Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Philadelphia, July 2002, pp. 180-187.
Association for Computational Linguistics.

A Hybrid Approach to Natural Language Web Search

Jennifer Chu-Carroll, John Prager, Yael Ravin and **Christian Cesar**
IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598, U.S.A.
{jencc,jprager,ravin,cesar}@us.ibm.com

Abstract

We describe a hybrid approach to improving search performance by providing a natural language front end to a traditional keyword-based search engine. The key component of the system is iterative query formulation and retrieval, in which one or more queries are automatically formulated from the user's question, issued to the search engine, and the results accumulated to form the hit list. New queries are generated by relaxing previously-issued queries using transformation rules, applied in an order obtained by reinforcement learning. This statistical component is augmented by a knowledge-driven hub-page identifier that retrieves a hub-page for the most salient noun phrase in the question, if possible. Evaluation on an unseen test set over the www.ibm.com public website with 1.3 million webpages shows that both components make substantial contribution to improving search performance, achieving a combined 137% relative improvement in the number of questions correctly answered, compared to a baseline of keyword queries consisting of two noun phrases.

1 Introduction

Keyword-based search engines have been one of the most highly utilized internet tools in recent years. Nevertheless, search performance remains unsatisfactory at most e-commerce sites (Hagen et al., 2000). Librarians and search professionals have traditionally favored Boolean keyword search systems, which, when successful, return a small set of relevant hits. However, the success of these systems critically depends on the choice of the right keywords and the appropriate Boolean operators. As the population of search engine users has grown beyond a small dedicated search professional community and as these new users are less familiar with the contents they are searching, it has become harder for them to formulate successful keyword queries. To improve search performance, one can improve search engine accuracy with respect to fixed keyword queries, or provide the search engine with *better* queries, those more likely to retrieve good results. While there is much on-going work in the IR community on the former topic, we have taken the latter approach by providing a natural language search interface and automatically generating keyword queries that utilize advanced search features typically unused by end users. We believe that natural language questions are easier for users to construct than keyword queries, thus shifting the burden of optimal query formulation from the user to the system. Such questions also eliminate much of the ambiguity of keyword queries that often leads to poor results. Furthermore, the methodology we describe may be applied to different search engines with only minor modification.

To transform natural language input into a search query, the system must identify information pertinent for search and utilize it to formulate keyword queries likely to retrieve relevant answers. We describe and evaluate a hybrid system, RISQUE, that adopts an iterative approach to query formulation and retrieval for search on the www.ibm.com public website with 1.3 million webpages. RISQUE may issue multiple queries per question, where a new query is generated by relaxing a previously issued query via transformation rule application, in an order obtained by reinforcement learning. In addition, RISQUE identifies a hub-page for the most salient noun phrase in the question, if possible, utilizing traditional knowledge-driven mechanisms. Evaluation on an unseen test set showed that both the machine-learned and knowledge-driven components made substantial contribution to improving RISQUE's performance, resulting in a combined 137% relative improvement in the number of questions correctly answered, compared to a baseline obtained by queries consisting of two noun phrases (2NP baseline).

2 Related Work

The popularity of natural language search is evidenced by the growing number of search engines, such as AskJeeves, Electric Knowledge, and Northern Light,[1] that offer such functionality. For most sites, we were only able to perform a cursory examination of their proprietary techniques. Adopting a similar approach as FAQFinder (Hammond et al., 1995), AskJeeves maintains a database of questions and webpages that provide answers to them. User questions are compared against those in the database, and links to webpages for the closest matches are returned. Similar to our approach, Electric Knowledge transforms a natural language question into a series of increasingly more general keyword queries (Bierner, 2001). However, their query formulation process utilizes hard-crafted regular expressions, while we adopt a more general machine learning approach for transformation rule application.

Our work is also closely related to question answering in the question analysis component (e.g., (Harabagui et al., 2001; Prager et al., 2000; Clarke et al., 2001; Ittycheriah et al., 2001)). In particular, Harabagui et al.(2001) also iteratively reformulate queries based partly on the search results. However, their mechanism for query reformulation is heuristic-based. We utilized machine learning to optimize the query formulation process.

