0 = a, \pi]^\intercal \mathbf{w} & (5.25)\\ &= \psi^\pi(s,a)^\intercal \mathbf{w} & (5.26) \end{aligned}$$

To conceptualize what ψ means: in the tabular case such as Gridworld using one hot encodings, the i^{th} component of ψ^π is the discounted sum of occurrences of reaching (s_i, a_i) of each possible transition while following π. As SF maintains linearity across time, any TD method can be used,

$$\psi^\pi(s,a) = \phi_(s,a) + \alpha[\gamma\psi^\pi(s',a) - \psi^\pi(s,a)], \tag{5.27}$$

and therefore can be trained in the same manner as Q-learning, referred to as SFQL. With Proposition 5.2.3 and Corollary 5.2.1.1, we see that this new approach can be applied within ISS framework (ISS-SFQL). In the non-tabular case where a gradient is used to learn ϕ, $\mathbf{w}$, and ψ, the optimization occurs via a two step process where ψ is optimized via the loss function:

$$\mathcal{L}(\theta_\psi) = \mathbb{E}[||\phi(s) - \gamma \max_{a'} \psi(s', a') - \psi(\phi(s), a)||_2^2] \tag{5.28}$$

and $\mathbf{w}, \phi$ via:

$$\mathcal{L}(\theta_{\mathbf{w}}, \theta_\phi) = \mathbb{E}[||r(s, a) - \phi(s)^\intercal \mathbf{w}||_2^2] \tag{5.29}$$

Algorithm 2 provides an overview of the entire learning process and Figure 5.6 illustrates the overall framework. We adopt target networks and a memory buffer as recommended in Hessel et al. [44] to improve stability of deep Q-learning agents during training.

Algorithm 2: Approach for ISS-SF.

```
Input: Memory replay $\mathcal{B}$, parameters $\theta_\phi, \theta_{\mathbf{w}}, \theta_\psi$, exploration probability
  $\epsilon \leq 1$, MDP: $(\mathcal{S}, \mathcal{A}, P, R, d_0, \gamma)$
for episode $l = 1$ to $L$ do
    initialize $s \sim d_0$
    if Bernoulli($\epsilon$) then
        sample action $a$ uniformly
    else
        $\phi(s) = \{f_{IR}(z_i)\}_0^{|s|}$
        $a = \text{argmax}_i \; \psi(z_i, i)^\intercal \mathbf{w}$
    Store transition $(z, i, r(s, a), s')$ in $\mathcal{B}$
    Randomly sample minibatch from $\mathcal{B}$
    Update $\theta_\phi, \theta_{\mathbf{w}}$ via Equation 5.29
    Update $\theta_\psi$ via Equation 5.28
    $s \leftarrow s'$
```

By defining a SF approach for IR, we force search to be decomposed into a representation component and a relevance component. Furthermore, $\psi^\pi(s, a)$ captures expected future steps within its construction forcing our model to predict a multi-step function even in sparse environments. Within the realm of IR, this is akin to ranking trajectories of documents such that when given new documents $\phi(s_t)$, the model needs to predict what would be the most relevant decision given potential related documents captured in ψ^{π}.

5.2.5 Experimental Setup

To evaluate the efficacy of TD based updates for IR, we examine representative IR models of varying complexities as policies over MSMARCO and Yahoo L4. The datasets were chosen due to their significant performance increase when used with deep learning models, requiring the use of ISS modification for Q-learning and SFQL.

5.2.5.1 Data

MSMARCO: [89] This collection is based on Bing queries and their corresponding results. Originally proposed for a question answering task, the annotated data was used to create a passage re-ranking task. The collection consists of 400M training tuples of query, relevant, and non-relevant passages with the development set containing ∼6,900 queries with each query corresponding to the top 1,000 passages retrieved via BM25.

Yahoo L4: [113] Consists of non-factoid questions which take the form of "Manner" questions. The collection consists of ∼142,000 queries and corresponding answers. The train, dev, and eval splits were 80%, 10%, and 10% respectively.

5.2.5.2 IR Policies – Network Architectures

In order to demonstrate the feasibility of a TD based RL approach for answer passage retrieval, we use three different representative architectures as policies for this task which cover recurrent, convolutional, and transformer paradigms. We note the absence of a BERT based model, as the experiments are to demonstrate the feasibility of ISSTD learning to varying architecture complexities. As BERT is used as a pretrained base [144], it does not offer additional insight into the behaviour of

ISS. We note that while past works have effectively incorporated RL techniques into IR, they have not used neural models beyond a single layer perceptron successfully.

MatchPyramid: This model consists of an initial interaction matrix of the query and document's embeddings that acts as an input into a series of convolutional layers. While a deep network, this model has shown to perform well on only a small number of training queries, as demonstrated on Robust04 and other TREC collections [94].

BLSTM: As these architectures have been used extensively for retrieval and are well studied [128, 82, 93], they provide a stable candidate to evaluate ISS-SFQL. The model consists of concatenating the query and document and feeding it into a bidirectional LSTM model. Max pooling over time is used prior to an upper feedforward network.

Transformer Kernel: Lastly, we introduce a state of the art model representative of the current neural retrieval architectures [126]. This approach uses transformer modules with a kernel pooling approach to learn a relevance score between a query and candidate document, and is the largest model of the architectures evaluated in this paper [46].

As ϕ is a multidimensional representation of a document, the standard architecture of the above models results in a scalar output. Therefore, we use the layers prior to the final output as ϕ and fix the output of each IR model to a fixed dimension. MatchPyramid and BLSTM use a 200 dimension final layer, and Transformer Kernel uses $\phi \in \mathbb{R}^{22}$. While significantly smaller than the other two approaches, this is due to the number of kernels used in the original paper and already offers a salient and compressed representation of the query-document relation.

5.2.5.3 Optimization

As SFQL is an optimization method over a neural model, π, we evaluate performance over the two other common learning methods: Pairwise hinge loss (supervised) and REINFORCE (PG).

Method	BLSTM	TK	MP
REINFORCE	0.288	0.319	0.304
ISS-Q-Learning	0.409	0.421	0.416
ISS-SFQL	0.556	0.535	0.484
Hinge loss (Supervised)	0.542	0.587	0.545

Table 5.5: Performance of training regimes over Yahoo L4 data evaluated via MRR.

Method	BLSTM	TK	MP
REINFORCE	0.338	0.295	0.306
ISS-Q-Learning	0.331	0.325	0.374
ISS-SFQL	0.541	0.653	0.488
Hinge loss (Supervised)	0.552	0.687	0.481

Table 5.6: Performance of training regimes over MSMARCO passage validation data under MRR.

All models and optimization methods use the Adam optimizer with tuned learning rates via search over $[10^{-5}, 10^{-1}]$. We initialize all embeddings with GloVE [96] with dimension of 300. Other hyperparameters for the models were taken from their corresponding works and demonstrated robustness across collections. The memory buffer for ISS-SFQL was constructed from $\{500, 2000, 10000, 20000\}$, and the target network and policy's updates per steps were selected from $\{1, 5, 10, 50\}$. The memory was not reset at each new episode. γ was selected from $\{0.5, 0.7, 0.9, 0.99\}$. We discuss the sensitivity of the algorithm to these hyperparameters below.

5.2.6 Results and Discussion

In this section, we investigate whether the results from Proposition 5.2.3 and Corollary 5.2.1.1 hold for real world examples and can achieve competitive results on deep neural architectures. In addition we perform a hyperparameter study to gain insight into the properties of ISSTD for IR.

5.2.6.1 Performance Benchmarks

Examining the performance on Yahoo L4 and MSMARCO, we see consistent performance across models and training regimes as shown in Tables 5.5 and 5.6. We remark on the competitive performance of ISS-SFQL, suggesting that ISSTD approaches are viable for IR tasks even in the case of deep neural models. While not the purpose of this paper, the surprising result of the close performance difference between the supervised pairwise hinge loss and ISS-SFQL suggests that RL methods can act as an effective training alternative to supervised approaches even in a sparse reward setting. Furthermore, this suggests that ISS-SFQL is a powerful option when one needs to include non-differentiable signals into a retrieval model such as diversity, fairness, or personification. Given the nature of instability of RL methods and the stability of supervised approaches, this provides promising insight into situations where no supervised training signal is available, such as query reformulation [85] or online recommendation systems where the input is significantly large. However, in the case of TK on MSMARCO, ISS-SFQL is not able to reach parity with the supervised approach whereas it does with BLSTM and MP models. As Popel and Martin [98] discuss, the transformer architecture is sensitive to both noise in the data and its updates. Q-learning methods essentially bootstrap up to an effective model so the additional noise introduced via this method prevents TK from properly learning.

Of particular note is the poor performance of REINFORCE. We attribute the close to random performance of these policies due to the point collapse of the policy. While training, the distribution $\pi(s)$ would collapse to a single action despite our efforts due to the gradient variance introduced via Markov chain Monte Carlo sampling [137] and the non-linear function approximation. We are careful to comment that this is not indicative that policy gradient methods are worse than TD approaches. However, by incorporating additional structures into a TD framework (SF), we can significantly increase the performance of RL based methods in cases where REINFORCE fails.

While not examined in this work, ISS-Actor Critic or ISS-DDPG could be alternative, more stable, options to use policy gradient in IR.

