Discriminative Information Retrieval for Question Answering Sentence Selection

Tongfei Chen and **Benjamin Van Durme**
Johns Hopkins University
{tongfei,vandurme}@cs.jhu.edu

Abstract

We propose a framework for discriminative IR atop linguistic features, trained to improve the recall of answer candidate passage retrieval, the initial step in text-based question answering. We formalize this as an instance of linear feature-based IR, demonstrating a 34% - 43% improvement in recall for candidate triage for QA.

1 Introduction

Question answering (QA) with textual corpora is typically modeled as first finding a candidate set of passages (sentences) that may contain an answer to a question, followed by an optional candidate reranking stage, and then finally an information extraction (IE) step to select the answer string. QA systems normally employ an information retrieval (IR) system to produce the initial set of candidates, usually treated as a black box, bag-of-words process that selects candidate passages best overlapping with the content in the question.

Recent efforts in corpus-based QA have been focused heavily on reranking, or *answer sentence selection*: filtering the candidate set as a supervised classification task to single out those that *answer* the given question. Extensive research has explored employing syntactic/semantic features (Yih et al., 2013; Wang and Manning, 2010; Heilman and Smith, 2010; Yao et al., 2013a) and recently using neural networks (Yu et al., 2014; Severyn and Moschitti, 2015; Wang and Nyberg, 2015; Yin et al., 2016). The shared aspect of all these approaches is that the quality of reranking a candidate set is upper-bounded by the initial set of candidates: unless one plans on reranking the *entire* corpus for each question as it arrives, one is still reliant on an initial IR stage in order to obtain a computationally feasible QA system. Huang et al. (2013) used neural networks and cosine distance to rank the candidates for IR, but without providing a method to search for the relevant documents in sublinear time.

We propose a framework for performing this triage step for QA sentence selection and other related tasks in sublinear time. Our method shows a log-linear model can be trained to optimize an objective function for downstream reranking, and the resulting trained weights can be reused to retrieve a candidate set. The content that our method retrieves is what the downstream components are known to prefer: it is *trainable* using the same data as employed in training candidate reranking. Our approach follows Yao et al. (2013b) who proposed the automatic coupling of QA sentence selection and IR by augmenting a bag-of-words query with desired named entity (NE) types based on a given question. While Yao et al. showed improved performance in IR as compared with an off-the-shelf IR system, the model was proof-of-concept, employing a simple linear interpolation between bag-of-words and NE features with a single scalar value tuned on a development set, kept static across all types of questions at test time. We generalize Yao et al.'s intuition by casting the problem as an instance of classification-based retrieval (Robertson and Spärck Jones, 1976), formalized as a discriminative retrieval model (Cooper et al., 1992; Gey, 1994; Nallapati, 2004) allowing for the use of NLP features. Our framework can then be viewed as an instance of linear feature-based IR, following Metzler and Croft (2007).

To implement this approach, we propose a general feature-driven abstraction for coupling retrieval and answer sentence selection.[1] Our experiments demonstrate state-of-the-art results on QA sentence selection on the dataset of Lin and Katz

[1] https://github.com/ctongfei/probe.

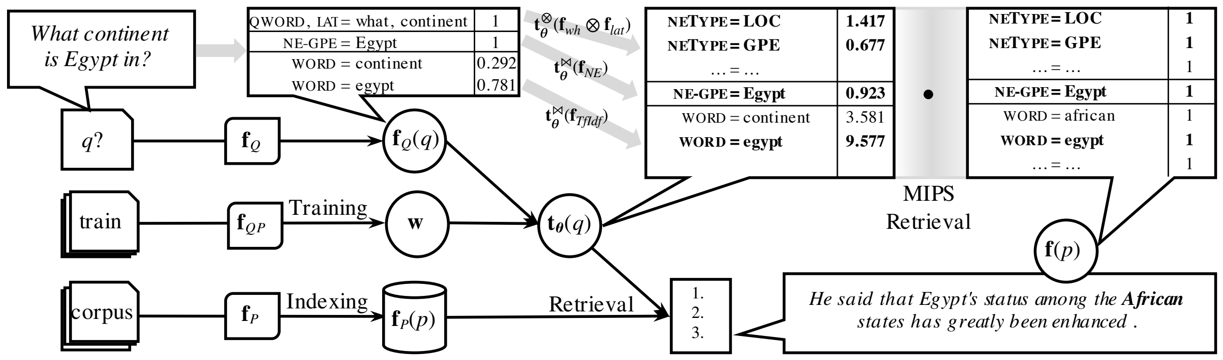

Figure 1: Steps in mapping natural language questions into weighted features used in retrieval.