[1]www.askjeeves.com, www.electricknowledge.com, and www.northernlight.com, respectively.

3 Data Analysis

To generate optimal keyword queries from natural language questions, we first analyzed a set of 502 questions related to the purchasing and support of ThinkPads (notebook computers) and their accessories, such as *"How do I set up hibernation for my ThinkPad?"* and *"Show me all p3 laptops."* Our analysis focused on three tasks. First, we attempted to identify an exhaustive set of *correct webpages* for each question, where a correct webpage is one that contains either an answer to the question or a hyperlink to such a page. Second, we manually formulated *successful* keyword queries from the question, i.e., queries which retrieved at least one correct webpage. Third, we attempted to discover general patterns in how the natural language questions may be transformed into successful keyword queries.

Our analysis eliminated 110 questions for which no correct webpage was found. Of the remaining 392 questions, we identified, on average, 4.37 correct webpages and 1.58 successful queries per question. We found that the characteristics of successful queries varied greatly. In the simplest case, a successful query may contain all the content bearing NPs in the question, such as **thinkpad AND "answering machine"** for *"Can I use my ThinkPad as an answering machine?"*[2] In the vast majority of cases, however, more complex transformations were applied to the question to result in a successful query. For instance, a successful query for *"How do I hook an external mouse to my laptop?"* is **(mouse OR mice) AND thinkpad AND +url:support**. In this case, the head noun *mouse* was inflected,[3] the premodifier *external* was dropped, *hook* was deleted, *laptop* was replaced by *thinkpad*, and a URL constraint was applied.

We observed that in our corpus, most successful queries can be derived by applying one or more transformation rules to the NPs and verbs in the questions. Table 1 shows the manually in-

[2]Our search engine (www.alltheweb.com) accepts a conjunction of terms (a word, quoted phrase, or disjunction of words/phrases), and inclusion/exclusion of text strings in the URL, such as **+url:support**.

[3]Many commercial search engines purposely do not inflect search words to avoid overgeneralization of queries.

Rule	Function
ConstrainURL	Apply URL constraints
RelaxNP	Relax phrase to conjunction of words
DropNP	Remove least salient NP
DropModifier	Remove premodifiers of nouns
DropVerb	Remove verb
ApplySynonyms	Add synonyms of NPs
Inflect	Inflect head nouns and verb

Table 1: Query Transformation Rules

duced commonly-used transformation rules based on our corpus analysis. Though the rules were quite straightforward to identify, the order in which they should be applied to yield optimal overall performance was non-intuitive. In fact, the best order we manually derived did not yield sufficient performance improvement over our baseline (see Section 7). We further hypothesize that the optimal rule application sequence may be dependent on question characteristics. For example, *DropVerb* may be a higher priority rule for *buy* questions than for *support* questions, since the verbs indicative of *buy* questions (typically *"buy"* or *"sell"*) are often absent in the target product pages. Therefore, we investigated a machine learning approach to automatically obtain the optimal rule application sequence.

4 A Reinforcement Learning Approach to Query Formulation

Our problem consists of obtaining an optimal strategy for choosing transformation rules to generate successful queries. A key feature of this problem is that feedback during training is often delayed, i.e., the positive effect of applying a rule may not be apparent until a successful query is constructed after the application of other rules. Thus, we adopt a reinforcement learning approach to obtain this optimal strategy.

4.1 Q Learning

We adopted the Q learning paradigm (Watkins, 1989; Mitchell, 1997) to model our problem as a set of possible *states*, S, and a set of *actions*, A, which can be performed to alter the current state. While in state $s \in S$ and performing action $a \in A$, the learner receives a reward $r(s,a)$, and advances to state $s' = \delta(s,a)$.

To learn an optimal control strategy that maximizes the cumulative reward over time, an evaluation function $Q(s,a)$ is defined as follows:

$$Q(s,a) \equiv r(s,a) + \gamma \max_{a'} Q(\delta(s,a),a') \quad (1)$$

In other words, $Q(s,a)$ is the immediate reward, $r(s,a)$, plus the discounted maximum future reward starting from the new state $\delta(s,a)$.

