A Hybrid Embedding Approach to Noisy Answer Passage Retrieval

Daniel Cohen and W. Bruce Croft

College of Information and Computer Sciences, University of Massachusetts Amherst, Amherst, MA, USA
{dcohen,croft}@cs.umass.edu

Abstract. Answer passage retrieval is an increasingly important information retrieval task as queries become more precise and mobile and audio interfaces more prevalent. In this task, the goal is to retrieve a contiguous series of sentences (a passage) that concisely addresses the information need expressed in the query. Recent work with deep learning has shown the efficacy of distributed text representations for retrieving sentences or tokens for question answering. However, determining the relevancy of answer passages remains a significant challenge, specifically when there exists a lexical and semantic gap between the text representation used for training and the collection's vocabulary. In this paper, we demonstrate the flexibility of a character based approach on the task of answer passage retrieval, agnostic to the source of embeddings and with improved performance in P@1 and MRR metrics over a word based approach as the collections degrade in quality.

Keywords: Answer Passage, Representation Learning, Hybrid Embedding

1 Introduction

A key part of an effective information retrieval (IR) system is the ability to identify the specific text relevant to a query. For some queries, the relevant text consists of documents or passages topically related, while other queries can best be answered by a few select words without the need for any additional text. The latter, known as factoid question answering (QA), has received significant attention with the rise of deep neural networks, achieving state of the art performance over term frequency based methods [17, 19, 5]. However, these neural models cannot be directly applied to larger bodies of text without a large degradation in performance [2].

Passage retrieval techniques have previously been developed to locate highly topically relevant passages spanning multiple sentences in documents [1, 23]. Answer passage retrieval focuses on finding text passages that directly answer questions expressed in more precise queries. In contrast to factoid QA that identifies specific words in a sentence, each word in the passage contributes to answering the query's information need. Thus, answer passage retrieval models must be able to capture long term dependencies commonly found in language in order to determine the relevancy of a passage. This disparity can be more easily seen with the following query:

Factoid QA Queries: When did James Dean die? How high is Everest?
Answer Passage Query: What kinds of harm do cruise ships do to sea life such as coral reefs, and what is the extent of their damage?

While the factoid QA queries rely on a few keywords and a specific request for information which can be resolved with a single word or number, answer passage queries require a more elaborate answer combining multiple aspects into a unified passage. The open ended nature of answer passage retrieval makes it difficult to apply these factoid QA models directly to answer passage retrieval [3]. This difficulty is compounded by the fact that passages can have little term overlap with the query, and poses a significant obstacle for conventional approaches as they rely on transforming text into vectors using domain knowledge.

One can view the difference in vocabulary as a missing text problem, in which case past work using character n-grams [8] has shown to be an effective method, out performing equivalent word based models on the task of retrieving optical character recognition degraded text. Recent work such as the Deep Structured Semantic Model (DSSM) [10] and the Deep Relevance Matching Model [7] has bridged the gap between document and answer passage retrieval by using a similar character hashing approach or a distributed representation to model interactions. However, they do not accurately retrieve answer passages as they are unable to model the sequential long term dependencies which contribute to the relevance of a passage.

Both of these issues are exacerbated by the caveat that pre-trained embeddings do not adapt well to collections with different vocabularies or where language differs from the corpus used for the word embedding training. The lexical and semantic shift permeates the network's hidden layers, resulting in a significant loss of performance when compared to embeddings trained on the actual test collection, particularly for the answer passage retrieval task [3]. Retraining embeddings to reflect a new collection can consistently improve performance [4]; however, it is impractical to create new local embeddings at run time as recent methods [3, 4, 15] rely on a time consuming optimization process requiring large amounts of data.

We approach these two challenges inherent to answer passage retrieval by leveraging a long short term memory (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.

2 Related Work

Deep learning for IR related tasks excels where standard approaches have failed. These methods almost all rely on a distributed representation of the vocabulary, referred to as word embeddings. The most common method in IR is the work by Mikolov et al. [15], word2vec, where they train a small neural network to predict the context around a word. The internal hidden representation of the network when predicting the context around a word becomes the embedding for that word. This results in similar words, such as *cat, dog, pet*, to have similar hidden representations.

Wang and Nyberg [21] use these embbeddings as input into their BiLSTM networks as an effective method for retrieving non-factoid answers. They use Google's pre-trained word2vec embeddings and boost the output of the network with term frequency statistics. This approach is able to outperform conventional IR approaches without the need of feature engineering.

