# Automatic Coupling of Answer Extraction and Information Retrieval

**Xuchen Yao** and **Benjamin Van Durme**
Johns Hopkins University
Baltimore, MD, USA

**Peter Clark**
Vulcan Inc.
Seattle, WA, USA

## Abstract

Information Retrieval (IR) and Answer Extraction are often designed as isolated or loosely connected components in Question Answering (QA), with repeated over-engineering on IR, and not necessarily performance gain for QA. We propose to tightly integrate them by coupling automatically learned features for answer extraction to a shallow-structured IR model. Our method is very quick to implement, and significantly improves IR for QA (measured in Mean Average Precision and Mean Reciprocal Rank) by 10%-20% against an uncoupled retrieval baseline in both document and passage retrieval, which further leads to a downstream 20% improvement in QA $F_1$.

## 1 Introduction

The overall performance of a Question Answering system is bounded by its Information Retrieval (IR) front end, resulting in research specifically on Information Retrieval for Question Answering (IR4QA) (Greenwood, 2008; Sakai et al., 2010). Common approaches such as query expansion, structured retrieval, and translation models show patterns of complicated engineering on the IR side, or isolate the upstream passage retrieval from downstream answer extraction. We argue that: 1. an IR front end should deliver exactly what a QA[1] back end needs; 2. many intuitions employed by QA should be and can be *re-used* in IR, rather than *re-invented*. We propose a coupled retrieval method with prior knowledge of its downstream QA component, that feeds QA with exactly the information needed.

As a motivating example, using the question `When was Alaska purchased` from the TREC 2002 QA track as the query to the Indri search engine, the top sentence retrieved from the accompanying AQUAINT corpus is:

```
Eventually Alaska Airlines will
allow all travelers who have
purchased electronic tickets
through any means.
```

While this relates `Alaska` and `purchased`, it is not a useful passage for the given question.[2] It is apparent that the question asks for a date. Prior work proposed predictive annotation (Prager et al., 2000; Prager et al., 2006): text is first annotated in a predictive manner (of what types of questions it might answer) with 20 answer types and then indexed. A question analysis component (consisting of 400 question templates) maps the desired answer type to one of the 20 existing answer types. Retrieval is then performed with both the question and predicated answer types in the query.

However, predictive annotation has the limitation of being labor intensive and assuming the underlying NLP pipeline to be accurate. We avoid these limitations by directly asking the downstream QA system for the information about *which entities answer which questions*, via two steps: 1. reusing the question analysis components from QA; 2. forming a query based on the most relevant answer features given a question from the learned QA model. There is no query-time overhead and no manual template creation. Moreover, this approach is more robust against, e.g., entity recognition errors, because answer typing knowledge is learned from how the data was *actually* labeled, not from how the data was *assumed* to be labeled (e.g., manual templates usually assume perfect labeling of named entities, but often it is not the case

[1]After this point in the paper we use the term QA in a narrow sense: QA without the IR component, i.e., answer extraction.

[2]Based on a non-optimized IR configuration, none of the top 1000 returned passages contained the correct answer: 1867.

in practice).

We use our statistically-trained QA system (Yao et al., 2013) that recognizes the association between question type and expected answer types through various features. The QA system employs a linear chain Conditional Random Field (CRF) (Lafferty et al., 2001) and tags each token as either an answer (ANS) or not (O). This will be our off-the-shelf QA system, which recognizes the association between question type and expected answer types through various features based on e.g., part-of-speech tagging (POS) and named entity recognition (NER).

With weights optimized by CRF training (Table 1), we can learn how answer features are correlated with question features. These features, whose weights are optimized by the CRF training, directly reflect what the most important answer types associated with each question type are. For instance, line 2 in Table 1 says that if there is a `when` question, and the current token's NER label is DATE, then it is likely that this token is tagged as ANS. IR can easily make use of this knowledge: for a `when` question, IR retrieves sentences with tokens labeled as DATE by NER, or POS tagged as CD. The only extra processing is to pre-tag and index the text with POS and NER labels. The analyzing power of discriminative answer features for IR comes *for free* from a trained QA system. Unlike predictive annotation, statistical evidence determines the best answer features given the question, with no manual pattern or templates needed.