The Q learning algorithm iteratively selects an action and updates $\hat{Q}$, an estimate of Q, as follows:

$$\begin{aligned} \hat{Q}_n(s,a) \quad \leftarrow \quad & (1-\alpha_n)\hat{Q}_{n-1}(s,a) + \\ & \alpha_n(r(s,a) + \max_{a'}\hat{Q}_{n-1}(s',a')) \end{aligned} \quad (2)$$

where $s' = \delta(s,a)$ and α_n is inversely proportional to the number of times a state/action pair <*s*,*a*> has been visited up to the *n*th iteration of the algorithm.[4]

Once the system learns $\hat{Q}$, it can select from the possible actions in state *s* based on $\hat{Q}(s,a_i)$.

4.2 Query Formulation Using Q Learning

To formulate our problem in the Q learning paradigm, we represent a state as a 6-tuple, <**qtype**, **url_constraint**, **np_phrase**, **num_nps**, **num_modifiers**, **num_verbs**>, where:

- **qtype** is *buy* or *support* depending on question classification.
- **url_constraint** is *true* or *false*, and determines if manually predefined URL restrictions will be applied in the query.
- **np_phrase** is *true* or *false*, and determines whether each NP will be searched for as a phrase or a conjunction of words.
- **num_nps** is an integer between 1 and 3, and determines how many NPs will be included in the query.
- **num_modifiers** is an integer between 0 and 2, and indicates the maximum number of premodifiers in each NP.
- **num_verbs** is 0 or 1, and determines if the verb will be included in the query.

[4]Equation (2) modifies (1) by taking a decaying weighted average of the current $\hat{Q}$ value and the new value to guarantee convergence of $\hat{Q}$ in non-deterministic environments. We explain in the next section why our query formulation problem in the Q learning framework is non-deterministic.

This representation is chosen based on the rules identified in Section 3. The actions, A, include the first 5 actions in Table 1, and the "undo" counterpart for each action.[5] Except for **qtype**, which remains static for a question, each remaining element in the tuple can be altered by one of the 5 pairs of actions in a straightforward manner. The state, s, and the question, q, generate a unique keyword query which results in a hit list, $h(s, q)$. The hit list is evaluated for correctness, whose result is used to define the reward function as follows:

$$r(s,a) = \begin{cases} 1 & \text{if } h(s',q) \text{ contains at least one} \\ & \text{correct webpage} \\ 0 & \text{if } h(s',q) \text{ has no correct page \&} \\ & |h(s',q)| < 10 \\ -1 & \text{otherwise} \end{cases}$$

where $s' = \delta(s, a)$. Note that our system operates in a non-deterministic environment because the reward is dependent not only on s and a, but also on q.[6]

Having defined S, A, δ, and r, $\hat{Q}$ is determined by applying the Q learning algorithm, using the update function in (2), to our corpus of 392 questions. For each question, an initial state is randomly selected within the bounds of the question. The system then iteratively selects and applies actions, and updates $\hat{Q}$ until a successful query is generated or the maximum number of iterations is reached (in our implementation, 15). The training algorithm iterates over all questions in the training set and terminates when $\hat{Q}$ converges.

5 RISQUE: A Hybrid System for Natural Language Search

5.1 System Overview

In addition to motivating machine learning based query transformation as our central approach to natural language search, our analysis revealed the need for several other key system components. As shown in Figure 1, RISQUE adopts a hybrid architecture that combines the utility of traditional knowledge-based methods and statistical approaches. Given a question, RISQUE first performs question analysis by extracting pertinent information to be used in query formulation, such as the NPs, VPs, and question type, and then orders the NPs in terms of their relative salience. This information is then used for hit list construction by two modules. The first component is the *hub-page identifier*, which retrieves, if possible, a hub page for the most salient NP in the question. The second component is the Q learning based query formulation and retrieval module that iteratively generates queries via transformation rule application and issues them to the search engine. The results from both processes are combined and accumulated until *n* distinct hits are retrieved.