5.2.6.2 Convergence

Table 5.7 reflects the benefit of incorporating novel RL methods. ISS-SFQL is able to quarter the required number of episodes compared to ISS-Q-learning in order to converge to a stable policy with respect to performance on a validation set. Furthermore, ISS-SFQL only requires twice as many samples as the supervised approach while undergoing a bootstrap procedure. Shown in Figure 5.7, we observe the behaviour of ISS-SF learning for IR. Characteristic of RL, we see an initial plateau around 10,000 samples prior to the agent identifying an effective retrieval function and continually refining it.

Method	**BLSTM**	**TK**	**MP**
REINFORCE	∞	∞	∞
ISS-Q-Learning	210k	410k	205k
ISS-SFQL	35k	135k	35k
Hinge loss (Supervised)	20k	90k	15k

Table 5.7: Number of episodes (queries) needed to converge to a stable retrieval model on Yahoo L4. Evaluation was done every 5,000 episodes. While epoch training is common in supervised approaches, we maintain the same sampling method for queries for all methods to maintain consistency. ∞ denotes that the method diverges.

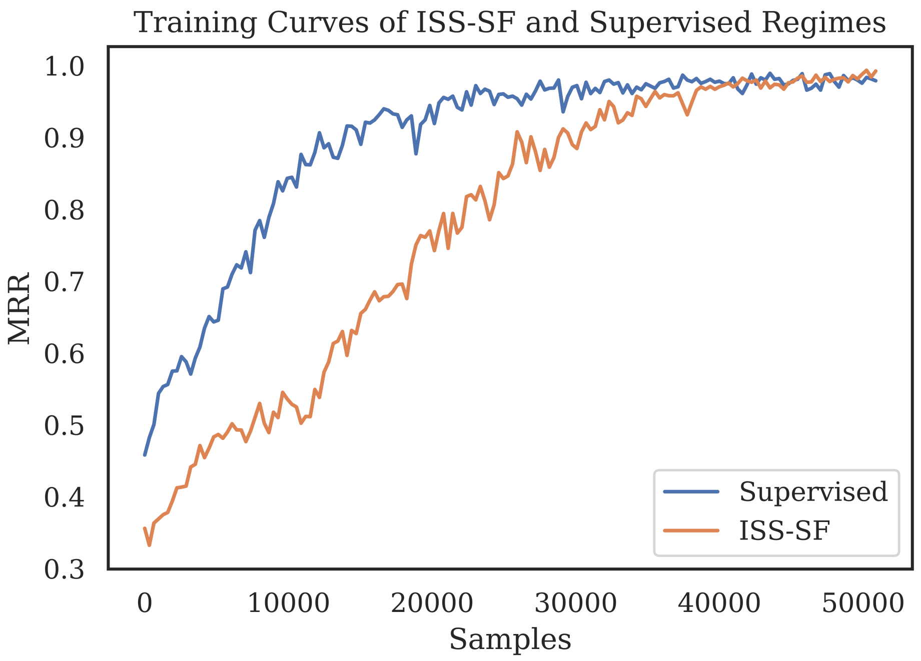
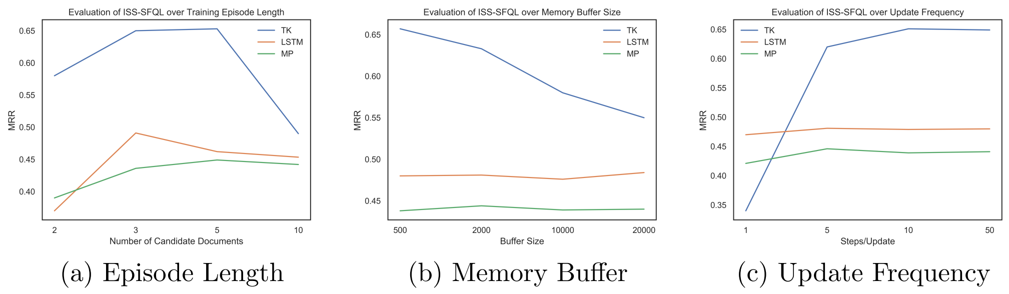


Figure 5.7: Training set curves of ISS-SF and hinge loss approaches.

5.2.6.3 Hyperparameter Sensitivity



(a) Episode Length (b) Memory Buffer (c) Update Frequency

Figure 5.8: Performance of ISS-SFQL across key hyperparameters on MSMARCO dataset.

Any RL algorithm has a substantial amount of hyperpameters which can drastically impact performance. We highlight the key aspects in Figure 5.8. Following from the discussion of the transformer's architecture being sensitive to noise, we observe a similar sensitivity to hyperparameters. One example of this is the impact of the update frequency of $\phi, \mathbf{w}$, and ψ. As the actual update to these values occur via minibatch from the memory buffer, updating too frequently has the potential to introduce too much noise into these minibatches. By slowing down the update frequency such that the model changes once every m steps, we increase the likelihood of sampling a more uniform batch to update.

This sensitivity is supported by the impact of replay size. We observe a decrease in performance with respect to TK as the memory buffer size increases. As Liu and Zou [68] discuss the impact of buffers with respect to agent performance, it is not uncommon to see this sensitivity. We hypothesize that it is due to storing older transitions which no longer benefit the agent during the minibatch updates. While the more stable BLSTM and MP models are robust to this noise, the transformer based architecture, TK, diverges.

Lastly, we examine the episode length, or initial size of D_{ur} from the MDP formulation. For reference, when set to 2, this mirrors the bandit situation as the final document acts as the terminal state and no future decisions need to be considered. Therefore, as the agent is allowed to make more decisions and observe more documents, it is able to better determine what is a relevant document and what is not. This can be potentially attributed to the multi-step nature of ψ. Not only does it need to estimate the reward directly via $\mathbf{w}$, but the construction requires the prediction of the subsequent documents. So when given a query and document $\phi(z_i)$, ψ has to estimate what other similar documents are in the collection. This auxiliary task defined in SF updates is supported by the episode length curve. The sharp decline for TK when the episode length equals 10 is due to limited capacity of its ϕ, which only has 22 dimensions by construction [46]. In this case, a slight modification for ISS-SFQL by expanding the number of kernels used might be a better alternative than directly truncating the final layer. A similar saturation occurs in the case of MP and BLSTM models, though without the performance drop off observed for TK.

5.3 Universal Retrieval Functions

Having defined $\mathbf{w}, \phi$ and ψ, as well as the heavily shared information across non-factoid QA collections as shown in Section 5.1, we are now able to construct the framework for URF which naturally follows from SF. The motivation behind URF is that every IR collection consists of ordered permutations created from the vocabulary V. If we extend V to cover words found across multiple collections, or the common crawl [96], we are able to cover the majority of text found in an IR collection. In the case of SF, we can represent the useful information for IR over the permutations of terms in V as ϕ, and then learn small relevance functions ψ and $\mathbf{w}$ that are sensitive to the relevance signal found across the different information needs of each collection. With a stable ϕ for representing text across multiple collections, we can create a federated IR model (URF) from $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_N]$. While saving a traditional neural IR model for each collection and evaluating these models on every new IR collection is infeasible due to the cost of storing and executing each full model, ϕ is recycled and ranking only requires N inner products. Furthermore, an URF framework can easily be extended to new collections by combining the relevance functions $\mathbf{W}$ if the information need of the queries is sufficiently close to a previous collection, or update $\mathbf{W}$ with $\mathbf{w}_{N+1}$. As we demonstrate, there exists substantial overlap when $N = 2$ for certain task combinations, enabling $\mathbf{W}$ to grow slowly.

5.3.1 Multitask to Multicollection Retrieval

We frame the multitask retrieval problem as a multicollection relevance model. Let $\{M_1, M_2, \ldots\} = \mathcal{M}$ represent the set of all collection-query pairs and their corresponding relevance needs defined by human judgements or click through logs. The objective then is that for any permutation of basis tasks B, a retrieval model trained on these basis tasks and a new task M_i not in B should be able to achieve parity with a retrieval model trained directly on M_i and sustain performance on all basis tasks under performance metric $\eta(f, M)$. Formally, given a subset $\mathcal{M}_B \subset \mathcal{M}$, $M_i \notin \mathcal{M}_B$ and a retrieval model f, $\eta(f_{\theta|\mathcal{M}_B,M_i}, M_i) = \eta(f_{\theta|M_i}, M_i)$ while maintaining $\eta(f_{\theta|\mathcal{M}_B}, M_b)\ \forall b \in B$. Furthermore, an effective multitask retrieval approach will minimize the samples needed to determine which M_j most closely resembles M_i as well as the number of updates needed to converge on parameters $\theta|\mathcal{M}_B, M_i$. By addressing these secondary constraints, an ideal multitask framework will be able to quickly infer which collection is most similar to a new collection and fit an effective ranking function, making this approach viable for sparse collections.

While past work has investigated the use of multitask training for IR, the *task* is often framed over facets of IR such as query completion-ranking, NLP-ranking or recommendation-ranking rather than ranking multiple collections [3, 69, 149]. Furthermore, the domain regularization and domain shift approaches such as the one introduced in Section 4.1 and subsequent work do not seek to preserve information on the original collection as much as it seeks to identify salient information to transfer [123]. The area of work that most closely resembles the multicollection framework leverages large models. Nogueira et al. [90] use a large sequence-to-sequence model pretrained on a variety of NLP tasks with the final upper layer fine tuned for the specific IR task. Yang et al. [144], use a similar approach with a different large pretrained model, BERT. Lastly, Yilmaz et al. [4] directly apply BERT to the multicollection objective, treating BERT as ϕ, and using multiple collections to iteratively tune a general relevance model on top. However, the authors do not report the results of the tuned model on the original collections.