To compare again predictive annotation with our approach: predictive annotation works in a *forward* mode, downstream QA is tailored for upstream IR, i.e., QA works on whatever IR retrieves. Our method works in reverse (*backward*): downstream QA dictates upstream IR, i.e., IR retrieves what QA wants. Moreover, our approach extends easily beyond fixed *answer types* such as named entities: we are already using POS tags as a demonstration. We can potentially use any helpful *answer features* in retrieval. For instance, if the QA system learns that `in order to` is highly correlated with `why` question through lexicalized features, or some certain dependency relations are helpful in answering questions with specific structures, then it is natural and easy for the IR component to incorporate them.

There is also a distinction between our method and the technique of *learning to rank* applied in QA (Bilotti et al., 2010; Agarwal et al., 2012). Our method is a *QA-driven* approach that provides supervision for IR from a learned QA model, while learning to rank is essentially an *IR-driven* approach: the supervision for IR comes from a labeled ranking list of retrieval results.

| **feature** | **label** | **weight** |
|---|---|---|
| qword=when\|$\text{POS}_0$=CD | ANS | 0.86 |
| qword=when\|$\text{NER}_0$=DATE | ANS | 0.79 |
| qword=when\|$\text{POS}_0$=CD | O | -0.74 |

Table 1: Learned weights for sampled features with respect to the label of *current* token (indexed by [0]) in a CRF. The larger the weight, the more "important" is this feature to help tag the current token with the corresponding label. For instance, line 1 says when answering a `when` question, and the POS of current token is CD (cardinal number), it is likely (large weight) that the token is tagged as ANS.

Overall, we make the following contributions:

- Our proposed method tightly integrates QA with IR and the reuse of analysis from QA does not put extra overhead on the IR queries. This QA-driven approach provides a holistic solution to the task of IR4QA.
- We learn statistical evidence about what the form of answers to different questions look like, rather than using manually authored templates. This provides great flexibility in using answer features in IR queries.

We give a full spectrum evaluation of all three stages of IR+QA: document retrieval, passage retrieval and answer extraction, to examine thoroughly the effectiveness of the method.[3] All of our code and datasets are publicly available.[4]

## 2 Background

Besides Predictive Annotation, our work is closest to structured retrieval, which covers techniques of dependency path mapping (Lin and Pantel, 2001; Cui et al., 2005; Kaisser, 2012), graph matching with Semantic Role Labeling (Shen and Lapata, 2007) and answer type checking (Pinchak et al., 2009), etc. Specifically, Bilotti et al. (2007) proposed indexing text with their semantic roles and named entities. Queries then include constraints of semantic roles and named entities for the predicate and its arguments in the question. Improvements in recall of answer-bearing sentences were shown over the bag-of-words baseline. Zhao and

[3]Rarely are all three aspects presented in concert (see §2).
[4]`http://code.google.com/p/jacana/`

Callan (2008) extended this work with approximate matching and smoothing. Most research uses parsing to assign deep structures. Compared to shallow (POS, NER) structured retrieval, deep structures need more processing power and smoothing, but might also be more precise. [5]

Most of the above (except Kaisser (2012)) only reported on IR or QA, but not both, assuming that improvement in one naturally improves the other. Bilotti and Nyberg (2008) challenged this assumption and called for tighter coupling between IR and QA. This paper is aimed at that challenge.

## 3 Method

Table 1 already shows some examples of features associating question types with answer types. We store the features and their learned weights from the trained model for IR usage.

We let the trained QA system guide the query formulation when performing coupled retrieval with Indri (Strohman et al., 2005), given a corpus already annotated with POS tags and NER labels. Then retrieval runs in four steps (Figure 1):

1. Question Analysis. The question analysis component from QA is reused here. In this implementation, the only information we have chosen to use from the question is the question word (e.g., `how`, `who`) and the lexical answer types (LAT) in case of what/which questions.
2. Answer Feature Selection. Given the question word, we select the 5 highest weighted features (e.g., POS[0]=CD for a `when` question).
3. Query Formulation. The original question is combined with the top features as the query.
4. Coupled Retrieval. Indri retrieves a ranked list of documents or passages.