In addition to the above components, RISQUE employs an ontology for the ThinkPad domain, which consists of 1) a hierarchy of about 500 domain objects, 2) nearly 400 instances of relationships, such as *isa* and *accessory-of*, between objects, and 3) a synonym dictionary containing about 1000 synsets. The ontology was manually constructed and took approximately 2 person-months for coverage in the ThinkPad domain. It provides pertinent information to the question pre-processing and query formulation modules, which we will describe in the next sections.

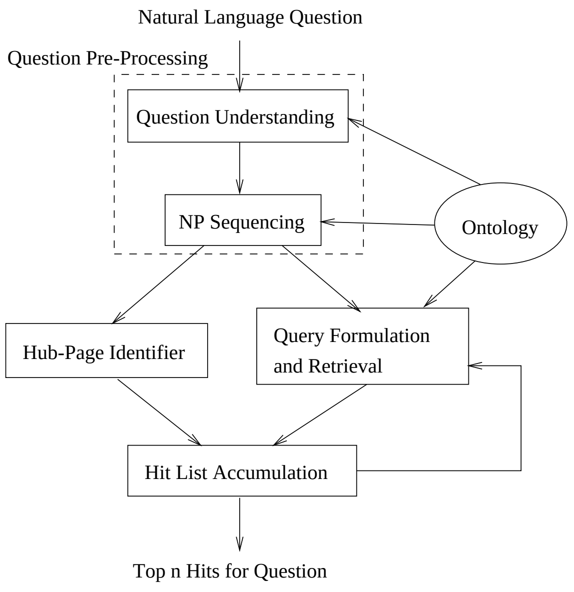

Figure 1: RISQUE Architecture

[5] Morphological and synonym expansions are applied at the outset, which was shown to result in better performance than optional application of those rules.

[6] It is theoretically possible to encode pertinent information in q in the state representation, thus making the environment deterministic. However, data sparseness problems associated with such a representation makes it impractical.

5.2 Question Pre-Processing

5.2.1 Question Understanding

RISQUE's question understanding component is based primarily on a rule-driven parser in the slot grammar framework (McCord, 1989). The resulting parse tree is first analyzed for NP/VP extraction. Each NP includes the head noun and up to two premodifiers, which covers most NPs in our domain. The NPs are further processed by a named-entity recognizer (Prager et al., 2000; Wacholder et al., 1997), with reference to domain-specific proper names in our ontology. Recognized compound terms, such as *"hard drive"*, are treated as single entities, rather than as head nouns (*"drive"*) with premodifiers (*"hard"*). This prevents part of the compound term from being dropped when the *DropModifier* transformation rule is applied.

The parse tree is also used to classify the question as *buy* or *support*. The classifier utilizes a set of rules based on lexical and part-of-speech information. For example, *"how"* tagged as a adverb (as in *"How do I ..."*) suggests a *support* question, while *"buy/sell"* used as a verb indicates a *buy* question. These rules were manually derived based on our training data.

5.2.2 NP Sequencing

Our analysis showed that when a successful query is to contain fewer NPs than in the question, it is not straightforward to determine which NPs to eliminate, as it requires both domain and content knowledge. However, we observed that less salient NPs are often removed first, where salience indicates the importance of the term in the search process. The relative salience of NPs in this context can, for the most part, be determined based on the ontological relationship between the NPs and knowledge about the website organization. For instance, if A is an *accessory-of* B, then A is more salient than B since, on our website, accessories typically have their own webpages with significantly more information than pages about, for instance, the ThinkPads with which they are compatible.

Our NP sequencer utilizes a rule-based reasoning mechanism to rank a set of NPs based on their relative salience, as determined by their relationship in the ontology.[7] Objects not present in the ontology are considered less important than those present. This process produces a list of NPs ranked in decreasing order of salience.