In Section 5.2.2, we defined the single collection MDP, $(\mathcal{S}, \mathcal{A}, \mathcal{R}, P, R, d_0, \gamma)$, to describe the environment over a single collection. $\mathcal{S}$ then represented the set of possible documents permutations within a collection, with $s \in \mathcal{S}$ as the actual candidate set of documents to be ranked for a given query. We expand the notion of $\mathcal{S}$ to cover all answer passage documents such that the MDP defined on Yahoo L4's, $\mathcal{S}_{L4} \subset \mathcal{S}_{\mathcal{C}}$, where

$\mathcal{C}$ is the collection of all document permutations across all candidate collections [9]. This reconstruction of our MDP allows us to extend the notion of the environment to a general representation

$$\mathcal{M} := \{(\mathcal{S}_{\mathcal{C}}, \mathcal{A}, \cdot, P, \cdot, d_0, \gamma)\} \tag{5.30}$$

where $\mathcal{M}$ is now the collection of all MDPs over $\mathcal{S}_{\mathcal{C}}$, regardless of their rewards, or relevance judgements. This means that it also includes environments where the rewards are not a linear combination of ϕ constructed in Section 5.2. Thus, $\mathcal{M}_\phi \subset \mathcal{M}$. However, Section 5.1 demonstrates the high amount of shared information, and we propose leveraging this shared information to best cover all of $\mathcal{M}$ despite the limitations of ϕ.

5.3.2 Method

We discuss the two core components of URF: (1) building an initial structure of the environment via ϕ, while collecting linear relevance functions $\mathbf{w}$. (2) After the basis set is completed, we introduce the concept of general policy improvement (GPI), which enables us to transfer information from the basis collections to a new retrieval task [9].

5.3.2.1 Building Basis Functions

We introduce a modified version of ISS-SF's Algorithm 2, which constructs policies, or retrieval models, from multiple collections shown in Algorithm 3 in order to create a basis set of functions. In this setting, we are able to run on N collections in parallel to train our SF state representation ϕ. This can be viewed a multitask framework of ISS-SF. A key advantage of ISS-SF is that ϕ does not model relevance directly, enabling it to retain additional information for use outside of $M_i \in \mathcal{M}$.

```
Algorithm 3: Building URF basis with ISS-SF
  Input: $N$ tasks, Memory replay $\mathcal{B}_{1...N}$, parameters $\theta_\phi, \theta_{\mathbf{w}_{1,...,N}}, \theta_{\psi_{1,...,N}}$,
    exploration probability $\epsilon \leq 1$, MDP: $(\mathcal{S}, \mathcal{A}, P, R, d_0, \gamma)$
  for episode $l = 1$ to $L$ do
      for task $M = 1$ to $N$ do
          if task_done(M) then
              continue to next task
          initialize $s \sim d_0$ from task $M$
          if Bernoulli($\epsilon$) then
              sample action $a$ uniformly
          else
              $\phi(s) = \{f_{IR}(z_i)\}_0^{|s|}$
              $a = \text{argmax}_i \ \psi(z_i, i)^\intercal \mathbf{w}$
          Store transition $(z, i, r(s, a), s')$ in $\mathcal{B}_M$
          Randomly sample minibatch from $\mathcal{B}_M$
          Update $\theta_\phi, \theta_{\mathbf{w}_M}$ via Equation 5.29
          Update $\theta_{\psi_M}$ via Equation 5.28
      return $\phi$, $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_N]$, $\Psi = [\psi_1, \ldots, \psi_N]$
```

Each episode is considered a single retrieval process for a query, and as we use a reranking process of the top n candidate documents retrieved by BM25, ϕ frequently iterates over all tasks. However, as opposed to RL where the training data is the same as the test data, a policy in IR must generalize to the unseen *test* data.

Thus, we modify the general multitask SF algorithm to handle the risk of overfitting. Using a validation set, the task is removed from rotation once it satisfies the early stopping criteria. At that point, $\mathbf{w}_M, \psi_M$ is saved. Once Algorithm 3 finishes, $\mathbf{W}, \Psi, \phi$ are tuned for 500 updates with their memory buffers merged. The best performance on the validation set during the tuning process is then selected for the basis function. While Tran et al. [123] introduce an approach to prevent overfitting on unbalanced collections, we leave improving the fine tuning process for future work.

5.3.2.2 General Policy Iteration

Incorporating recent work by Barreto et al. [10, 9], we are able to leverage the idea of the policy improvement theorem where a policy's value function is able to produce another policy that is at least no worse than the former's. In the case where we have multiple policies represented by $\mathbf{W}, \Psi$, GPI is able to produce a policy a new policy no worse than the set of previous policies by acting greedily.

Succinctly, GPI states if $M \in \mathcal{M}$ and let $Q_i^{\pi_j^*}$ be the action value function of an optimal policy of $M_j \in \mathcal{M}$ when executed in $M_i \in \mathcal{M}$. Given approximations $\{Q_i^{\pi_1}, Q_i^{\pi_2}, \dots, Q_i^{\pi^n}\}$ such that $||Q_i^{\pi_j^*} - Q_i^{\pi_j}||_\infty \leq \epsilon \; \forall j \in \{1, 2, \dots, n\}$, and let

$$\pi(s) \in \text{argmax}_a \max_j Q_i^{\pi_j}. \tag{5.31}$$

Then

$$||Q^* - Q^\pi||_\infty \leq \frac{2}{1-\gamma}(||r - r_i||_\infty + \min_j ||r_i - r_j||_\infty + \epsilon). \tag{5.32}$$

Where Q^* is an optimal action-value function on M and Q^π is the value function of π in M. Examining the GPI proposition demonstrates that it satisfies our task requirement. If $M_i = M_j$ then the right error term goes to 0 and we are left with the same error as found in M_i.. If we now consider $M_i \neq M_j$, the objective is to identify some $\mathbf{w}_i$ such that $\phi(s,a)^\intercal \mathbf{w_i} \approx r(s,a)$. Therefore, the first term on the right hand side captures the distance between M and any $\mathcal{M}_\phi$ [9].

We use this result as a foundation for URF, which is a modification of GPI&SF framework introduce by Barreto et al [9]. While RL often has to deal with potential long term rewards, Ψ plays an important role in identifying effective actions to take in a given state, retrieval is an entirely greedy option and $\phi^\intercal \mathbf{w}$ provides an effective ranking choice as it provides the estimated reward of ranking the current document highest. Therefore in Algorithm 4, we focus on $\mathbf{w}$ to reduce possible transfer issues across collections. As ϕ is fixed once finished training on the collections in $\mathcal{M}$, URF has two options to incorporate a new task. (1) It can *extend* its basis by learning a new $\mathbf{w}_{N+1}$ relevance function directly from ϕ. (2) Alternatively, if the basis tasks are sufficiently close to the new task, the new relevance function can be a linear combinations of previous N relevance functions,

$$\mathbf{w}_{N+1} = \sum_{i=0}^{|\mathbf{W}|} \beta_i \mathbf{w}_i, \tag{5.33}$$

such that $\sum_{i=0}^{|\mathbf{W}|} \beta_i = 1$. This can be viewed a modification of the GPI proposition, where the action value function becomes a composition and the worst case becomes 2ϵ, but has the benefit of creating "new" tasks by interpolation and reducing $||r - r_i||$ and $||r_i - r_j||$ by allowing r_i to lie in the convex hull of all $M \in \mathcal{M}_B$.

Algorithm 4: URF's Pseudo-GPI

```
Input: $\mathbf{W}, \Psi$, exploration probability $\epsilon \leq 1$, MDP: $(\mathcal{S}, \mathcal{A}, P, R, d_0, \gamma)$, extend
for episode $l = 1$ to $L$ do
    initialize $s \sim d_0$ from task $M$
    if Bernoulli($\epsilon$) then
        sample action $a$ uniformly
    else
        $\phi(s) = \max_i \{f_{IR}(z_i)\}_0^{|s|\intercal} \mathbf{w}$
        $a = i$
    r $\leftarrow$ take action $a$ in MDP
    if extend then
        $\mathbf{w} \leftarrow ||r(s,a) - \phi(s)^\intercal \mathbf{w}||_2^2]$
    else
        $\beta \leftarrow ||r(s,a) - \phi(s)^\intercal \sum_{i=0}^{|\mathbf{W}|} \beta_i \mathbf{w}_i||_2^2$
```

5.3.3 Experimental Setup

5.3.3.1 Collections

We evaluate URF on two collections from previous sections. Yahoo L4 has been used consistently across this thesis, and InsuranceQA in Section 4.1.2.1. In addition, we use the MSMARCO collection, which acts as a non sparse collection given the number of relevant documents.

L4 We use Yahoo's Webscope L4 high quality "Manner" collection [113]. For evaluation and training, all answers that were not the highest voted were removed from the collection to reduce label noise during training and provide a better judgment of performance during evaluation. Training, development, and test sets were created from a 80-10-10 split. Telescoping is used to create answer pools for evaluation from the top 10 BM25 retrieved answers as in [21].

