A Two-Stage Approach to Retrieving Answers for How-To Questions

Ling Yin
CMIS, University of Brighton,
Brighton, BN2 4GJ, United Kingdom
Y.Ling@brighton.ac.uk

Abstract

This paper addresses the problem of automatically retrieving answers for how-to questions, focusing on those that inquire about the *procedure* for achieving a specific goal. For such questions, typical information retrieval methods, based on key word matching, are better suited to detecting the content of the goal (e.g., 'installing a Windows XP server') than the general nature of the desired information (i.e., procedural, a series of steps for achieving this goal). We suggest dividing the process of retrieving answers for such questions into two stages, with each stage focusing on modeling one aspect of a how-to question. We compare the two-stage approach with two alternative approaches: a baseline approach that only uses the content of the goal to retrieve relevant documents and another approach that explores the potential of automatic query expansion. The result of the experiment shows that the two-stage approach significantly outperforms the baseline but achieves similar result with the systems using automatic query expansion techniques. We analyze the reason and also present some future work.

1 Introduction

How-To questions constitute a large proportion of questions on the Web. Many how-to questions inquire about the *procedure* for achieving a specific goal. For such questions, typical information retrieval (IR) methods, based on key word matching, are better suited to detecting the content of the goal (e.g., installing a Windows XP server) than the general nature of the desired information (i.e., procedural, a series of steps for achieving this goal). The reasons are given as below.

First, documents that describe a procedure often do not contain the word 'procedure' itself, but we are able to abstract the concept 'procedure' from cues such as 'first', 'next' and 'then', all of which indicate sequential relationships between actions. Secondly, We expect that the word 'procedure' or the phrase 'how to' will occur in a much broader context than the words in the goal. In other words, a document that contains the words in the goal is more likely to be relevant than a document that contains the word 'procedure' or the phrase 'how to'. Without noticing this difference, treating the two parts equally in the retrieving process will get many noisy documents.

Many information requests seem to show such a structure, with one part identifying a specific topic and another part constraining the kind of information required about this topic (Yin and Power, 2005). The second part is often omitted when selecting retrieval terms from the request to construct an effective query for an IR system, such as in Picard (1999).

The first point given above suggests that using cues such as 'first' and 'next' to expand the initial query may help in retrieving more relevant documents. Expansion terms can be generated automatically by query expansion techniques. The typical process is: (1) use the initial query to retrieve documents (referred to as the first round of retrieval); (2) consider a few top ranked documents as relevant and the rest irrelevant; (3) compare the relevant set with the irrelevant set to extract a list of most distinctive terms; (4) use the extracted terms to retrieve documents (referred to as the second round of retrieval).

However, query expansion may not constitute a good solution, because its effectiveness largely

depends on the quality of the few top ranked documents retrieved in the first round when the aforementioned two problems are not yet tackled.

Our solution is to divide the process of retrieving answers for such questions into two stages: (1) use typical IR approaches for retrieving documents that are relevant to the specific goal; (2) use a text categorization approach to re-rank the retrieved documents according to the proportion of procedural text they contain. By 'procedural text' we refer to ordered lists of steps, which are very common in some instructional genres such as online manuals.

In this report, we will briefly introduce the text categorization approach (details are presented in (Yin and Power, 2006)) and will explain in more concrete terms how it is integrated into the two-stage architecture proposed above. We will compare the performance of our two-stage architecture with a baseline system that uses only the content of the goal to retrieve relevant documents (equivalent to the first stage in the two-stage architecture). We will also compare the two-stage approach with systems that applies automatic query expansion techniques.

This paper is organized as follows. Section 2 introduces some relevant work in IR and question answering (QA). Section 3 talks about the text categorization approach for ranking procedural documents, covering issues such as the features used, the training corpus, the design of a classification model as well as some experiments for evaluation. Section 4 talks about integrating the text categorizer into the two-stage architecture and presents some experiments on retrieving relevant documents for how-to questions. Section 5 provides a short summary and presents some future work.

2 Related Work

The idea of applying text categorization technology to help information retrieval is not new. In particular, text categorization techniques are widely adopted to filter a document source according to specific information needs. For example, Stricker et al. (2000) experiment on several news resources to find news addressing specific topics. They present a method for automatically generating "discriminant terms" (Stricker et al., 2000) for each topic that are then used as features to train a neural network classifier. Compared to these approaches, the novelty of our study lies in the idea that an information request consists of two different parts that should be retrieved in different ways and the whole retrieval process should adopt a two-stage architecture.

A research area that is closely related to IR is question answering (QA), the differences being a) the input of a QA system is a question rather than a few key words; b) a QA system aims to extract answers to a question rather than retrieving relevant documents only. Most QA systems do adopt a two-stage architecture (if not consider the initial question analysis stage), i.e., perform IR with a few content words extracted from the query to locate documents likely to contain an answer and then use information extraction (IE) to find the text snippets that match the question type (Hovy et al., 2001; Elworthy, 2000). However, most question answering systems target factoid questions – the research of non-factoid questions started only a few years ago but limited to several kinds, such as definitional questions (Xu et al., 2003) and questions asking for biographies (Tsur et al., 2004).