5.3 Hub-Page Identifier

As with most websites, the ThinkPad pages on www.ibm.com are organized hierarchically, with a dozen or so *hub-pages* that serve as good starting points for specific sub-topics, such as mobile accessories and personal printers. However, since these hub-pages are typically not content-rich, they often do not receive high scores from the search engine (over which we have no control). Thus, we developed a mechanism to explicitly retrieve these hub-pages when appropriate, and to combine its results with the outcome of the actual search process.

The hub-page identifier consists of a mapping from a subset of the named entities in the ontology to their corresponding hub-pages.[8] For each question, the hub-page identifier retrieves the hub-page for the most salient NP, if possible, which is presented as the first entry in the hit list.

5.4 Reinforcement Learning Based Query Formulation

This main component of RISQUE iteratively formulates queries, issues them to the search engine, and accumulates the results to construct the hit list. The query formulation process starts with the most constrained query, and each new query is a relaxation of a previously issued query, obtained by applying one or more transformation rules to the current query. The transformation rules are applied in the order obtained by the Q learning algorithm as described in Section 4.2.

The initial state of the query formulation process is as follows: **url_constraint** and **np_phrase** are set to *true*, while the other attributes are set to their respective maximum values based on the outcome of the question understanding process. This initial state represents the most constrained query possible for the given question, and allows for subsequent relaxation via the application of transformation rules.

[7]We are aware that factors involving deeper question understanding come into play in determining relative salience. We leave investigation of such features as future work.

[8]For reasons of robustness, we actually map a named entity to manually selected keywords which, when issued to the search engine, retrieves the desired hub-page as the first hit.

When a state s, is visited, a query is generated based on s and the question. The query terms are instantiated based on the values of **np_phrase**, **num_nps**, **num_modifiers**, and **num_verbs** in s and the question itself, while URL constraints may be applied based on **url_constraint** and **qtype**. Finally, synonyms expansion is applied based on the synonym dictionary in the ontology, while morphological expansion is performed on all NPs using a rule-based inflection procedure.

After a query is issued, the search results are incorporated into the hit list, and duplicate hits are removed. A transformation rule $a_{max} = argmax_a \hat{Q}(s, a)$ is applied to yield the new state. $\hat{Q}(s, a_{max})$ is then decreased to remove it from further consideration. This iterative process continues until the hit list contains 10 or more elements.

6 Example

To illustrate RISQUE's end-to-end operation, consider the question *"Do you sell a USB hub for a ThinkPad?"*

The question is classified as a *buy* question, given presence of the verb *sell*. In addition, two NPs are identified:

NP1: head = USB hub
NP2: head = ThinkPad

Note that *"USB hub"* is identified as a compound noun by our named-entity recognizer. The NP sequencer determines that USB hub is more salient than ThinkPad since the former is an accessory of the latter.

The hub-page identifier finds the networking devices hub-page for *USB hub*, presented as the first entry in the hit list in Figure 2, where correct webpages are boldfaced.

Next, RISQUE invokes its iterative query formulation process to populate the remaining hit list entries. The initial state is <**qtype** = *buy*, **url_constraint** = *true*, **np_phrase** = *true*, **num_nps** = *2*, **num_modifiers** = *0*, **num_verbs** = *0*>. This state generates the query shown as "Query 2" in Figure 2, which results in 6 hits, of which 4 are correct.

RISQUE selects the optimal transformation rule for the current state, which is *ReinstateModifier*. Since neither NP has any modifier, a second rule, *RelaxNP* is attempted, which resulted in a new query that did not retrieve any previously unseen hits. Next, RISQUE selects *ConstrainNP* and *RelaxURL*, resulting in the query shown as "Query 3" in Figure 2.[9] Note that relaxing the URL constraint results in retrieval of USB hub support pages.

7 Performance Evaluation and Analysis

We evaluated RISQUE's performance on 102 questions in the ThinkPad domain previously unseen to both RISQUE's knowledge-based and statistical components. The top 10 hits returned by RISQUE for each question were manually evaluated for correctness as in Section 3. A 2NP baseline was obtained by extracting up to two most salient NPs in each question, searching for the conjunction of all words in the NPs, and manually evaluating the 10 top hits returned.