InsuranceQA: In the InsuranceQA dataset, questions are created from real user submissions and the high quality answers come from insurance professionals. The dataset consists of 12,887 QA pairs for training, 1,000 pairs for validation, and two tests sets containing 1,800 pairs. For testing, each of the 1,800 QA pairs is evaluated with top 10 candidate answers from the 100 candidate version.

MSMARCO: This collection is based on Bing queries and their corresponding results. Originally proposed for a question answering task, the annotated data was used to create a passage re-ranking task. The collection consists of 400M training tuples of query, relevant, and non-relevant passages with the development set containing ~6,900 queries with each query corresponding to the top 1,000 passages retrieved via BM25 [89].

5.3.3.2 Architectures

As URF is intended to be used over any neural retrieval model in the same vein as ISS-TD, we use the same architectures from the Section 5.2.5.2: MatchPyramid (MP) [94], BLSTM [128, 92], and Transformer Kernel (TK) [46].

5.3.3.3 Evaluation

We evaluate the performance of the URF across three tasks which represent the three collections L4, MSMARCO, and InsuranceQA. For each collection, URF is initialized using the other two collections as its basis relevance models in $\mathbf{W}$, and then incorporates the last collection under Algorithm 4. We consider both the extension where another $\mathbf{w}$ is added to $\mathbf{W}$ that operates over ϕ as well as using only a linear combination of $\mathbf{W}$ to evaluate the third task. In addition, we benchmark the performance of URF on its basis set with respect to ISS-SF as to determine the amount of degradation that occurs when ϕ must capture a larger subset of $\mathcal{S}_{\mathcal{C}}$. As all collections have a single relevant document per query, we use MRR to evaluate the performance of URF.

5.3.3.4 Baselines

To the the best of our knowledge, there has been little work investigating multi collection performance for neural retrieval models. The adversarial regularization approach introduced in Section 4.1 requires extensive study of the regularized model as demonstrated by the Duet architecture in the discussion of negative results. Expanding this approach to cover MP and TK is non-trivial, and reasonable attempts were made with no success as the models diverged. Lastly, the domain regularization approach covers the zero shot learning problem, whereas URF specifically covers the situation of incorporating additional data into a trained model. Nogueira et al. [90] and Yilmaz et al. [4] have explored incorporating very large transformer architectures for retrieval over collections with minimal training data. However, these approaches are not comparable as they rely on the billions of parameters of a general language model (BERT, T5) trained independently of any retrieval task.

Therefore, we consider the reasonable baseline of fine tuning the upper layer of a supervised retrieval model. In this paradigm, the model is trained on alternating batches of the two collections in the same manner as URF, and then subsequently fine tuned on the target collection by incorporating an additional final layer.

5.3.3.5 Training and Hyperparameters

The learning rate and other hyperparameters were selected from the top performing hyperparameters in Section 5.2.5.3. The new $\mathbf{w}$ was trained with a learning rate of 10^{-3} and the Adam [57] optimizer. The tuning stage was done over 500 updates, and evaluated on the validation set every 50 updates. The tuned model with the best performance for both collections over the 500 updates was then selected as the basis for URF.

5.3.4 Results and Discussion

In this section, we discuss the overall performance of URF with and without extension, as well as an investigation into the behaviour of the algorithm under certain hyperparameters.

5.3.4.1 New Task Performance

	BLSTM		MP		TK	
source $\rightarrow$ *target*	Mono	Cross	Mono	Cross	Mono	Cross
Supervised						
(L4, InsuranceQA)$\rightarrow$ MSMARCO	0.5512	0.4118	0.4806	0.4199	0.6873	0.4682
(MSMARCO, InsuranceQA)$\rightarrow$ L4	0.5497	0.4094	0.4870	0.4746	0.5863	0.4903
(L4, MSMARCO)$\rightarrow$ InsuranceQA	0.6970	**0.6259**	0.6196	0.6340	0.7216	0.4319
URF - Restricted						
(L4, InsuranceQA)$\rightarrow$ MSMARCO	0.5314	0.3613	0.4884	0.4876	0.6526	0.5205
(MSMARCO, InsuranceQA)$\rightarrow$ L4	0.5569	0.4305	0.4835	0.4357	0.5353	0.5081
(L4, MSMARCO)$\rightarrow$ InsuranceQA	0.6931	0.4469	0.6213	0.6012	0.7306	0.6103
URF - Extended						
(L4, InsuranceQA)$\rightarrow$ MSMARCO	0.5314	**0.4339***	0.4884	**0.5040***	0.6526	**0.5496***
(MSMARCO, InsuranceQA)$\rightarrow$ L4	0.5569	**0.5223***	0.4835	**0.4855***	0.5353	**0.5318***
(L4, MSMARCO)$\rightarrow$ InsuranceQA	0.6931	0.6197	0.6213	**0.6476**	0.7306	**0.6193**

Table 5.8: Performance over new collections where the collections in parentheses represent the basis tasks. *Cross* and *Mono* compares the performance of incorporating the collection as a new task as opposed to being directly trained on it. Lastly. restricted and extended refer to extending the basis of URF. * indicates $p < .05$ with respect to baseline. Bold represents best performance for the transfer task.

URF - Supervised: Shown in Table 5.8, we observe the advantage of URF over supervised fine tuning, as URF is able to consistently improve over the baseline fine tuning approach despite having less tunable parameters (URF - Restricted). This is particularly true for the TK architecture across all collections, where URF improves scores from 5% - 37% when compared to supervised fine tuning. Furthermore, we observe URF-Expanded significantly outperforming supervised fine tuning for almost every permutation of basis set and architecture choice demonstrating the strength of this approach. Furthermore, as URF-Restricted is not shown ϕ directly, the improved performance of URF-Restricted when compared to supervised fine tuning reflects the advantage of creating a new relevance function from within the convex hull of $\mathbf{W}$.

Extended vs Restricted: In the same Table 5.8, examining these two regimes under *URF - Restricted* and *URF - Extended*, we are able to observe the similarity in relevance signals across collections. While significantly outperforming the supervised fine tuning approach, there are certain basis configurations that can approximate a retrieval function learned over the significantly richer representation ϕ. In particular, URF-Restricted TK on InsuranceQA can approximate URF-Extended which has access to ϕ. Lastly, there is an architecture sensitive response under URF. While TK has similar performance on both versions of URF, BLSTM based URF undergoes a dramatic improvement between operating in the convex hull and learning a new relevance function. One possible cause of this is that $\mathbf{W}$ in the BLSTM architecture is not leveraging all of ϕ's output, which becomes a critical issue as the neural model operates over a new text distribution.

5.3.4.2 URF - Single Task Comparison:

Model	**(L4, MSMARCO)**	**(MSMARCO, InsuranceQA)**	**(L4, InsuranceQA)**
BLSTM	(-1.9%, -3.9%)	(+1.7%, 0.0%)	(-8.2%, -0.6%)
MP	(-1.3%, -1.5%)	(0.0%, -0.3%)	(-7.7%, -1.0%)
TK	(+0.1%, -24.3%)	(+5.5%, +3.8%)	(-8.8%, -5.7%)

Table 5.9: MRR Performance of URF on basis set after tuning.

While we have demonstrated the feasibility of URF covering new datasets, we now address the question of whether Algorithm 3 preserves basis task performance. As shown in Table 5.9, the final basis relevance models used for URF suffer a collection dependent degradation, with the exception of (MSMARCO, InsuranceQA) where these two collections have a synergistic effect. However, this discrepancy is the effect of the imbalance between collections and the tuning process as we observe less than 0.5% degradation during training the basis sets. Specifically, a model will achieve parity with its single task performance prior to early stopping, and if two collections differ in the required time to converge to an effective model, ϕ will have significantly changed by the tuning process to the point where it is no longer effective for one of the basis collections and is not recoverable via simple tuning. We then observe the 8-24% degradation if this occurs. This phenomena is also observed in Guo et al. [41], where a general IR model over multiple collections is not as effective as an in-domain trained model.

5.3.4.3 Hyperparameter Study

Depth-Width	**L4**	**MSMARCO**
3-700	0.5587	0.3943
3-400	0.5548	0.4371
2-700	0.5508	0.3743
2-400	0.5203	0.3675
1-400	0.4824	0.3441

Table 5.10: MRR Performance of URF on basis set as a function of number of layers (depth) and width of BLSTM model. Both L4 and MSMARCO are part of the basis set.

With respect to incorporating a new task, we observed a high propensity towards basis models within the first 100 tuning steps, to the point where there is poor generalization occurring towards the end of the tuning stage, despite consistent on task generalization performance. This result is unsurprising as one can view both training and validation datasets for the on task performance as part of the training distribution. While the IR model might not overfit within L4 or MSMARCO for example, its transformations are becoming more refined on that data leading to poor out of distribution performance, even for relevance separated ϕ.

Model Capacity: Shown in Table 5.10, the effectiveness of ϕ is dependent on the size of the model with respect to $|\mathcal{M}_\phi|$. On the single collection retrieval task, a single layer BLSTM is able to achieve comparable performance with multiple layers. However, when ϕ must model a larger state space, the same model is not able to fully capture the needed high level features for ψ and $\mathbf{w}$ to determine relevance. This is demonstrated as the single layer model performs significantly worse on both collections, with a 14% and 22% decay in performance for L4 and MSMARCO respectively. As discussed in Montufar et al. [86], narrower and deep models are able to recycle mappings as the model can use the depth to map the input space to a few common points in the upper layers. This provides a potential reason as to why the 3-700 BLSTM model failed to generalize to MSMARCO despite the stability afforded to BLSTM based architectures. An interesting note is that MP and TK architectures do not suffer this degradation. This is possibly due to the deeper architecture handling multiple collections or the models were constructed with enough parameters.