As motivated in the introduction, this framework is aimed at providing the following benefits:

**Reuse of QA components on the IR side**. IR reuses both code for question analysis and top weighted features from QA.

**Statistical selection of answer features**. For instance, the NER tagger we used divides location into two categories: GPE (geo locations) and LOC (non-GPE ). Both of them are learned to be important to `where` questions.

**Error tolerance along the NLP pipeline.** IR and QA share the same processing pipeline. Systematic errors made by the processing tools are tolerated, in the sense that if the same preprocessing error is made on both the question and sentence, an answer may still be found. Take the previous `where` question, besides NER[0]=GPE and NER[0]=LOC, we also found oddly NER[0]=PERSON an important feature, due to that the NER tool sometimes mistakes PERSON for LOC. For instance, the volcano name `Mauna Loa` is labeled as a PERSON instead of a LOC. But since the importance of this feature is recognized by downstream QA, the upstream IR is still motivated to retrieve it.

Queries were lightly optimized using the following strategies:

**Query Weighting** In practice query words are weighted:

```
#weight(1.0 When 1.0 was 1.0 Alaska 1.0 purchased
α #max(#any:CD #any:DATE))
```

with a weight $\alpha$ for the answer types tuned via cross-validation.

Since NER and POS tags are not lexicalized they accumulate many more counts (i.e. term frequency) than individual words, thus we in general downweight by setting $\alpha < 1.0$, giving the expected answer types "enough say" but not "too much say":

**NER Types First** We found NER labels better indicators of expected answer types than POS tags. The reasons are two-fold: 1. In general POS tags are too coarse-grained in answer types than NER labels. E.g., NNP can answer `who` and `where` questions, but is not as precise as PERSON and GPE. 2. POS tags accumulate even more counts than NER labels, thus they need separate downweighting. *Learning* the interplay of these weights in a joint IR/QA model, is an interesting path for future work. If the top-weighted features are based on NER, then we do not include POS tags for that question. Otherwise POS tags are useful, for instance, in answering `how` questions.

**Unigram QA Model** The QA system uses *up to* trigram features (Table 1 shows examples of unigram and bigram features). Thus it is able to learn, for instance, that a POS sequence of IN CD NNS is likely an answer to a `when` question (such as: `in 5 years`). This requires that the IR queries

[5] Ogilvie (2010) showed in chapter 4.3 that keyword and named entities based retrieval actually outperformed SRL-based structured retrieval in MAP for the answer-bearing sentence retrieval task in their setting. In this paper we do not intend to re-invent another parse-based structure matching algorithm, but only use shallow structures to show the idea of coupling QA with IR; in the future this might be extended to incorporate "deeper" structure.

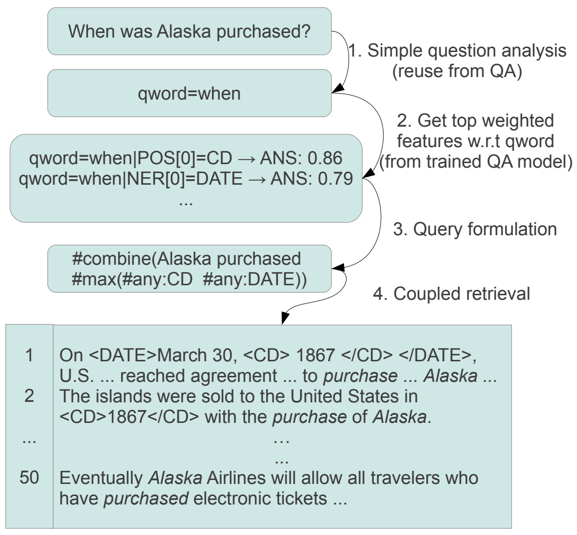
When was Alaska purchased?
1. Simple question analysis (reuse from QA)
qword=when
2. Get top weighted features w.r.t qword (from trained QA model)
qword=when|POS[0]=CD → ANS: 0.86
qword=when|NER[0]=DATE → ANS: 0.79
...
3. Query formulation
#combine(Alaska purchased #max(#any:CD #any:DATE))
4. Coupled retrieval
1 On <DATE>March 30, <CD> 1867 </CD> </DATE>, U.S. ... reached agreement ... to purchase ... Alaska ...
2 The islands were sold to the United States in <CD>1867</CD> with the purchase of Alaska.
... ...
50 Eventually Alaska Airlines will allow all travelers who have purchased electronic tickets ...