Tan et al. [19] expand on neural retrieval for QA and create larger LSTM-CNN and CNN-LSTM networks. In this work, the initial layer is a BiLSTM layer with an attention mechanism, and feeds into a CNN. An attention mechanism allows the network to focus on information specifically relevant to both the query and answer rather than modeling the entire text independently. The final output of their networks is the cosine similarity between the question and answer embeddings. Again, this work uses google's pre-trained word2vec embeddings as the initial input. They use a similar attention mechanism, but prime the network with the query prior to processing the answer text, allowing the network to attend temporally. Santos et al. [5] investigate another attention mechanism by pooling the rows and columns of a similarity matrix and use the softmax to weight the LSTM or CNN representation of the question and answer respectively. This has been shown to outperform the method used in [19].

Cohen and Croft [3] demonstrate that updating word embeddings via backpropagation during training results in significant improvement when compared to standard embeddings. Diaz et al. [4] propose training word embedding vectors on topically-constrained corpora, instead of large topically-unconstrained corpora. These locally trained embedding vectors were shown to perform well for the query expansion task.

Due to the limitations of word embeddings with unseen vocabulary and new collections, Zhang et al. [25] demonstrate that a character level embedding fed into a deep CNN is an effective method of categorizing text. The deep CNN is able to recognize abstract text concepts and apply them to ontology classification, sentiment analysis, and text categorization. They compare their CNN to a word2vec [15] based LSTM model, and the CNN results in lower testing errors on all datasets.

Kim et al. [12] leverage this work to create a hybrid CNN-LSTM neural language model. Their network involves a single layer CNN with temporal pooling that feeds into an LSTM model to predict the next character. Their model parses each word separately as a concatenation of character embeddings as opposed to [25], which concatenates the entire passage. Again, the character based approach outperforms the word based approach using perplexity as a metric, and it is able to robustly handle words not seen during training.

In the realm of IR, Huang et al. [10] have capitalized on character level representations when creating deep structured semantic model (DSSM). They chose to represent the text as a series of character trigrams rather than the conventional word based approach. For example, the word *#good#* would be represented as (*#go, goo, ood, od#*), where # represents the start and end of a word. This reduces the dimensionality of one hot encodings from the size of the vocabulary to the number of distinct trigrams found in the collection, resulting in a 4 to 16 fold reduction. In addition, this character based approach allows for scaling up to very large vocabularies for use in realistic web searches.

There has been work in using convolutional networks to construct representations for short text; however, this has only been done on short factoid text or knowledge base (KB) question answering. In general, these models do not attempt to capture long term dependencies critical in an answer passage retrieval task and do not leverage recurrent networks for learning passage length representations. Golub and Ziadong [9] use a character CNN to capture deep representations of KB entity and predicates for factoid QA over a KB. Meng et al. [13] use a structure similar to Severyn and Moschitti [17] with the input as character embeddings, which works well on factoid QA tasks such as TREC QA, but fails to outperform traditional baselines as the passages increase in length [3].

3 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 1 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.

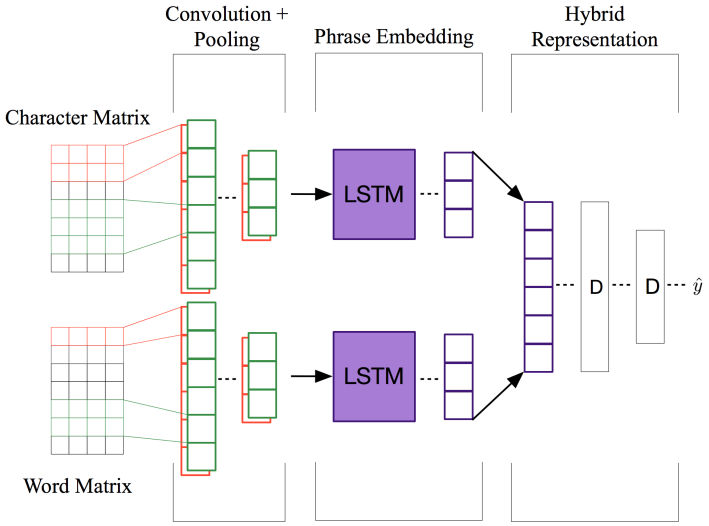

Fig. 1. A compressed representation of the Hybrid architecture.