(2006), and we show significant improvements over a bag-of-words of baseline on a novel Wikipedia-derived dataset we introduce here, based on WIKIQA (Yang et al., 2015).

2 Approach

Formally, given a candidate set $\mathcal{D} = \{p_1, \cdots, p_N\}$, a query q and a scoring function $F(q, p)$, an IR system retrieves the top-k items under the objective

$$\arg\max_{p \in \mathcal{D}} F(q, p). \tag{1}$$

If the function F is simple enough (e.g. *tf-idf*), it could be easily solved by traditional IR techniques. However, tackling this problem with a complex F via straightforward application of supervised classification (e.g., recent neural network based models) requires a traversal over all possible candidates, i.e. the corpus, which is computationally infeasible for any reasonable collection.

Let $\mathbf{f}_Q(q)$ refer to feature extraction on the query q, with corresponding candidate-side feature extraction $\mathbf{f}_P(p)$ on the candidate, and finally $\mathbf{f}_{QP}(q, p)$ extracts features from a (query, candidate) pair is defined in terms of $\mathbf{f}_Q$ and $\mathbf{f}_P$ via composition (defined later):

$$\mathbf{f}_{QP}(q, p) = C(\mathbf{f}_Q(q), \mathbf{f}_P(p)). \tag{2}$$

From a set of query/candidate pairs we can train a model M such that given the feature vector of a pair (q, p), its returning value $M(\mathbf{f}_{QP}(q, p))$ represents the predicted probability of whether the passage p answers the question q. This model is chosen to be a log-linear model with the feature weight vector $\boldsymbol{\theta}$, leading to the optimization problem

$$\arg\max_{p \in \mathcal{D}} \boldsymbol{\theta} \cdot \mathbf{f}_{QP}(q, p). \tag{3}$$

This is in accordance with the pointwise reranker approach, and is an instance of the linear feature-based model of Metzler and Croft (2007). Under specific compositional operations in $\mathbf{f}_{QP}$ the following transformation can be made:

$$\boldsymbol{\theta} \cdot \mathbf{f}_{QP}(q, p) = \mathbf{t}_{\boldsymbol{\theta}}(\mathbf{f}_Q(q)) \cdot \mathbf{f}_P(p). \tag{4}$$

This is elaborated in § 4. We project the original feature vector of the query $\mathbf{f}_Q(q)$ to a transformed version $\mathbf{t}_{\boldsymbol{\theta}}(\mathbf{f}_Q(q))$: this transformed vector is dependent on the model parameters $\boldsymbol{\theta}$, where the association learned between the query and the candidate is incorporated into the transformed vector. This is a weighted, trainable generalization of *query expansion* in traditional IR systems.

Under this transformation we observe that the joint feature function $\mathbf{f}_{QP}(q, p)$ is decomposed into two parts with no interdependency – the original problem in Eq. (4) is reduced to a standard *maximum inner product search* (MIPS) problem as seen on the RHS of Eq. (4). Under sparse assumptions (where the query vector and the candidate feature vector are both sparse), this MIPS problem can be efficiently (sublinearly) solved using classical IR techniques (multiway merging of postings lists).

3 Features

A feature vector can be seen as an associative array that maps features in the form "KEY=*value*" to real-valued weights. One item in a feature vector $\mathbf{f}$ is denoted as "(KEY $=$ *value*, *weight*)", and a feature vector can be seen as a set of such tuples. We write $\mathbf{f}_{(\text{KEY}=\textit{value})} = \textit{weight}$ to indicate that the features serve as keys to the associative array, and θ_X is the weight of the feature X in the trained model $\boldsymbol{\theta}$.

3.1 Question features

$\mathbf{f}_{wh}$: Question word, typically the *wh*-word of a sentence. If it is a question like "*How many*", the word after the question word is also included in the feature, i.e., feature "(QWORD=*how many*, 1)" will be added to the feature vector.

$\mathbf{f}_{lat}$: Lexical answer type (LAT), if the query has a question word:"what" or "which", we identify the LAT of this question (Ferrucci et al., 2010), which is defined as the head word of the first NP after the question word. E.g., "*What is the city of brotherly love?*" would result in "(LAT=*city*, 1)". [2]

$\mathbf{f}_{NE}$: All the named entities (NE) discovered in this question. E.g., "(NE-PERSON=*Margaret Thatcher*, 1)" would be generated if Thatcher is mentioned.