Figure 1: Coupled retrieval with queries directly constructed from highest weighted features of downstream QA. The retrieved and ranked list of sentences is POS and NER tagged, but only query-relevant tags are shown due to space limit. A bag-of-words retrieval approach would have the sentence shown above at rank 50 at its top position instead.

look for a consecutive IN CD NNS sequence. We drop this strict constraint (which may need further smoothing) and only use unigram features, not by simply extracting "good" unigram features from the trained model, but by re-training the model with only unigram features. In answer extraction, we still use up to trigram features. [6]

## 4 Experiments

We want to measure and compare the performance of the following retrieval techniques:

1. *uncoupled retrieval* with an off-the-shelf IR engine by using the question as query (baseline),
2. *QA-driven coupled retrieval* (proposed), and
3. *answer-bearing retrieval* by using both the question and known answer as query, only evaluated for answer extraction (upper bound),

at the three stages of question answering:

1. Document retrieval (for relevant docs from corpus), measured by Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR).
2. Passage retrieval (finding relevant sentences from the document), also by MAP and MRR.
3. Answer extraction, measured by $F_1$.

[6]This is because the weights of unigram to trigram features in a loglinear CRF model is a balanced consequence for maximization. A unigram feature might end up with lower weight because another trigram containing this unigram gets a higher weight. Then we would have missed this feature if we only used top unigram features. Thus we re-train the model with only unigram features to make sure weights are "assigned properly" among only unigram features.

| set | questions | | sentences | |
|---|---|---|---|---|
| | #all | #pos. | #all | #pos. |
| TRAIN | 2205 | 1756 (80%) | 22043 | 7637 (35%) |
| TEST$_{\text{gold}}$ | 99 | 88 (89%) | 990 | 368 (37%) |

Table 2: Statistics for AMT-collected data (total cost was around $800 for paying three Turkers per sentence). Positive questions are those with an answer found. Positive sentences are those bearing an answer.

All coupled and uncoupled queries are performed with Indri v5.3 (Strohman et al., 2005).

### 4.1 Data

**Test Set for IR and QA** The MIT109 test collection by Lin and Katz (2006) contains 109 questions from TREC 2002 and provides a near-exhaustive judgment of relevant documents for each question. We removed 10 questions that do not have an answer by matching the TREC answer patterns. Then we call this test set **MIT99**.

**Training Set for QA** We used Amazon Mechanical Turk to collect training data for the QA system by issuing answer-bearing queries for TREC1999-2003 questions. For the top 10 retrieved sentences for each question, three Turkers judged whether each sentence contained the answer. The inter-coder agreement rate was 0.81 (Krippendorff, 2004; Artstein and Poesio, 2008).

The 99 questions of MIT99 were extracted from the Turk collection as our TEST$_{\text{gold}}$ with the remaining as TRAIN, with statistics shown in Table 2. Note that only 88 questions out of MIT99 have an answer from the top 10 query results.

Finally both the training and test data were sentence-segmented and word-tokenized by NLTK (Bird and Loper, 2004), dependency-parsed by the Stanford Parser (Klein and Manning, 2003), and NER-tagged by the Illinois Named Entity Tagger (Ratinov and Roth, 2009) with an 18-label type set.

**Corpus Preprocessing for IR** The AQUAINT (LDC2002T31) corpus, on which the MIT99 questions are based, was processed in exactly the same manner as was the QA training set. But only sentence boundaries, POS tags and NER labels were kept as the annotation of the corpus.

### 4.2 Document and Passage Retrieval

We issued uncoupled queries consisting of question words, and QA-driven coupled queries consisting of both the question and expected answer types, then retrieved the top 1000 documents, and

| type | coupled | | uncoupled | |
|---|---|---|---|---|
| | MAP | MRR | MAP | MRR |
| document | **0.2524** | **0.4835** | 0.2110 | 0.4298 |
| sentence | **0.1375** | **0.2987** | 0.1200 | 0.2544 |

Table 3: Coupled vs. uncoupled document/sentence retrieval in MAP and MRR on MIT99. Significance level (Smucker et al., 2007) for both MAP: $p < 0.001$ and for both MRR: $p < 0.05$.

finally computed MAP and MRR against the gold-standard MIT99 per-document judgment.