3.1 Character Embeddings

As opposed to previous work in IR with neural networks [14, 19, 17, 5], 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 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. [15] with a skipgram window of 5 to create the character embeddings.

As in [25], 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 removed as they did not improve performance, and k was chosen to be 20. The 33 other characters are shown below:

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

3.2 Embedding-level Convolutional Layer

One can view a convolution as sliding a fixed width filter, $\mathbf{f}$, 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 [17, 19].

3.3 Recurrent Neural Network

A recurrent neural network (RNN) is a type of neural network architecture that captures temporal information [6]. Information previously seen is represented in the network's internal state h_{t-i} from the previous step. The cumulative nature of the hidden state, $\mathbf{h}_t$ results in the network incorporating all of the inputs up to t. As passage retrieval involves capturing long term dependencies over multiple sentences, this type of network is uniquely suited to the task of answer passage retrieval.

To adequately handle the length of answer passages, we adopt a modified RNN structure, called bidirectional long short term memory (BiLSTM) networks [6]. This architecture adds additional structures to a RNN to better control the information across sequential inputs by using internal gates with their own activation functions as well as processing the text in both directions and summing the representations at each timestep. All internal activations of the BiLSTM cell were $tanh$ functions.

3.4 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 timesteps 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 1, that represents the relevance of the input.

3.5 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 [6]. 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 [24, 14, 16], we adopt a method that primes the network similar to machine translation [21] 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 [19, 13, 17].

4 Experiments

The three datasets used for our experiments were (1) Yahoos Webscope L4, (2) nfl6, which is a lower quality answer passage set created from Yahoos general Webscope L6[1], and (3) a web answer passage collection, called WebAP[2]. These collections were chosen to reflect the answer passage retrieval task while still possessing distinct properties. Training, validation, and testing sets were created via a 64-16-20 split. Detailed statistics for each collection are shown in Table 2

L4 dataset: This Yahoo collection has been used previously in [18, 21, 3] for answer passage retrieval and is sometimes referred to as the "manner" collection and are high

[1] https://ciir.cs.umass.edu/downloads/WebAP/

[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 1. 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.

quality. Each question contains a noun and verb, and each answer is well formed. An example query from this collection indicative of the quality is "*How can I safely open a geode?*" 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 to remove label noise during training and provide more accurate results during evaluation.

nfL6 dataset: Introduced in [3], this dataset consists of 87,361 questions that are best answered by a passage rather than a single sentence. Unlike L4, the questions in this dataset are more generic, such as *"Why do teachers go abroad?"* and *"Why do people steal?"*. Furthermore, answers are not as high quality due to the method of creation.

WebAP dataset: In order to investigate the performance of a character based model in a different retrieval environment, we use the WebAP collection from Keikha et al. [11]. In contrast to the above 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.

.

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

4.1 Baselines

We compare our Hybrid model to previous deep learning implementations and BM25. As little 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 [21] non-factoid BiLSTM model prior to boosting, Tan et al.'s [19] factoid QA CNN-LSTM model, and Severyn and Moschitti's Convolutional Deep Neural Network (CDNN) [17]. 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 Word-CNN-LSTM and Char-CNN-LSTM respectively. Exact configurations of the BiLSTM, Char-CNN-LSTM, and Word-CNN-LSTM models are shown in Table 1. We also include DSSM [10] as a competitive character level baseline and DRMM [7] 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].

4.2 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 passages as well as the importance on the first passage retrieved for mobile and audio search. Similar to [3, 21], the test collection was created from pooling the top 10 results from a BM25 search for each question, and including the correct answer passage as the 10^{th} answer if it is not included in the list. We perform this adjustment in order to include all queries even if BM25 fails to return a top ten result as to not bias evaluation towards models that favor term frequency features. For WebAP, the top 100 retrieved results were used for reranking in the same manner, and five fold cross-validation was performed for evaluation.

4.3 Setup and Training

Our CNN-LSTM based networks were optimized via RMSprop [20] over a binary cross entropy function. The networks were trained until the metrics over the validation set stopped improving.