$\mathbf{f}_{TfIdf}$: The L_2-normalized *tf-idf* weighted bag-of-words feature of this question. An example feature would be "(WORD = *author*, 0.454)".

3.2 Passage features

All passage features are constrained to be binary.

$\mathbf{f}_{BoW}$: Bag-of-words: any distinct word *x* in the passage will generate a feature "(WORD=*x*, 1)".

$\mathbf{f}_{NEType}$: Named entity type. If the passage contains a name of a person, a feature "(NE-TYPE=PERSON, 1)" will be generated.

$\mathbf{f}_{NE}$: Same as the NE feature for questions.

4 Feature vector operations

Composition Here we elaborate the composition C of the question feature vector and passage feature vector, defining two operators on feature vectors: Cartesian product ($\otimes$) and join ($\bowtie$).

For any feature vector of a question $\mathbf{f}_Q(q) = \{(k_i = v_i, w_i)\}, (w_i \leq 1)$[3] and any feature vector of a passage $\mathbf{f}_P(p) = \{(k_j = v_j, 1)\}$, the Cartesian product and join of them is defined as

$$\begin{aligned} \mathbf{f}_Q(q) \otimes \mathbf{f}_P(p) &= \{((k_i, k_j) = (v_i, v_j), w_i)\} \\ \mathbf{f}_Q(q) \bowtie \mathbf{f}_P(p) &= \{((k_i = k_j) = 1, w_i)\}. \end{aligned}$$

Notation $(k_i = k_j) = 1$ denotes a feature for a question/passage pair, that when present, witnesses the fact that that the value for feature k_i on the question side is the same as the feature k_j on the passage side.

The composition that generates the feature vector for the question/passage pair is therefore defined as

$$\begin{array}{lllll} C(& \mathbf{f}_Q(q) & , & \mathbf{f}_P(p) &) \\ = & (\mathbf{f}_{wh}(q) \otimes \mathbf{f}_{lat}(q)) & \otimes & \mathbf{f}_{NEType}(p) & \\ + & (\mathbf{f}_{wh}(q) \otimes \mathbf{f}_{lat}(q)) & \otimes & \mathbf{f}_{BoW}(p) & \\ + & \mathbf{f}_{NE}(q) & \bowtie & \mathbf{f}_{NE}(p) & \\ + & \mathbf{f}_{TfIdf}(q) & \bowtie & \mathbf{f}_{BoW}(p) & . \end{array} \quad (5)$$

$(\mathbf{f}_{wh}(q) \otimes \mathbf{f}_{lat}(q)) \otimes \mathbf{f}_{NEType}(p)$ captures the association of question words and lexical answer types with the expected type of named entities. $(\mathbf{f}_{wh}(q) \otimes \mathbf{f}_{lat}(q)) \otimes \mathbf{f}_{BoW}(p)$ captures the relation between some question types with certain words in the answer. $\mathbf{f}_{NE}(q) \bowtie \mathbf{f}_{NE}(p)$ captures named entity overlap between questions and answering sentences.

$\mathbf{f}_{TfIdf}(q) \bowtie \mathbf{f}_{BoW}(p)$ measures general *tf-idf*-weighted context word overlap. Using only this feature without the others effectively reduces the system to a traditional *tf-idf*-based retrieval system.

Projection Given a question, it is desired to know what kind of features that its potential answer might have. Once this is known, an index searcher will do the work to retrieve the desired passage.

For the Cartesian product of features, we define

$$\mathbf{t}^{\otimes}_{\boldsymbol{\theta}}(\mathbf{f}) = \{(k' = v', w\theta_{(k,k')=(v,v')}) | (k = v, w) \in \mathbf{f}\},$$

for all k', v' such that $\theta_{(k,k')=(v,v')} \neq 0$, i.e. feature $(k, k') = (v, v')$ appears in the trained model.

For join, we have

$$\mathbf{t}^{\bowtie}_{\boldsymbol{\theta}}(\mathbf{f}) = \{(k' = v, w\theta_{(k=k')=1}) | (k = v, w) \in \mathbf{f}\},$$

for all k' such that $\theta_{(k=k')=1} \neq 0$, i.e. feature $(k = k') = 1$ appears in the trained model.