CHAPTER 6
CONCLUSION

In this work, we introduce approaches in three core areas to overcome the lack of training data that is needed to train neural models for the answer passage retrieval task: text representation, domain transfer, and robust temporal difference methods for URF. In Chapter 3, we introduce text representation approaches for answer passage retrieval which leverage local embeddings to reduce the training demand on the downstream neural model. Expanding on this work to cover text not present in the training data, we incorporate a hybrid architecture to dynamically construct out of vocabulary terms and cover unique collection specific term uses. Both of these approaches demonstrate increasing effectiveness as the collections consistently become more sparse or noisy.

In Chapter 4, we leverage data from similar domains to further increase performance over portions of the collection when there is no training data available. Designing an adversarial system where the reverse gradient specifically ablates domain information, we produce a neural answer passage retrieval model that is able to retrieve over a new domain without any training data, achieving a 3%-12% increase in performance. Expanding on the definition of domain, we examine the situation where the sparse labels result in a neural model only seeing a portion of the collection during training, we introduce a policy based negative sampling strategy to better cover a collection during training. We observe that while this approach does not result in improved peak performance, the policy is able to ensure more consistent performance across starting parameter distributions.

In Chapter 5, we demonstrate there exists a significant overlap of information within the layers of a neural model for different QA collections. Acting on this information, we extend reinforcement learning to sparse answer passage collections by ISS-TD which enables the use of successor features [10] and substantially improves performance on sparse collections when compared to alternative RL approaches. Lastly, we introduce URF, a method to incorporate relevance models in an incremental fashion acting as the first instance of gracefully incorporating new collections into an IR model. Under this regime, we are able to reach parity with on task performance under certain settings, which provides a promising direction of future study. While not discussed in this thesis, one can easily expand URF to address specific demographic or diversity constraints on a relevance function by updating the relevance vector $\mathbf{w}$ directly through the ISS-TD updates.

6.1 Future Work

Despite the contributions made in this thesis, there remains a substantial amount of work to be done in this direction. As we have seen with BERT [25], the potential to incorporate effective information from non-traditional resources to improve performance on a new collection warrants future study. In the case of IR and our work, this direction would address a more detailed policy based sampling approach able to select individual documents, improve the stability of adversarial retrieval, and incorporate lifelong learning into URF.

Larger action space for policy based sample selection: As AC-IR demonstrated capable control over a BM25-random action space, an open research direction is individual document selection to allow a neural model to make the most out of a limited query set. Tang and Hui [119] suggest a way to view an entire document collection to allow the agent to move away from function based actions. Alternatively, one can incorporate a diversification function over various retrieval functions to enable greater coverage of a collection [127].

Adversarial domain adaptation with improved convergence: While effective when converged, the adversarial framework is sensitive to learning rate, and can potentially ablate vital information which causes the IR model to diverge. A possible cause of this is due the pathological curvature found in neural network optimization [155]. While Adam [57] is effective while still being a first order optimization method, the combination of two gradient updates in opposing directions under a first order method might lead to this observed instability. A natural gradient based approach, while second order, would provide insight into improving the behaviour of the adversarial regularization.

Expanding ϕ gracefully in URF: Although the URF framework is able to outperform supervised fine-tuning, it suffers from the information bound in ϕ. Future work would include incorporating additional collections with the hypothesis that as $\mathcal{M}_B$ grows, ϕ is able to adequately perform on new collections. However, an alternative approach could incorporate lifelong learning for ϕ while maintaining a linear relation to previous relevance functions. This feature would allow for a flexible starting basis such that an IR system would be able to iteratively incorporate new collections, with less and less samples needed for each additional collection. One possible way is to leverage work in linear programming, delayed column generation [8], where we can adaptively expand the dimension of ϕ and $\mathbf{w}$ to incorporate new collections while maintaining the constraints that $\phi^\intercal \mathbf{w}$ maintains performance on the basis set. This significantly reduces the complexity of the task as increasing the dimension of ϕ is significantly less challenging than requiring the output of a non-linear function remain consistent when updating its parameters.

BIBLIOGRAPHY

[1] Adhikari, Ashutosh, Ram, Achyudh, Tang, Raphael, Hamilton, William L., and Lin, Jimmy. Exploring the limits of simple learners in knowledge distillation for document classification with docbert. In *Proceedings of the 5th Workshop on Representation Learning for NLP, RepL4NLP@ACL 2020, Online, July 9, 2020* (2020), Spandana Gella, Johannes Welbl, Marek Rei, Fabio Petroni, Patrick S. H. Lewis, Emma Strubell, Min Joon Seo, and Hannaneh Hajishirzi, Eds., Association for Computational Linguistics, pp. 72–77.

[2] Adi, Yossi, Zeghidour, Neil, Collobert, Ronan, Usunier, Nicolas, Liptchinsky, Vitaliy, and Synnaeve, Gabriel. To reverse the gradient or not: an empirical comparison of adversarial and multi-task learning in speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019* (2019), IEEE, pp. 3742–3746.

[3] Ahmad, Wasi Uddin, Chang, Kai-Wei, and Wang, Hongning. Multi-task learning for document ranking and query suggestion. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings* (2018), OpenReview.net.

[4] Akkalyoncu Yilmaz, Zeynep, Yang, Wei, Zhang, Haotian, and Lin, Jimmy. Cross-domain modeling of sentence-level evidence for document retrieval. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (Hong Kong, China, Nov. 2019), Association for Computational Linguistics, pp. 3490–3496.

[5] Al-Hawamdeh, S., and Willett, P. Paragraph-based searching in full-text documents. *Electron. Publ. Origin. Dissem. Des. 2*, 4 (Nov. 1989), 179–192.

[6] Alain, Guillaume, and Bengio, Yoshua. Understanding intermediate layers using linear classifier probes. *CoRR abs/1610.01644* (2016).

[7] Arora, Sanjeev, Li, Yuanzhi, Liang, Yingyu, Ma, Tengyu, and Risteski, Andrej. Random walks on context spaces: Towards an explanation of the mysteries of semantic word embeddings. *CoRR abs/1502.03520* (2015).

[8] Barnhart, Cynthia, Johnson, Ellis L., Nemhauser, George L., Savelsbergh, Martin W. P., and Vance, Pamela H. Branch-and-price: Column generation for solving huge integer programs. *Operations Research 46*, 3 (1998), 316–329.

[9] Barreto, Andre, Borsa, Diana, Quan, John, Schaul, Tom, Silver, David, Hessel, Matteo, Mankowitz, Daniel, Zidek, Augustin, and Munos, Remi. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *Proceedings of the 35th International Conference on Machine Learning* (Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018), Jennifer Dy and Andreas Krause, Eds., vol. 80 of *Proceedings of Machine Learning Research*, PMLR, pp. 501–510.

[10] Barreto, André, Dabney, Will, Munos, Rémi, Hunt, Jonathan J., Schaul, Tom, Silver, David, and van Hasselt, Hado. Successor features for transfer in reinforcement learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA* (2017), pp. 4055–4065.

[11] Birattari, Mauro, and Dorigo, Marco. How to assess and report the performance of a stochastic algorithm on a benchmark problem: mean or best result on a number of runs? *Optimization letters 1*, 3 (2007), 309–311.

[12] Bjerva, Johannes, Plank, Barbara, and Bos, Johan. Semantic tagging with deep residual networks. In *COLING 2016, Technical Papers, December 11-16, 2016, Osaka, Japan* (2016), pp. 3531–3541.

[13] Bojarski, Mariusz, Testa, Davide Del, Dworakowski, Daniel, Firner, Bernhard, Flepp, Beat, Goyal, Prasoon, Jackel, Lawrence D., Monfort, Mathew, Muller, Urs, Zhang, Jiakai, Zhang, Xin, Zhao, Jake, and Zieba, Karol. End to end learning for self-driving cars. *CoRR abs/1604.07316* (2016).

[14] Bowman, Samuel R., Angeli, Gabor, Potts, Christopher, and Manning, Christopher D. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326* (2015).

[15] Burges, Chris J.C. From ranknet to lambdarank to lambdamart: An overview. Tech. Rep. MSR-TR-2010-82, June 2010.

[16] Calauzènes, Clément, Usunier, Nicolas, and Gallinari, Patrick. On the (non)-existence of convex, calibrated surrogate losses for ranking. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 197–205.

[17] Callan, James P. Passage-level evidence in document retrieval. In *ACM SIGIR* (New York, NY, USA, 1994), SIGIR '94, Springer-Verlag New York, Inc., pp. 302–310.

[18] Cao, Yunbo, Xu, Jun, Liu, Tie-Yan, Li, Hang, Huang, Yalou, and Hon, Hsiao-Wuen. Adapting ranking svm to document retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval* (2006), ACM, pp. 186–193.

[19] Choshen, Leshem, Fox, Lior, Aizenbud, Zohar, and Abend, Omri. On the weaknesses of reinforcement learning for neural machine translation. In *International Conference on Learning Representations* (2020).