We selected the 2NP baseline based on statistics of keyword query logs on our website, which show that 98.2% of all queries contain 4 keywords or less. Furthermore, most three and four-word queries contain two distinct noun phrases, such as "visualage for java" and "application framework for e-business". Thus, we use the 2NP baseline as an approximation of user keyword search performance for our natural language questions.[10]

We compared RISQUE's performance to the baseline using three metrics:[11]

1. **Total correct**: number of questions for which at least one correct webpage is retrieved.

2. **Average correct**: average number of correct webpages retrieved per question.

3. **Average rank**: average rank of the first correct webpage in the hit list.

The evaluation results are summarized in Table 2, where the first and last rows show the 2NP baseline and RISQUE's performance, respectively. The

[9]A set of negative URL constraints is applied at all times to best exclude parts of the website unrelated to ThinkPads.

[10]This is likely too high an estimate for current keyword search performance, since the majority of user queries employ only one noun phrase.

[11]We chose not to evaluate our results using the traditional IR recall measure because for our task, it is often sufficient to return one page that answers the question instead of attempting to retrieve all relevant pages.

Question: Do you have a USB hub for a ThinkPad?
Query 1: hub-page identifier
1 ***Communications, Networking, Input/Output Devices ...***

Query 2: (thinkpad thinkpads laptop laptops notebook notebooks) (“usb hub” “usb hubs”)
+url: (commerce accessories proven products thinkpad)
-url: research press rs6000 eserver ...
2 ***Mobile Accessories ...***
3 ***4-Port Ultra-Mini USB Hub***
4 *ThinkPad TransNote Port Extender*
5 ***Belkin ExpressBus 7-Port USB Hub***
6 *Portable Drive Bay 2000*
7 ***Belkin BusStation 7 Port Modular USB Hub***

Query 3: (thinkpad thinkpads laptop laptops notebook notebooks) (“usb hub” “usb hubs”)
-url: research press rs6000 eserver ...
8 *Java technology for the universal serial bus*
9 *Multiport USB Hub - Printer compatibility list*
10 *Multi-Port USB Hub - Overview*

Figure 2: RISQUE Results for Sample Question

	Total Correct	Average Correct	Average Rank
2NPs	30	0.63	4.0
Fixed Order	45	1.24	2.71
RISQUE w/o hub identifier	56	1.67	2.25
RISQUE	71	1.87	2.11

Table 2: RISQUE Evaluation Results

results show that RISQUE correctly answered 71 questions, a 137% relative improvement over the baseline. Furthermore, the average number of correct answers found nearly tripled, while, on average, the rank of the first correct answer improved from 4.0 to 2.11.

Table 2 further shows performance figures that evaluate the individual contribution of RISQUE’s two main components, the hub-page identifier and the iterative query formulation module. Comparison between the last two rows in Table 2 shows the effectiveness of the hub-page identifier, which substantially increased the number of questions correctly answered, but resulted in only minor gain using the other two performance metrics. To assess the effectiveness of the query formulation module, we used the best manually-derived rule application sequence obtained in Section 3. We compared these *fixed order* performance figures to those for *RISQUE w/o hub identifier* which shows that applying Q learning to derive an optimal state-dependent rule application order resulted in fairly substantial improvement using all three metrics.

Maxq	10	5	3	2	1
RISQUE	5.07	4.47	2.89	1.98	1
RISQUE w/o hpi	4.26	3.86	2.80	1.93	1

Table 3: Average Queries Issued for Select *Maxq*s

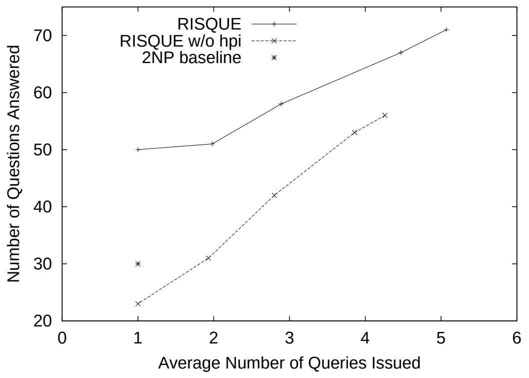

Figure 3: # Queries Issued vs. System Performance

One of RISQUE’s parameters, *maxq*, specifies the maximum number of distinct queries it can issue to the search engine for each question. Table 3 shows the average number of queries actually issued for select values of *maxq*.[12] Figure 3 shows how performance degrades when fewer queries are issued as a result of lowering *maxq* for both RISQUE and RISQUE without the hub-page identifier. It shows that, with the exception of RISQUE’s performance