It can be shown from the definitions above that

$$\begin{aligned} \mathbf{t}^{\otimes}_{\boldsymbol{\theta}}(\mathbf{f}) \cdot \mathbf{g} &= \boldsymbol{\theta} \cdot (\mathbf{f} \otimes \mathbf{g}); \\ \mathbf{t}^{\bowtie}_{\boldsymbol{\theta}}(\mathbf{f}) \cdot \mathbf{g} &= \boldsymbol{\theta} \cdot (\mathbf{f} \bowtie \mathbf{g}). \end{aligned}$$

Then the transformed feature vector $\mathbf{t}(q)$ of an expected answer passage given a feature vector of a question $\mathbf{f}_Q(q)$ is:

$$\mathbf{t}(q) = \mathbf{t}^{\otimes}_{\boldsymbol{\theta}}(\mathbf{f}_{wh}(q) \otimes \mathbf{f}_{lat}(q)) + \mathbf{t}^{\bowtie}_{\boldsymbol{\theta}}(\mathbf{f}_{NE}(q) + \mathbf{f}_{TfIdf}(q)).$$

Calculating the vector $\mathbf{t}(q)$ is computationally efficient because it only involves sparse vectors.

We have formally proved Eq. (4) by the feature vectors we proposed, showing that given a question, we can reverse-engineer the features we expect to be present in a candidate using the transformation function $\mathbf{t}_{\boldsymbol{\theta}}$, which we will then use as a query vector for retrieval.

[2] If the question word is not "what" or "which", generate an empty feature (LAT=$\emptyset$, 1).

[3] If $w_i > 1$, the vector can always be normalized so that the weight of every feature is less than 1.

Retrieval We use Apache LUCENE[4] to build the index of the corpus, which, in the scenario of this work, is the feature vectors of all candidates $\mathbf{f}_P(p), p \in \mathcal{D}$. This is an instance of weighted bag-of-features instead of common bag-of-words.

For a given question q, we first compute its feature vector $\mathbf{f}(q)$ and then compute its transformed feature vector $\mathbf{t}_{\boldsymbol{\theta}}(q)$ given model parameters $\boldsymbol{\theta}$, forming a weighted query. We modified the similarity function of LUCENE when executing multiway postings list merging so that fast efficient maximum inner product search can be achieved. This classical IR technique ensures sublinear performance because only vectors with at least one overlapping feature, instead of the whole corpus, is traversed. [5]

5 Experiments

TREC Data We use the training and test data from Yao et al. (2013b). Passages are retrieved from the AQUAINT Corpus (Graff, 2002), which is NER-tagged by the Illinois Named Entity Tagger (Ratinov and Roth, 2009) with an 18-label entity type set. Questions are parsed using the Stanford CORENLP (Manning et al., 2014) package. Each question is paired with 10 answer candidates from AQUAINT, annotated for whether it answers the question via crowdsourcing. The test data derives from Lin and Katz (2006), which contains 99 TREC questions that can be answered in AQUAINT. We follow Nallapati (2004) and undersample the negative class, taking 50 sentences uniformly at random from the AQUAINT corpus, per query, filtered to ensure no such sentence matches a query's answer pattern as negative samples to the training set.

Wikipedia Data We introduce a novel evaluation for QA retrieval, based on WIKIQA (Yang et al., 2015), which pairs questions asked to Bing with their most associated Wikipedia article, along with sentence-level annotations on the introductory section of those articles as to whether they answer the question. [6]

[4]`http://lucene.apache.org`.

[5]The closest work on indexing we are aware of is by Bilotti et al. (2007), who transformed linguistic structures to structured constraints, which is different from our approach of directly indexing linguistic features.

[6]Note that as compared to the TREC dataset, there are some questions in WIKIQA which are not answerable based on the provided context alone. E.g. "*who is the guy in the wheelchair who is smart*" has the answer "Professor Stephen Hawking , known for being a theoretical physicist , has appeared in many works of popular culture ." This sets the upper bound on performance with WIKIQA below 100% when using contemporary question answering techniques, as assumed here.

We automatically aligned WIKIQA annotations, which was based on an unreported version of Wikipedia, with the Feb. 2016 snapshot, using for our corpus the introductory section of *all* Wikipedia articles, processed with Stanford CORENLP. Alignment was performed via string edit distance, leading to a 55% alignment to the original annotations. Table 1 dev/test reflects the subset resulting from this alignment; all of the original WIKIQA train was used in training, along with 50 negative examples randomly sampled per question.