Only a few studies have addressed procedural questions. Murdok and Croft (2002) distinguish between "task-oriented questions" (i.e., ask about a process) and "fact-oriented questions" (i.e., ask about a fact) and present a method to automatically classify questions into these two categories. Following this work, Kelly et al. (2002) explore the difference between documents that contain relevant information to the two different types of questions. They conclude, "lists and FAQs occur in more documents judged relevant to task-oriented questions than those judged relevant to fact-oriented questions" (Kelly et al., 2002: 645) and suggest, "retrieval techniques specific to each type of question should be considered" (Kelly et al., 2002: 647). Schwitter et al. (2004) present a method to extract answers from technical documentations for How-questions. To identify answers, they match the logical form of a sentence against that of the question and also explore the typographical conventions in technical domains. The work that most resembles ours is Takechi et al. (2003), which uses word n-grams to classify (as procedural or non-procedural) list passages extracted using HTML tags. Our approach, however, applies to whole documents, the aim being to measure the degree of *procedurality* — i.e., the proportion of procedural text they contain.

3 Ranking Procedural Texts

Three essential elements of a text categorization approach are the features used to represent the document, the training corpus and the machine learning method, which will be described in section 3.1, 3.2 and 3.3 respectively. Section 3.4 presents experiments on applying the learned model to rank documents in a small test set.

3.1 Feature Selection and Document Representation

Linguistic Features and Cue Phrases

We targeted six procedural elements: actions, times, sequence, conditionals, preconditions, and purposes. These elements can be recognized using linguistic features or cue phrases. For example, an action is often conveyed by an imperative; a precondition can be expressed by the cue phrase 'only if'. We used all the syntactic and morphological tags defined in Connexor's syntax analyzer[1]. There are some redundant tags in this set. For example, both the syntactic tag '@INFMARK>' and the morphological tag 'INFMARK>' refer to the infinitive marker 'to' and therefore always occur together at the same time. We calculated the Pearson's product-moment correlation coefficient (r) (Weisstein, 1999) between any two tags based on their occurrences in sentences of the whole training set. We removed one in each pair of strongly correlated tags and finally got 34 syntactic tags and 34 morphological tags. We also handcrafted a list of relevant cue phrases (44), which were extracted from documents by using the Flex tool[2] for pattern matching. Some sample cue phrases and the matching patterns are shown in table 1.

Procedural Element	Cue Phrase	Pattern
Precondition	'only if'	[Oo]nly[[:space:]]if[[:space:]]
Purpose	'so that'	[sS]o[[:space:]]that[[:space:]]
Condition	'as long as'	([Aa]s) [[:space:]]long[[:space:]]as[[:space:]]
Sequence	'first'	[fF]irst [[:space:][:punct:]]
Time	'now'	[nN]ow[[:space:][:punct:]]

Table 1. Sample cue phrases and matching patterns.

Modeling Inter-Sentential Feature Co-occurrence

Some cue phrases are ambiguous and therefore cannot reliably suggest a procedural element. For example, the cue phrase 'first' can be used to represent a ranking order or a spatial relationship as well as a sequential order. However, it is more likely to represent a sequential order between actions if there is also an imperative in the same sentence. Indeed, sentences that contain both an ordinal number and an imperative are very frequent in procedural texts. We compared between the procedural training set and the non-procedural training set to extract distinctive feature co-occurrence patterns, each of which has only 2 features. The formulae used to rank patterns with regard to their distinctiveness can be found in (Yin and Power, 2006).

Document Representation

Each document was represented as a vector $\vec{d}_j = \{x_{1j}, x_{2j}, ..., x_{Nj}\}$, where x_{ij} represents the number of sentences in the document that contains a particular feature normalized by the document length. We compare the effectiveness of using individual features (x_{ij} refers to either a single linguistic feature or a cue phrases) and of using feature co-occurrence patterns (x_{ij} refers to a feature co-occurrence pattern).

3.2 Corpus Preparation

Pagewise [3] provides a list of subject-matter domains, ranging from household issues to arts and entertainment. We downloaded 1536 documents from this website (referred to hereafter as the Pagewise collection). We then used some simple heuristics to select documents from this set to build the initial training corpus. Specifically, to build the procedural set we chose documents with titles containing key phrases 'how to' and 'how can I' (209 web documents); to build the non-procedural set, we chose documents which did not include these phrases in their titles, and which also had no phrases like 'procedure' and 'recipe' within the body of the text (208 web documents).

Samples drawn randomly from the procedural set (25) and non-procedural set (28) were submitted to two human judges, who assigned procedurality scores from 1 (meaning no procedural text at all) to 5 (meaning over 90% procedural text). The Kendall tau-b agreement (Kendall, 1979) between the two rankings was 0.821. Overall, the average scores for the procedural and non-procedural samples were 3.15 and 1.38. We used these 53 sample documents as part of the test set and the

[1] Refer to http://www.connexor.com/

[2] Refer to http://www.gnu.org/software/flex/flex.html

[3] Refer to http://www.essortment.com

remaining documents as the initial training set (184 procedural and 180 non-procedural).