[20] Cohen, Daniel, Ai, Qingyao, and Croft, W. Bruce. Adaptability of Neural Networks on Varying Granularity IR Tasks. In *SIGIR Neu-IR Workshop* (Pisa, Italy, 2016).

[21] Cohen, Daniel, and Croft, W. Bruce. End to End Long Short Term Memory Networks for Non-Factoid Question Answering. In *ICTIR* (Newark, DE, USA, 2016).

[22] Cover, Thomas M., and Thomas, Joy A. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, 2006.

[23] Dayan, Peter. Improving generalization for temporal difference learning: The successor representation. *Neural Computation 5*, 4 (1993), 613–624.

[24] Dehghani, Mostafa, Zamani, Hamed, Severyn, Aliaksei, Kamps, Jaap, and Croft, W. Bruce. Neural ranking models with weak supervision. In *SIGIR* (2017), ACM.

[25] Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, and Toutanova, Kristina. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)* (2019), pp. 4171–4186.

[26] Diaz, Fernando, Mitra, Bhaskar, and Craswell, Nick. Query expansion with locally-trained word embeddings. In *ACL* (2016), ACL, pp. 367–377.

[27] Donahue, Jeff, Krähenbühl, Philipp, and Darrell, Trevor. Adversarial feature learning. *CoRR abs/1605.09782* (2016).

[28] dos Santos, Cícero Nogueira, Tan, Ming, Xiang, Bing, and Zhou, Bowen. Attentive pooling networks. *CoRR abs/1602.03609* (2016).

[29] Frankle, Jonathan, Dziugaite, Gintare Karolina, Roy, Daniel M., and Carbin, Michael. The lottery ticket hypothesis at scale. *CoRR abs/1903.01611* (2019).

[30] Furuta, Ryosuke, Inoue, Naoto, and Yamasaki, Toshihiko. Pixelrl: Fully convolutional network with reinforcement learning for image processing. *IEEE Trans. Multimedia 22*, 7 (2020), 1704–1719.

[31] Ganin, Yaroslav, Ustinova, Evgeniya, Ajakan, Hana, Germain, Pascal, Larochelle, Hugo, Laviolette, François, Marchand, Mario, and Lempitsky, Victor. Domain-adversarial training of neural networks. *J. Mach. Learn. Res. 17*, 1 (2016), 2096–2030.

[32] Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. In *NIPS 2014* (2014), Curran Associates, Inc., pp. 2672–2680.

[33] Goodman, Joshua. Classes for fast maximum entropy training. In *ICASSP* (2001).

[34] Gordon, Geoffrey J. Stable function approximation in dynamic programming. In *Machine Learning Proceedings 1995*, Armand Prieditis and Stuart Russell, Eds. Morgan Kaufmann, San Francisco (CA), 1995, pp. 261 – 268.

[35] Gordon, Geoffrey J. Reinforcement learning with function approximation converges to a region. In *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA* (2000), Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, Eds., MIT Press, pp. 1040–1046.

[36] Graves, A., Jaitly, N., and Mohamed, A.-R. Hybrid speech recognition with deep bidirectional lstm. In *ASRU, 2013* (Dec. 2013), pp. 273–278.

[37] Graves, Alex, Bellemare, Marc G., Menick, Jacob, Munos, Rémi, and Kavukcuoglu, Koray. Automated curriculum learning for neural networks. In *ICML* (2017), Proceedings of Machine Learning Research, PMLR.

[38] Guan, Dongyi, Zhang, Sicong, and Yang, Hui. Utilizing query change for session search. In *SIGIR* (2013), pp. 453–462.

[39] Guo, Jiafeng, Fan, Yixing, Ai, Qingyao, and Croft, W. Bruce. A deep relevance matching model for ad-hoc retrieval. In *CIKM '16* (New York, NY, USA, 2016), ACM, pp. 55–64.

[40] Guo, Jiafeng, Fan, Yixing, Pang, Liang, Yang, Liu, Ai, Qingyao, Zamani, Hamed, Wu, Chen, Croft, W. Bruce, and Cheng, Xueqi. A deep look into neural ranking models for information retrieval. *CoRR abs/1903.06902* (2019).

[41] Guo, Mandy, Yang, Yinfei, Cer, Daniel, Shen, Qinlan, and Constant, Noah. Multireqa: A cross-domain evaluation for retrieval question answering models, 2020.

[42] Gysel, Christophe Van, de Rijke, Maarten, and Kanoulas, Evangelos. Neural vector spaces for unsupervised information retrieval. *ACM Trans. Inf. Syst. 36*, 4 (June 2018).

[43] Hassoun, Mohamad H. *Fundamentals of Artificial Neural Networks*, 1st ed. MIT Press, Cambridge, MA, USA, 1995.

[44] Hessel, Matteo, Modayil, Joseph, van Hasselt, Hado, Schaul, Tom, Ostrovski, Georg, Dabney, Will, Horgan, Dan, Piot, Bilal, Azar, Mohammad Gheshlaghi, and Silver, David. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018* (2018), Sheila A. McIlraith and Kilian Q. Weinberger, Eds., AAAI Press, pp. 3215–3222.

[45] Hoffman, Judy, Tzeng, Eric, Park, Taesung, Zhu, Jun-Yan, Isola, Phillip, Saenko, Kate, Efros, Alexei A., and Darrell, Trevor. Cycada: Cycle-consistent adversarial domain adaptation. *CoRR abs/1711.03213* (2017).

[46] Hofstätter, Sebastian, Zlabinger, Markus, and Hanbury, Allan. Interpretable & Time-Budget-Constrained Contextualization for Re-Ranking. In *Proc. of ECAI* (2020).

[47] Hornik, Kurt. Approximation capabilities of multilayer feedforward networks. *Neural Netw. 4*, 2 (Mar. 1991), 251–257.

[48] Hu, Yujing, Da, Qing, Zeng, Anxiang, Yu, Yang, and Xu, Yinghui. Reinforcement Learning to Rank in E-Commerce Search Engine: Formalization, Analysis, and Application. *arXiv:1803.00710 [cs]* (May 2018). arXiv: 1803.00710.

[49] Huang, Po-Sen, He, Xiaodong, Gao, Jianfeng, Deng, Li, Acero, Alex, and Heck, Larry. Learning deep structured semantic models for web search using clickthrough data. In *CIKM '13* (2013), ACM, pp. 2333–2338.

[50] Iyyer, Mohit, Boyd-Graber, Jordan, Claudino, Leonardo, Socher, Richard, and Daumé III, Hal. A neural network for factoid question answering over paragraphs. In *EMNLP* (2014).

[51] Janz, David, Hron, Jiri, Hernández-Lobato, José Miguel, Hofmann, Katja, and Tschiatschek, Sebastian. Successor Uncertainties: Exploration and Uncertainty in Temporal Difference Learning. *arXiv:1810.06530 [cs, stat]* (Oct. 2018). arXiv: 1810.06530.

[52] Jones, Karen Sparck, Walker, Steve, and Robertson, Stephen E. A probabilistic model of information retrieval: development and comparative experiments - part 1. *Inf. Process. Manage. 36*, 6 (2000), 779–808.

[53] Jordan, Scott, Cohen, Daniel, and Thomas, Philip. Using cumulative distribution based performance analysis to benchmark models. In *NeurIPS Workshop* (2018).

[54] Karpathy, Andrej, Johnson, Justin, and Li, Fei-Fei. Visualizing and understanding recurrent networks. *CoRR abs/1506.02078* (2015).

[55] Keikha, Mostafa, Park, Jae Hyun, Croft, W. Bruce, and Sanderson, Mark. Retrieving passages and finding answers. In *ADCS '14* (New York, NY, USA, 2014), ACM, pp. 81:81–81:84.

[56] Khalid, Mahboob Alam, and Verberne, Suzan. Passage retrieval for question answering using sliding windows. In *Coling 2008: Proceedings of the 2Nd Workshop on Information Retrieval for Question Answering* (Stroudsburg, PA, USA, 2008), IRQA '08, Association for Computational Linguistics, pp. 26–33.

[57] Kingma, Diederik P., and Ba, Jimmy. Adam: A method for stochastic optimization. *CoRR abs/1412.6980* (2014).

[58] Konda, Vijay R., and Tsitsiklis, John N. Actor-critic algorithms. In *Advances in Neural Information Processing Systems 12*, S. A. Solla, T. K. Leen, and K. Müller, Eds. MIT Press, 2000, pp. 1008–1014.

[59] Krizhevsky, Alex. Learning multiple layers of features from tiny images. Tech. rep., 2009.

[60] Lample, Guillaume, Ballesteros, Miguel, Subramanian, Sandeep, Kawakami, Kazuya, and Dyer, Chris. Neural architectures for named entity recognition. In *HLT-NAACL* (2016).

[61] Le, Quoc V., and Mikolov, Tomas. Distributed representations of sentences and documents. *CoRR abs/1405.4053* (2014).

[62] Levy, Omer, Goldberg, Yoav, and Dagan, Ido. Improving distributional similarity with lessons learned from word embeddings. *TACL 3* (2015), 211–225.

[63] Li, Jiwei, Chen, Xinlei, Hovy, Eduard H., and Jurafsky, Dan. Visualizing and understanding neural models in NLP. In *NAACL HLT 2016, San Diego California, USA, June 12-17, 2016* (2016), pp. 681–691.