To find the best weighting $\alpha$ for coupled retrieval, we used 5-fold cross-validation and finalized at $\alpha = 0.1$. Table 3 shows the results. Coupled retrieval outperforms (20% by MAP with $p < 0.001$ and 12% by MRR with $p < 0.01$) uncoupled retrieval significantly according to paired randomization test (Smucker et al., 2007).

For passage retrieval, we extracted relevant single sentences. Recall that MIT99 only contains document-level judgment. To generate a test set for sentence retrieval, we matched each sentence from relevant documents provided by MIT99 for each question against the TREC answer patterns.

We found no significant difference between retrieving sentences from the documents returned by document retrieval or directly from the corpus. Numbers of the latter are shown in Table 3. Still, coupled retrieval is significantly better by about 10% in MAP and 17% in MRR.

### 4.3 Answer Extraction

Lastly we sent the sentences to the downstream QA engine (trained on TRAIN) and computed $F_1$ per $K$ for the top $K$ retrieved sentences, [7] shown in Figure 2. The best $F_1$ with coupled sentence retrieval is 0.231, 20% better than $F_1$ of 0.192 with uncoupled retrieval, both at $K = 1$.

The two descending lines at the bottom reflect the fact that the majority-voting mechanism from the QA system was too simple: $F_1$ drops as $K$ increases. Thus we also computed $F_1$'s assuming perfect voting: a voting oracle that always selects the correct answer as long as the QA system produces one, thus the two ascending lines in the center of Figure 2. Still, $F_1$ with coupled retrieval is always better: reiterating the fact that coupled retrieval covers more answer-bearing sentences.

[7] Lin (2007), Zhang et al. (2007), and Kaisser (2012) also evaluated on MIT109. However their QA engines used web-based search engines, thus leading to results that are neither reproducible nor directly comparable with ours.

Finally, to find the upper bound for QA, we drew the two upper lines, testing on $\text{TEST}_{\text{gold}}$ described in Table 2. The test sentences were obtained with answer-bearing queries. This is assuming almost perfect IR. The gap between the top two and other lines signals more room for improvements for IR in terms of better coverage and better rank for answer-bearing sentences.

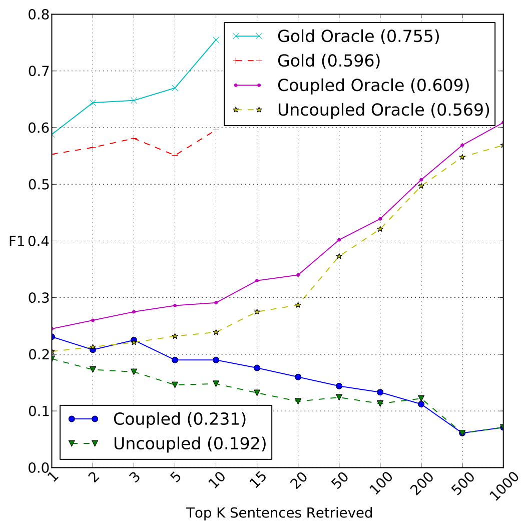

Figure 2: $F_1$ values for answer extraction on MIT99. Best $F_1$'s for each method are parenthesized in the legend. "Oracle" methods assumed perfect voting of answer candidates (a question is answered correctly if the system ever produced one correct answer for it). "Gold" was tested on $\text{TEST}_{\text{gold}}$.

## 5 Conclusion

We described a method to perform coupled information retrieval with a prior knowledge of the downstream QA system. Specifically, we coupled IR queries with automatically learned answer features from QA and observed significant improvements in document/passage retrieval and boosted $F_1$ in answer extraction. This method has the merits of not requiring hand-built question and answer templates and being flexible in incorporating various answer features automatically learned and optimized from the downstream QA system.

## Acknowledgement

We thank Vulcan Inc. for funding this work. We also thank Paul Ogilvie, James Mayfield, Paul McNamee, Jason Eisner and the three anonymous reviewers for insightful comments.