5 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 Char-CNN-LSTM and Word-CNN-LSTM 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. Of particular note is the poor performance of the traditional factoid or sentence QA models, Severyn and Moschitti's CDNN [17] and Tan et al. cosine similarity based approach [19]. While both of these models perform close to

[3] https://code.google.com/archive/p/word2vec/
[4] https://dumps.wikimedia.org/enwiki/20160501/

Implementation		L4		nfl6		WebAP	
		P@1	**MRR**	**P@1**	**MRR**	**P@1**	**MRR**
BM25		.0738	.1412	.1312	.2660	.3000	.4120
DSSM [10]		.0805	.2477	.0905	.2576	.2150	.3127
DRMM [7]		.1416	.3291	.2844	.3350	.2558	.4064
Tan et al. [19]		.2473	.4217	.2139	.3934	.2047	.3612
CDNN [17]		.0989	.2434	.1438	.2842	.2122	.3834
BiLSTM [21]	Local	.4726	.6329	.2471	.4710	.3177	.4618
	Pretrained	.4492	.6129	.2332	.4287	.3059	.4502
Word-CNN-LSTM	Local	.4523	.6206	.3482	.5327	.2947	.4136
	Pretrained	.4514	.6190	.3406	.5236	.3159	.4411
Char-CNN-LSTM	Local	.4132	.5801	.3211	.4987	.3531	.5148
	Pretrained	.4137	.5798	.3214	.4983	**.3533**	**.5150**
Hybrid	Local	.4608†	.6241	**.3517***†	5429*†	.3017	.4410
	Pretrained	**.4798***†	**.6407***†	.3516*†	**.5433***†	.3215	.4716

Table 3. 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 (Word/Char-CNN-LSTM

state of the art on WikiQA and TREC QA, they achieve significantly worse results on the answer passage retrieval task. While not benchmarked, Meng et al. [13] possess a similar structure to CDNN and thus, Meng et al.'s character based model would not perform well due to the shared architecture and lack of temporal structure.

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 [22]. 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 Char-CNN-LSTM component of the Hybrid model outperforming both a standard BiLSTM and the Word-CNN-LSTM on the WebAP collection.

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 Word-CNN-LSTM and Char-CNN-LSTM consisting of only word and character embedding inputs respectively. Examining Word-CNN-LSTM'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 CNN-LSTM interaction. 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 Char-CNN-LSTM 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.

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. Updating word embeddings during training allows for a richer representation for the network to use in the hidden layers [3]. 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 Word-CNN-LSTM model suffers when moving from local to pretrained embeddings.

Cross Collection Performance As the Hybrid model was able to effectively handle the noisy nfl6, we investigated the performance of these models across collections. Specifically, we evaluated the ability of these models to generalize outside of the language distribution in which they were trained. The BiLSTM and Hybrid models, as well as the Word-CNN-LSTM and Char-CNN-LSTM subcomponents, were trained on both L4 and nfl6 collections and evaluated on the WebAP dataset. The results in Table 5 show that the Hybrid approach is again the most robust to retrieval tasks that significantly differ from the training collection. Vocabulary overlap between the training and evaluation set are shown in Table 5.

Model	**Trained**	**Transfer**	**P@1**	**MRR**
BM25	N/A	WebAP	.3000	.4120
BiLSTM [21]	L4+nfl6	WebAP	.0941	.2116
Word-CNN-LSTM	L4+nfl6	WebAP	.1176	.2718
Char-CNN-LSTM	L4+nfl6	WebAP	.1602*	.2937*
Hybrid	L4+nfl6	WebAP	**.1836***	**.3115***

Table 4. 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.

P@1	**Shared Vocabulary**	Char-CNN-LSTM	Word-CNN-LSTM	BiLSTM
Correct	Total Overlap	.42	.57	.59
	μ	.38	.42	.45
	σ	.08	.05	.06

Table 5. A representation of the vocabulary overlap between training and correctly ranked candidates answers. Answers are labeled correct only if they are ranked first, reflecting the P@1 metric. Total Overlap is shared vocabulary across all questions, while μ and σ are mean and standard deviation of shared vocabulary with respect to individual questions.

6 Conclusion and Future Work

This paper demonstrates the use of a CNN-LSTM model based on hybrid embedding for the complex task of answer passage retrieval. We leverage past work with character based models to bridge the gap between local and global embeddings performance for answer passage tasks. Additionally, we illustrate the advantage of incorporating a character embedding with temporal structure for collections that suffer from a small number of training examples or lower quality tokens.