	# of questions			# of
	train	dev	test	sentences
TREC/AQUAINT	2150	53	99	23,398,942
WIKIQA/Wikipedia	2118	77	157	20,368,761

Table 1: Summary of the datasets.

Setup The model is trained using LIBLINEAR (Fan et al., 2008), with heavy L_1-regularization (feature selection) to the maximum likelihood objective. The model is tuned on the dev set, with the objective of maximizing recall.

Baseline systems Recent work in neural network based *reranking* is not directly applicable here as those are *linear* with respect to the number of candidate sentences, which is computationally infeasible given a large corpus.

Off-the-shelf LUCENE: Directly indexing the sentences in LUCENE and do sentence retrieval. This is equivalent to maximum *tf-idf* retrieval.

Yao et al. (2013b): A retrieval system which augments the bag-of-words query with desired named entity types based on a given question.

Evaluation metrics (1) *R@1k*: The recall in top-1000 retrieved list. Contrary to normal IR systems which optimize precision (as seen in metrics such as P@10), our system is a triaging system whose goal is to *retrieve good candidates* for downstream reranking: high recall within a large set of initial candidates is our foremost aim. (2) *b-pref* (Buckley and Voorhees, 2004): is designed for situations where relevance judgments are known to be far from complete,[7] computing a preference relation of whether judged relevant documents are retrieved ahead of judged irrelevant document; (3) *MAP*:

[7]This is usually the case in passage retrieval, where complete annotation of all sentences in a large corpus as to whether they answer each question is not feasible beyond a small set (such as the work of Lin and Katz (2006)).

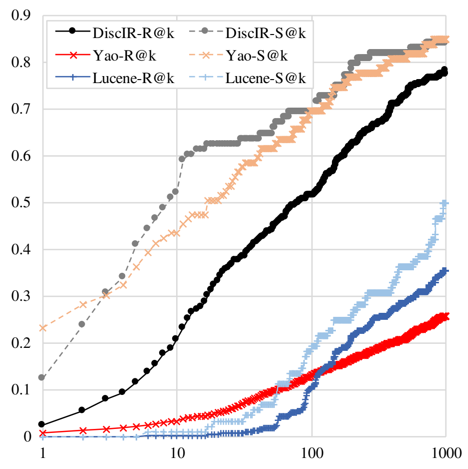

Figure 2: The R@k and S@k curve for different models in the TREC/AQUAINT setting.

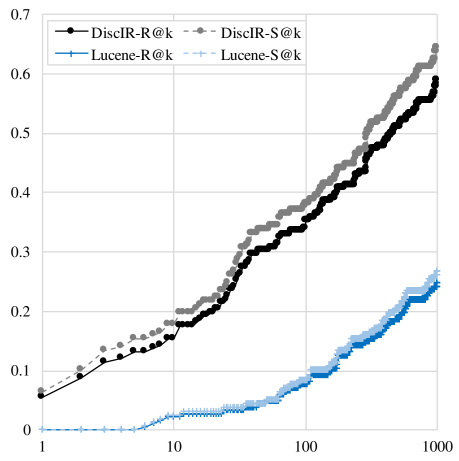

Figure 3: The R@k and S@k curve for different models in the WIKIQA/Wikipedia setting.

mean average precision; and (4) *MRR*: mean reciprocal rank. We are most concerned with (1,2), and (3,4) are reported in keeping with prior work.

Results Our approach (DiscIR) significantly outperforms Yao et al. in R@1k and b-pref, demonstrating the effectiveness of trained weighted queries compared to binary augmented features. The performance gain with respect to off-the-shelf LUCENE with reranking shows that our weighted augmented queries by decomposition is superior to vanilla *tf-idf* retrieval, as can be shown in Table 2.

	R@1k	**b-pref**	MAP	MRR
	TREC / AQUAINT			
LUCENE (dev)	52.44%	41.95%	9.63%	13.94%
LUCENE (test)	35.47%	38.22%	9.78%	15.06%
Yao+ (test)[8]	25.88%	45.41%	13.75%	**29.87%**
DiscIR (dev)	71.34%	70.69%	20.07%	30.34%
DiscIR (test)	**78.20%**	**75.15%**	**17.84%**	25.30%
	WIKIQA / Wikipedia			
LUCENE (dev)	25.00%	25.97%	1.83%	1.83%
LUCENE (test)	24.73%	25.69%	0.58%	0.72%
DiscIR (dev)	60.00%	61.69%	9.56%	9.65%
DiscIR (test)	**58.79%**	**60.88%**	**10.26%**	**11.42%**

Table 2: Performance of the QA retrieval systems.