[64] Li, Jiwei, and Jurafsky, Dan. Do multi-sense embeddings improve natural language understanding? In *EMNLP 2015, Lisbon, Portugal, September 17-21, 2015* (2015), pp. 1722–1732.

[65] Li, Piji, Bing, Lidong, and Lam, Wai. Actor-critic based training framework for abstractive summarization. *CoRR abs/1803.11070* (2018).

[66] Li, Xin, and Roth, Dan. Learning question classifiers. In *COLING - Volume 1* (Stroudsburg, PA, USA, 2002), COLING '02, ACL, pp. 1–7.

[67] Lillicrap, Timothy P., Hunt, Jonathan J., Pritzel, Alexander, Heess, Nicolas, Erez, Tom, Tassa, Yuval, Silver, David, and Wierstra, Daan. Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings* (2016), Yoshua Bengio and Yann LeCun, Eds.

[68] Liu, Ruishan, and Zou, James. The Effects of Memory Replay in Reinforcement Learning. *arXiv:1710.06574 [cs, stat]* (Oct. 2017). arXiv: 1710.06574.

[69] Liu, Xiaodong, Gao, Jianfeng, He, Xiaodong, Deng, Li, Duh, Kevin, and Wang, Ye-yi. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (Denver, Colorado, May–June 2015), Association for Computational Linguistics, pp. 912–921.

[70] Liu, Xiaoyong, and Croft, W. Bruce. Passage retrieval based on language models. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management* (New York, NY, USA, 2002), CIKM '02, ACM, pp. 375–382.

[71] Long, Mingsheng, and Wang, Jianmin. Learning multiple tasks with deep relationship networks. *CoRR abs/1506.02117* (2015).

[72] Luong, Thang, Pham, Hieu, and Manning, Christopher D. Effective approaches to attention-based neural machine translation. In *EMNLP* (2015), The Association for Computational Linguistics, pp. 1412–1421.

[73] Ma, Xiaofei, Xu, Peng, Wang, Zhiguo, Nallapati, Ramesh, and Xiang, Bing. Universal text representation from bert: An empirical study, 2019.

[74] Ma, Xuezhe, and Hovy, Eduard H. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *ACL 2016, August 7-12, 2016, Berlin, Germany* (2016).

[75] Maas, Andrew L., Daly, Raymond E., Pham, Peter T., Huang, Dan, Ng, Andrew Y., and Potts, Christopher. Learning word vectors for sentiment analysis. In *ACL* (Portland, Oregon, USA, June 2011), Association for Computational Linguistics, pp. 142–150.

[76] Marcus, Mitchell, Kim, Grace, Marcinkiewicz, Mary Ann, MacIntyre, Robert, Bies, Ann, Ferguson, Mark, Katz, Karen, and Schasberger, Britta. The penn treebank: Annotating predicate argument structure. In *HLT* (Stroudsburg, PA, USA, 1994), Association for Computational Linguistics, pp. 114–119.

[77] Meng, Lingxun, Li, Yan, Liu, Mengyi, and Shu, Peng. Skipping word: A character-sequential representation based framework for question answering. In *CIKM '16* (2016), pp. 1869–1872.

[78] Metzler, Donald, and Bruce Croft, W. Linear feature-based models for information retrieval. *Inf. Retr. 10*, 3 (June 2007), 257–274.

[79] Miao, Yishu, Yu, Lei, and Blunsom, Phil. Neural variational inference for text processing. In *ICML* (2016), JMLR.org, pp. 1727–1736.

[80] Mikolov, Tomas, Chen, Kai, Corrado, Greg, and Dean, Jeffrey. Efficient estimation of word representations in vector space. *CoRR abs/1301.3781* (2013).

[81] Mikolov, Tomáš, Sutskever, Ilya, Deoras, Anoop, Hai Son, Le, Kombrink, Stefan, and Cernock, Jaň. Subword language modeling with neural networks. *preprint* (2010).

[82] Miller, John, and Hardt, Moritz. Stable recurrent models. In *International Conference on Learning Representations* (2019).

[83] Mitra, Bhaskar, and Craswell, Nick. An introduction to neural information retrieval. *Foundations and Trends in Information Retrieval* (2018).

[84] Mitra, Bhaskar, Diaz, Fernando, and Craswell, Nick. Learning to match using local and distributed representations of text for web search. In *WWW 17* (2017), pp. 1291–1299.

[85] Montazeralghaem, Ali, Zamani, Hamed, and Allan, James. A reinforcement learning framework for relevance feedback. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2020), SIGIR '20, Association for Computing Machinery, p. 59–68.

[86] Montufar, Guido F, Pascanu, Razvan, Cho, Kyunghyun, and Bengio, Yoshua. On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2924–2932.

[87] Nanni, Federico, Mitra, Bhaskar, Magnusson, Matt, and Dietz, Laura. Benchmark for complex answer retrieval. In *Proc. ICTIR* (2017), ACM, pp. 293–296.

[88] Ng, Andrew Y., Harada, Daishi, and Russell, Stuart J. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of ICML* (1999).

[89] Nguyen, Tri, Rosenberg, Mir, Song, Xia, Gao, Jianfeng, Tiwary, Saurabh, Majumder, Rangan, and Deng, Li. MS MARCO: A human generated machine reading comprehension dataset. *CoRR abs/1611.09268* (2016).

[90] Nogueira, Rodrigo, Jiang, Zhiying, and Lin, Jimmy. Document ranking with a pretrained sequence-to-sequence model, 2020.

[91] O'Connor, John. Text searching retrieval of answer-sentences and other answer-passages. *Journal of the American Society for Information Science 24*, 6 (1973), 445–460.

[92] Palangi, Hamid, Deng, Li, Shen, Yelong, Gao, Jianfeng, He, Xiaodong, Chen, Jianshu, Song, Xinying, and Ward, Rabab K. Deep sentence embedding using the long short term memory network: Analysis and application to information retrieval. *CoRR abs/1502.06922* (2015).

[93] Palangi, Hamid, Deng, Li, Shen, Yelong, Gao, Jianfeng, He, Xiaodong, Chen, Jianshu, Song, Xinying, and Ward, Rabab K. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Trans. Audio, Speech & Language Processing 24*, 4 (2016), 694–707.

[94] Pang, Liang, Lan, Yanyan, Guo, Jiafeng, Xu, Jun, Wan, Shengxian, and Cheng, Xueqi. Text matching as image recognition. In *Proceedings of AAAI* (2016), AAAI Press.

[95] Paulus, Romain, Xiong, Caiming, and Socher, Richard. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations* (2018).

[96] Pennington, Jeffrey, Socher, Richard, and Manning, Christopher D. Glove: Global vectors for word representation. In *EMNLP* (2014), pp. 1532–1543.

[97] Ponte, Jay M., and Croft, W. Bruce. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 1998), SIGIR '98, ACM, pp. 275–281.

[98] Popel, Martin, and Bojar, Ondřej. Training Tips for the Transformer Model. *The Prague Bulletin of Mathematical Linguistics 110*, 1 (Apr. 2018), 43–70.

[99] Precup, Doina, Sutton, Richard S., and Singh, Satinder P. Eligibility traces for off-policy policy evaluation. In *Proceedings of ICML* (2000), Morgan Kaufmann.

[100] Qin, Tao, and Liu, Tie-Yan. Introducing letor 4.0 datasets, 2013.

[101] Rajpurkar, Pranav, Zhang, Jian, Lopyrev, Konstantin, and Liang, Percy. Squad: 100, 000+ questions for machine comprehension of text. *CoRR abs/1606.05250* (2016).

[102] Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review* (1958), 65–386.

[103] Roy, Nicholas, and Mccallum, Andrew. Toward optimal active learning through monte carlo estimation of error reduction. In *ICML* (2001).

[104] Rozantsev, Artem, Salzmann, Mathieu, and Fua, Pascal. Beyond sharing weights for deep domain adaptation. *CoRR abs/1603.06432* (2016).

[105] Sang, Erik F. Tjong Kim, and Meulder, Fien De. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003* (2003), pp. 142–147.

[106] Schulman, John, Wolski, Filip, Dhariwal, Prafulla, Radford, Alec, and Klimov, Oleg. Proximal policy optimization algorithms. *CoRR abs/1707.06347* (2017).

[107] Seo, Min Joon, Kembhavi, Aniruddha, Farhadi, Ali, and Hajishirzi, Hannaneh. Bidirectional attention flow for machine comprehension. In *ICLR* (Toulon, France, 2017).

[108] Severyn, Aliaksei, and Moschitti, Alessandro. Learning to rank short text pairs with convolutional deep neural networks. In *SIGIR* (New York, NY, USA, 2015), SIGIR '15, ACM, pp. 373–382.

[109] Shen, Yelong, Huang, Po-Sen, Gao, Jianfeng, and Chen, Weizhu. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2017), KDD '17, Association for Computing Machinery, p. 1047–1055.

[110] Singhal, Amit. Modern information retrieval: a brief overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering 24* (2001), 2001.

[111] Søgaard, Anders, and Goldberg, Yoav. Deep multi-task learning with low level tasks supervised at lower layers. In *ACL (2)* (2016), The Association for Computer Linguistics.