We evaluated our models on three collections, Yahoo Webscope L4, nfl6, and WebAP. The Hybrid model outperforms all baselines while avoiding the use of local embeddings save for the case of WebAP, where only the character subnetwork has significantly greater performance. Incorporating character embeddings into the CNN-LSTM structure results in a model able to adapt especially well compared to a standard BiLSTM network. Given that the baselines perform well on the factoid QA or ad-hoc retrieval tasks, their relatively poor performance on answer passage retrieval demonstrates that this is a different task. While not looked at in this paper, a potential improvement to bridge the gap between word and character embeddings would be to use a gate mechanism when joining the word and character sub networks through a dynamic process.

7 Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval, in part by NSF IIS-1160894 and in part by NSF grant #IIS-1419693. 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.

References

1. J. P. Callan. Passage-level evidence in document retrieval. In *ACM SIGIR*, SIGIR '94, pages 302–310, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

2. D. Cohen, Q. Ai, and W. B. Croft. Adaptability of neural networks on varying granularity ir tasks. In *SIGIR Neu-IR Workshop*, Pisa, Italy, 2016.
3. D. Cohen and W. B. Croft. End to end long short term memory networks for non-factoid question answering. In *ICTIR*, Newark, DE, USA, 2016.
4. F. Diaz, B. Mitra, and N. Craswell. Query expansion with locally-trained word embeddings. In *ACL*, pages 367–377. ACL, 2016.
5. C. N. dos Santos, M. Tan, B. Xiang, and B. Zhou. Attentive pooling networks. *CoRR*, abs/1602.03609, 2016.
6. A. Graves, N. Jaitly, and A.-R. Mohamed. Hybrid speech recognition with deep bidirectional lstm. In *ASRU, 2013*, pages 273–278, Dec 2013.
7. J. Guo, Y. Fan, Q. Ai, and W. B. Croft. A deep relevance matching model for ad-hoc retrieval. In *CIKM '16*, pages 55–64, New York, NY, USA, 2016. ACM.
8. S. M. Harding, W. B. Croft, and C. Weir. Probabilistic retrieval of ocr degraded text using n-grams. In *ECDL*, pages 345–359, Pisa, Italy, 1997. Springer.
9. X. He and D. Golub. Character-level question answering with attention. In *EMNLP*, pages 1598–1607. ACL, 2016.
10. P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. Learning deep structured semantic models for web search using clickthrough data. In *CIKM '13*, pages 2333–2338. ACM, 2013.
11. M. Keikha, J. Park, and W. B. Croft. Evaluating answer passages using summarization measures. In *SIGIR 2014*, pages 963–966, 2014.
12. Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush. Character-aware neural language models. In *AAAI*, pages 2741–2749, 2016.
13. L. Meng, Y. Li, M. Liu, and P. Shu. Skipping word: A character-sequential representation based framework for question answering. In *CIKM '16*, pages 1869–1872, 2016.
14. Y. Miao, L. Yu, and P. Blunsom. Neural variational inference for text processing. In *ICML*, pages 1727–1736. JMLR.org, 2016.
15. T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *ICLR Workshop*, Scottsdale, AZ, USA, 2013.
16. M. J. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi. Bidirectional attention flow for machine comprehension. In *ICLR*, Toulon, France, 2017.
17. A. Severyn and A. Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *SIGIR*, SIGIR '15, pages 373–382, New York, NY, USA, 2015. ACM.
18. M. Surdeanu, M. Ciaramita, and H. Zaragoza. Learning to rank answers on large online qa collections. In *ACL:HLT*, pages 719–727, 2008.
19. M. Tan, B. Xiang, and B. Zhou. Lstm-based deep learning models for non-factoid answer selection. *CoRR*, abs/1511.04108, 2015.
20. T. Tielman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. 2012.
21. D. Wang and E. Nyberg. 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*, pages 707–712, 2015.
22. K. Wei, R. Iyer, and J. Bilmes. Submodularity in data subset selection and active learning. In *ICML*, Lille, France, 2015.
23. L. Yang, Q. Ai, D. Spina, R. Chen, L. Pang, W. B. Croft, J. Guo, and F. Scholer. Beyond factoid QA: effective methods for non-factoid answer sentence retrieval. In *ECIR 2016, Padua, Italy, March 20-23, 2016. Proceedings*, pages 115–128, 2016.
24. W. Yin, H. Schütze, B. Xiang, and B. Zhou. ABCNN: attention-based convolutional neural network for modeling sentence pairs. *TACL*, 4:259–272, 2016.
25. X. Zhang, J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. In *NIPS*, NIPS'15, pages 649–657, Montreal, Canada, 2015.