We also plot the performance of these systems at different ks on a log-scale (shown in Fig. 2 and Fig. 3). We use two metrics here: recall at k (R@k) and success at k (S@k). Success at k is the percentage of queries in which there was at least one relevant answer sentence among the first k retrieved result by a specific system, which is the true upper bound for downstream tasks.

Again, DiscIR demonstrated significantly higher recalls than baselines at different ks and across different datasets. Success rate at different ks are also uniformly higher than LUCENE, and at most ks higher than the model of Yao et al.'s.

[8]Results on dev data is not reported in Yao et al. (2013b).

6 Conclusion and Future Work

Yao et al. (2013b) proposed coupling IR with features from downstream question answer sentence selection. We generalized this intuition by recognizing it as an instance of discriminative retrieval, and proposed a new framework for generating weighted, feature-rich queries based on a query. This approach allows for the straightforward use of a downstream feature-driven model in the candidate selection process, and we demonstrated how this leads to a significant gain in recall, b-pref and MAP, hence providing a larger number of correct candidates that can be provided to a downstream (neural) reranking model, a clear next step for future work.

Acknowledgements

Thank you to Xuchen Yao for assistance in evaluation against prior work, and to Paul McNamee and James Mayfield for feedback. This research benefited from support by a Google Faculty Award, the JHU Human Language Technology Center of Excellence (HLTCOE), and DARPA DEFT. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes. The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements of DARPA or the U.S. Government.

References

Matthew W. Bilotti, Paul Ogilvie, Jamie Callan, and Eric Nyberg. 2007. Structured retrieval for question answering. In *Proceedings of the ACM SIGIR conference on Research and Development in Information Retrieval*, pages 351–358.

Chris Buckley and Ellen M. Voorhees. 2004. Retrieval evaluation with incomplete information. In *Proceedings of the ACM SIGIR conference on Research and Development in Information Retrieval*, pages 25–32.

William S. Cooper, Fredric C. Gey, and Daniel P. Dabney. 1992. Probabilistic retrieval based on staged logistic regression. In *Proceedings of the ACM SIGIR conference on Research and Development in Information Retrieval*, pages 198–210.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research (JMLR)*, 9:1871–1874.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, et al. 2010. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79.

Fredric Gey. 1994. Inferring probability of relevance using the method of logistic regression. In *Proceedings of the ACM SIGIR conference on Research and Development in Information Retrieval*, pages 222–231.

David Graff. 2002. The AQUAINT Corpus of English News Text LDC2002T31. *Linguistic Data Consortium.*

Michael Heilman and Noah A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1011–1019. Association for Computational Linguistics.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. *Proceedings of CIKM*, pages 2333–2338.

Jimmy Lin and Boris Katz. 2006. Building a reusable test collection for question answering. *Journal of the American Society for Information Science and Technology*, 57(7):851–861.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60. Association for Computational Linguistics.

Donald Metzler and W. Bruce Croft. 2007. Linear feature-based models for information retrieval. *Information Retrieval*, 10(3):257–274.

Ramesh Nallapati. 2004. Discriminative models for information retrieval. In *Proceedings of the ACM SIGIR conference on Research and Development in Information Retrieval*, pages 64–71.

Lev Ratinov and Dan Roth, 2009. *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, chapter Design Challenges and Misconceptions in Named Entity Recognition, pages 147–155. Association for Computational Linguistics.

Stephen Robertson and Karen Spärck Jones. 1976. Relevance weighting of search terms. *Journal of American Society for Information Sciences*, 27(3):129–146.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the ACM SIGIR conference on Research and Development in Information Retrieval*, pages 373–382.

Mengqiu Wang and Christopher Manning. 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1164–1172. Coling 2010 Organizing Committee.

Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 707–712. Association for Computational Linguistics.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018. Association for Computational Linguistics.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013a. Answer extraction as sequence tagging with tree edit distance. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 858–867. Association for Computational Linguistics.

Xuchen Yao, Benjamin Van Durme, and Peter Clark. 2013b. Automatic coupling of answer extraction and information retrieval. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 159–165. Association for Computational Linguistics.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1744–1753. Association for Computational Linguistics.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association of Computational Linguistics*, 4:259–272.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *NIPS Deep Learning and Representation Learning Workshop*.