[112] Song, Fei, and Croft, W. Bruce. A general language model for information retrieval. In *Proceedings of the Eighth International Conference on Information and Knowledge Management* (New York, NY, USA, 1999), CIKM '99, ACM, pp. 316–321.

[113] Surdeanu, Mihai, Ciaramita, Massimiliano, and Zaragoza, Hugo. Learning to rank answers on large online qa collections. In *ACL:HLT* (2008), pp. 719–727.

[114] Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to sequence learning with neural networks. *CoRR abs/1409.3215* (2014).

[115] Sutton, Richard, and Barto, Andrew. *Reinforcement Learning.* MIT, 2016.

[116] Sutton, Richard S, McAllester, David A., Singh, Satinder P., and Mansour, Yishay. Policy gradient methods for reinforcement learning with function approximation. In *Advances in NIPS.* MIT Press, 2000.

[117] Szepesvári, Csaba, and Smart, William D. Interpolation-based q-learning. In *ICML '04* (Banff, Alberta, Canada, 2004), ACM Press, p. 100.

[118] Tan, Ming, Xiang, Bing, and Zhou, Bowen. Lstm-based deep learning models for non-factoid answer selection. *CoRR abs/1511.04108* (2015).

[119] Tang, Zhiwen, and Yang, Grace Hui. Corpus compression for deep reinforcement learning in natural language environments. In *The 41st International ACM SIGIR Conference on Research Development in Information Retrieval - DRL4IR Workshop* (2020), SIGIR '20.

[120] Tielman, T, and Hinton, George. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[121] Tishby, Naftali, and Zaslavsky, Noga. Deep learning and the information bottleneck principle. *CoRR abs/1503.02406* (2015).

[122] Tong, Simon, and Koller, Daphne. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res. 2* (Mar. 2002), 45–66.

[123] Tran, Brandon, Karimzadehgan, Maryam, Pasumarthi, Rama Kumar, Bendersky, Michael, and Metzler, Donald. Domain adaptation for enterprise email search. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2019), SIGIR'19, Association for Computing Machinery, p. 25–34.

[124] Tzeng, Eric, Hoffman, Judy, Saenko, Kate, and Darrell, Trevor. Adversarial discriminative domain adaptation. In *CVPR 17* (2017), vol. 1, p. 4.

[125] Tzeng, Eric, Hoffman, Judy, Zhang, Ning, Saenko, Kate, and Darrell, Trevor. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474* (2014).

[126] Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N., Kaiser, Lukasz, and Polosukhin, Illia. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA* (2017), pp. 5998–6008.

[127] Vikraman, Lakshmi, Croft, W. Bruce, and OConnor, Brendan. Exploring diversification in non-factoid question answering. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval* (2018), pp. 223–226.

[128] Wang, Di, and Nyberg, Eric. A recurrent neural network based answer ranking model for web question answering. In *WebQA Workshop, SIGIR '15, Santiago, Chile.*

[129] Wang, Di, and Nyberg, Eric. A long short-term memory model for answer sentence selection in question answering. In *ACL-IJCNLP, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers* (2015), pp. 707–712.

[130] Wang, Jun, Yu, Lantao, Zhang, Weinan, Gong, Yu, Xu, Yinghui, Wang, Benyou, Zhang, Peng, and Zhang, Dell. IRGAN: A minimax game for unifying generative and discriminative information retrieval models. In *SIGIR* (2017), ACM, pp. 515–524.

[131] Wang, Mengqiu, Smith, Noah A., and Mitamura, Teruko. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP '07*.

[132] Wang, Yu, and Jin, Hongxia. A Deep Reinforcement Learning Based Multi-Step Coarse to Fine Question Answering (MSCQA) System. *Proceedings of the AAAI Conference on Artificial Intelligence 33*, 01 (July 2019), 7224–7232. Number: 01.

[133] Wang, Yu-Xiong, Ramanan, Deva, and Hebert, Martial. Learning to model the tail. In *Advances in NIPS*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 7029–7039.

[134] Watkins, Christopher J. C. H., and Dayan, Peter. Q-learning. In *Machine Learning* (1992), pp. 279–292.

[135] Wei, Kai, Iyer, Rishabh, and Bilmes, Jeff. Submodularity in data subset selection and active learning. In *ICML* (Lille, France, 2015).

[136] Wei, Zeng, Xu, Jun, Lan, Yanyan, Guo, Jiafeng, and Cheng, Xueqi. Reinforcement learning to rank with markov decision process. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2017), SIGIR '17, Association for Computing Machinery, p. 945–948.

[137] Williams, Ronald J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn. 8*, 3-4 (May 1992), 229–256.

[138] Wu, Lijun, Tian, Fei, Qin, Tao, Lai, Jianhuang, and Liu, Tie-Yan. A study of reinforcement learning for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (Brussels, Belgium, Oct.-Nov. 2018), Association for Computational Linguistics, pp. 3612–3621.

[139] Wu, Lijun, Tian, Fei, Qin, Tao, Lai, Jianhuang, and Liu, Tie-Yan. A study of reinforcement learning for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (Brussels, Belgium, Oct. 2018), Association for Computational Linguistics, pp. 3612–3621.

[140] Xia, Long, Xu, Jun, Lan, Yanyan, Guo, Jiafeng, Zeng, Wei, and Cheng, Xueqi. Adapting Markov Decision Process for Search Result Diversification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '17* (Shinjuku, Tokyo, Japan, 2017), ACM Press, pp. 535–544.

[141] Xing, X., and Chang, D. E. Deep reinforcement learning based robot arm manipulation with efficient training data through simulation. In *2019 19th International Conference on Control, Automation and Systems (ICCAS)* (2019), pp. 112–116.

[142] Xu, Jun, Wei, Zeng, Xia, Long, Lan, Yanyan, Yin, Dawei, Cheng, Xueqi, and Wen, Ji-Rong. Reinforcement Learning to Rank with Pairwise Policy Gradient. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event China, July 2020), ACM, pp. 509–518.

[143] Yang, Liu, Ai, Qingyao, Spina, Damiano, Chen, Ruey-Cheng, Pang, Liang, Croft, W. Bruce, Guo, Jiafeng, and Scholer, Falk. Beyond factoid QA: effective methods for non-factoid answer sentence retrieval. In *ECIR 2016, Padua, Italy, March 20-23, 2016. Proceedings* (2016), pp. 115–128.

[144] Yang, Wei, Zhang, Haotian, and Lin, Jimmy. Simple applications of BERT for ad hoc document retrieval. *CoRR abs/1903.10972* (2019).

[145] Yang, Yi, Yih, Wen-tau, and Meek, Christopher. Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP '15* (Sept. 2015).

[146] Yih, Wen-Tau, Chang, Ming-Wei, Meek, Christopher, and Pastusiak, Andrzej. Question answering using enhanced lexical semantic models. In *ACL* (Aug. 2013), ACL.

[147] Yin, Wenpeng, Schütze, Hinrich, Xiang, Bing, and Zhou, Bowen. ABCNN: attention-based convolutional neural network for modeling sentence pairs. *TACL 4* (2016), 259–272.

[148] Yosinski, Jason, Clune, Jeff, Nguyen, Anh, Fuchs, Thomas, and Lipson, Hod. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579* (2015).

[149] Zamani, Hamed, and Croft, W. Bruce. Learning a joint search and recommendation model from user-item interactions. In *Proceedings of the 13th International Conference on Web Search and Data Mining* (New York, NY, USA, 2020), WSDM '20, Association for Computing Machinery, p. 717–725.

[150] Zamani, Hamed, Mitra, Bhaskar, Song, Xia, Craswell, Nick, and Tiwary, Saurabh. Neural ranking models with multiple document fields. In *Proc. WSDM* (2018), ACM, pp. 700–708.

[151] Zaremba, Wojciech, Sutskever, Ilya, and Vinyals, Oriol. Recurrent neural network regularization. *CoRR abs/1409.2329* (2014).

[152] Zeng, Wei, Xu, Jun, Lan, Yanyan, Guo, Jiafeng, and Cheng, Xueqi. Multi Page Search with Reinforcement Learning to Rank. In *Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval* (Tianjin, China, Sept. 2018), ICTIR '18, Association for Computing Machinery, pp. 175–178.

[153] Zhai, Chengxiang, and Lafferty, John. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems (TOIS) 22*, 2 (2004), 179–214.

[154] Zhang, Chiyuan, Bengio, Samy, Hardt, Moritz, Recht, Benjamin, and Vinyals, Oriol. Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings* (2017).

[155] Zhang, Guodong, Martens, James, and Grosse, Roger B. Fast convergence of natural gradient descent for over-parameterized neural networks. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8082–8093.

[156] Zhang, Sicong, Luo, Jiyun, and Yang, Hui. A pomdp model for content-free document re-ranking. In *Proceedings of the 37th International ACM SIGIR Conference on Research Development in Information Retrieval* (New York, NY, USA, 2014), SIGIR '14, Association for Computing Machinery, p. 1139–1142.

[157] Zhang, Weinan, Chen, Tianqi, Wang, Jun, and Yu, Yong. Optimizing top-n collaborative filtering via dynamic negative item sampling. In *SIGIR* (2013), ACM, pp. 785–788.

[158] Zhang, Xiang, Zhao, Junbo, and LeCun, Yann. Character-level convolutional networks for text classification. In *NIPS* (Montreal, Canada, 2015), NIPS'15, pp. 649–657.