## References

Arvind Agarwal, Hema Raghavan, Karthik Subbian, Prem Melville, Richard D. Lawrence, David C. Gondek, and James Fan. 2012. Learning to rank for robust question answering. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, CIKM '12, pages 833–842, New York, NY, USA. ACM.

Ron Artstein and Massimo Poesio. 2008. Inter-Coder Agreement for Computational Linguistics. *Computational Linguistics*, 34(4):555–596.

M.W. Bilotti and E. Nyberg. 2008. Improving text retrieval precision and answer accuracy in question answering systems. In *Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering*, pages 1–8.

M.W. Bilotti, P. Ogilvie, J. Callan, and E. Nyberg. 2007. Structured retrieval for question answering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 351–358. ACM.

M.W. Bilotti, J. Elsas, J. Carbonell, and E. Nyberg. 2010. Rank learning for factoid question answering with linguistic and semantic constraints. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 459–468. ACM.

Steven Bird and Edward Loper. 2004. Nltk: The natural language toolkit. In *The Companion Volume to the Proceedings of 42st Annual Meeting of the Association for Computational Linguistics*, pages 214–217, Barcelona, Spain, July.

Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. 2005. Question answering passage retrieval using dependency relations. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 400–407, New York, NY, USA. ACM.

Mark A. Greenwood, editor. 2008. *Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering*. Coling 2008 Organizing Committee, Manchester, UK, August.

Michael Kaisser. 2012. Answer Sentence Retrieval by Matching Dependency Paths acquired from Question/Answer Sentence Pairs. In *EACL*, pages 88–98.

Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. In *In Proc. the 41st Annual Meeting of the Association for Computational Linguistics*.

Klaus H. Krippendorff. 2004. *Content Analysis: An Introduction to Its Methodology*. Sage Publications, Inc, 2nd edition.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

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

D. Lin and P. Pantel. 2001. Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(4):343–360.

Jimmy Lin. 2007. An exploration of the principles underlying redundancy-based factoid question answering. *ACM Trans. Inf. Syst.*, 25(2), April.

P. Ogilvie. 2010. *Retrieval using Document Structure and Annotations*. Ph.D. thesis, Carnegie Mellon University.

Christopher Pinchak, Davood Rafiei, and Dekang Lin. 2009. Answer typing for information retrieval. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 1955–1958, New York, NY, USA. ACM.

John Prager, Eric Brown, Anni Coden, and Dragomir Radev. 2000. Question-answering by predictive annotation. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '00, pages 184–191, New York, NY, USA. ACM.

J. Prager, J. Chu-Carroll, E. Brown, and K. Czuba. 2006. Question answering by predictive annotation. *Advances in Open Domain Question Answering*, pages 307–347.

L. Ratinov and D. Roth. 2009. Design challenges and misconceptions in named entity recognition. In *CoNLL*, 6.

Tetsuya Sakai, Hideki Shima, Noriko Kando, Ruihua Song, Chuan-Jie Lin, Teruko Mitamura, Miho Sugimito, and Cheng-Wei Lee. 2010. Overview of the ntcir-7 aclia ir4qa task. In *Proceedings of NTCIR-8 Workshop Meeting*, Tokyo, Japan.

D. Shen and M. Lapata. 2007. Using semantic roles to improve question answering. In *Proceedings of EMNLP-CoNLL*, pages 12–21.

M.D. Smucker, J. Allan, and B. Carterette. 2007. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 623–632. ACM.

T. Strohman, D. Metzler, H. Turtle, and W.B. Croft. 2005. Indri: A language model-based search engine for complex queries. In *Proceedings of the International Conference on Intelligent Analysis*, volume 2, pages 2–6. Citeseer.

Xuchen Yao, Benjamin Van Durme, Peter Clark, and Chris Callison-Burch. 2013. Answer Extraction as Sequence Tagging with Tree Edit Distance. In *Proceedings of NAACL 2013*.

Xian Zhang, Yu Hao, Xiaoyan Zhu, Ming Li, and David R. Cheriton. 2007. Information distance from a question to an answer. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 874–883, New York, NY, USA. ACM.

L. Zhao and J. Callan. 2008. A generative retrieval model for structured documents. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 1163–1172. ACM.