[12]*Maxq* is 10 for the results in Table 2.

when only one query is issued,[13] the number of questions answered have a near-linear relationship with the number of queries issued for both systems. Notice that without the hub-page identifier, RISQUE's performance when issuing an average of 1.93 queries per question is nearly the same as that of the 2NP baseline, while it performs worse than the baseline when issuing only one query per question. This is because our iterative query formulation process intentionally begins with the most constrained query, resulting in an empty hit list in many cases.

8 Conclusions and Future Work

In this paper, we described and evaluated RISQUE, a hybrid system for performing natural language search on a large company public website. RISQUE utilizes a two-pronged approach to generate hit lists for answering natural language questions. On the one hand, RISQUE employs a hub-page identifier to retrieve, when possible, a hub-page for the most salient NP in the question. On the other hand, RISQUE adopts a statistical iterative query formulation and retrieval mechanism that generates new queries by applying transformation rules to previously-issued queries. By employing these two components in parallel, RISQUE takes advantages of both knowledge-driven and machine learning approaches, and achieves an overall 137% relative improvement in the number of questions correctly answered on an unseen test set, compared to a baseline of 2NP keyword queries.

In our current work, we are focusing on expanding system coverage to other domains. In particular, we plan to investigate semi-automatic methods for extracting ontological knowledge from existing webpages and databases.

Acknowledgments

We would like to thank Dave Ferrucci and Nanda Kambhatla for helpful discussions, Ruchi Kalra and Jerry Cwiklik for data preparation, Eric Brown and the anonymous reviewers for helpful comments on an earlier draft of this paper, as well as Mike Moran and Alex Holt for providing technical assistance with ibm.com search.

[13] In most cases, this query is issued by the hub-page identifier, which has a higher success rate than the queries generated by the query formulation module.

References

G. Bierner. 2001. Alternative phrases and natural language information retrieval. In *Proc. 39th ACL*, pages 58–65.

C. Clarke, G. Cormack, and T. Lynam. 2001. Exploiting redundancy in question answering. In *Proc. 24th SIGIR*, pages 358–365.

P. Hagen, H. Manning, and Y. Paul. 2000. Must search stink? *The Forrester Report*, June.

K. Hammond, R. Burke, C. Martin, and S. Lytinen. 1995. Faq finder: A case-based approach to knowledge navigation. In *AAAI Spring Symposium on Information Gathering in Heterogeneous Environments*, pages 69–73.

S. Harabagui, D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Girju, V. Rus, and P. Morarescu. 2001. The role of lexico-semantic feedback in open-domain textual question-answering. In *Proc. 39th ACL*, pages 274–281.

A. Ittycheriah, M. Franz, W-J. Zhu, and A. Ratnaparkhi. 2001. Question answering using maximum entropy components. In *Proc. 2nd NAACL*, pages 33–39.

M. McCord. 1989. Slot grammar: A system for simpler construction of practical natural language grammars. In *Natural Language and Logic*, pages 118–145.

T. Mitchell. 1997. *Machine Learning*. McGraw Hill.

J. Prager, E. Brown, A. Coden, and D. Radev. 2000. Question-answering by predictive annotation. In *Proc. 23rd SIGIR*, pages 184–191.

N. Wacholder, Y. Ravin, and M. Choi. 1997. Disambiguation of proper names in text. In *Proc. 5th ANLP*.

C. Watkins. 1989. *Learning from Delayed Rewards*. Ph.D. thesis, King's College.