This initial training corpus is far from ideal: first, it is small in size; a more serious problem is that many positive training examples do not contain a major proportion of procedural text. In our experiments, we used this initial training set to bootstrap a larger training set.

3.3 Learning Method

Although shown to be not so effective in some previous studies (Yang, 1999; Yang and Liu, 1999), Naive Bayes classifier is one of the most commonly-used classifiers for text categorization. Here we introduce a model adapted from the Naive Bayes classifier from the weka-3-4 package (Witten and Frank, 2000).

The Naive Bayes classifier scores a document $\vec{d}_j$ according to whether it is a typical member of its set — i.e., the probability of randomly picking up a document like it from the procedural class ($p(\vec{d}_j \mid C = procedural)$). This probability is estimated from the training corpus. As mentioned before, the average procedural score of the procedural training set is low. Therefore, there is obviously a danger that a true procedural document will be ranked lower than a document that contains less procedural texts when using this training set to train a Naive Bayes classifier. Although our procedural training set is not representative of the procedural class, by comparing it with the non-procedural training set, we are able to tell the difference between procedural documents and non-procedural documents. We adapted the Naive Bayes classifier to focus on modeling the difference between the two classes. For example, if the procedural training set has a higher average value on feature Xi than the non-procedural training set, we inferred that a document with a higher feature value on Xi should be scored higher. To reflect this rule, we scored a document $\vec{d}_j$ by the probability of picking a document with a lower feature value from the procedural class (i.e., $p(X_i < x_{ij} \mid C = procedural)$). Again this probability is estimated from the training set.

The new model will be referred to hereafter as the Adapted Naive Bayes classifier. The details of this new model can be found in (Yin and Power, 2006).

3.4 Experiments on Ranking Procedural Texts

There are two sources from which we compiled the training and testing corpora: the Pagewise collection and the SPIRIT collection. The SPIRIT collection contains a terabyte of HTML that are crawled from the web starting from an initial seed set of a few thousands universities and other educational organizations (Clarke et al., 1998).

Our test set contained 103 documents, including the 53 documents that were sampled previously and then separated from the initial training corpus, another 30 documents randomly chosen from the Pagewise collection and 20 documents chosen from the SPIRIT collection. We asked two human subjects to score the procedurality for these documents, following the same instruction described in section 3.2. The correlation coefficient (Kendall tau-b) between the two rankings was 0.725, which is the upper bound of the performance of the classifiers.

We first used the initial training corpus to bootstrap a larger training set (378 procedural documents and 608 non-procedural documents), which was then used to select distinctive feature co-occurrence patterns and to train different classifiers. We compared the Adapted Naive Bayes classifier with the Naive Bayes classifier and three other classifiers, including Maximum Entropy (ME) [4], Alternating Decision Tree (ADTree) (Freund and Mason, 1999) and Linear Regression (Witten and Frank, 2000).

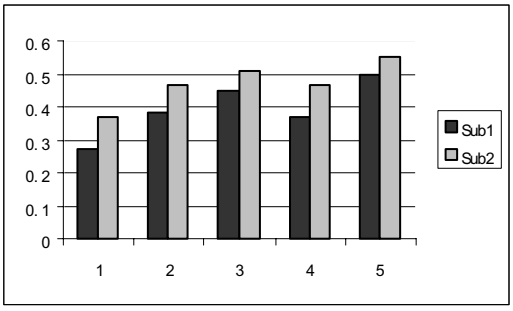

Figure 1. Ranking results using individual features: 1 refers to Adapted Naive Bayes, 2 refers to Naive Bayes, 3 refers to ME, 4 refers to ADTree and 5 refers to Linear Regression.

Ranking Method	Agreement with Subject 1	Agreement with Subject 2	Average
Adapted Naive Bayes	0.270841	0.367515	0.319178
Naive Bayes	0.381921	0.464577	0.423249
ME	0.446283	0.510926	0.478605

[4] Refer to http://homepages.inf.ed.ac.uk/s0450736/maxent.html

ADTree	0.371988	0.463966	0.417977
Linear Regression	0.497395	0.551597	0.524496

Table 2. Ranking results using individual features.

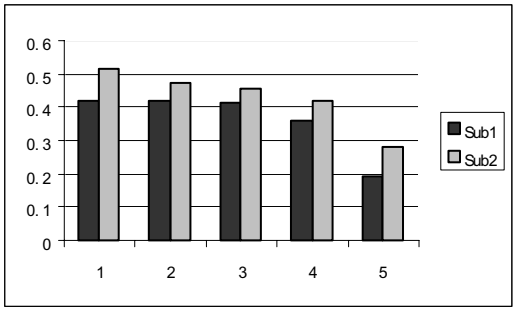

Figure 2. Ranking results using feature co-occurrence patterns: 1 refers to Adapted Naive Bayes, 2 refers to Naive Bayes, 3 refers to ME, 4 refers to ADTree and 5 refers to Linear Regression.

Ranking Method	Agreement with Subject 1	Agreement with Subject 2	Average
Adapted Naive Bayes	0.420423	0.513336	0.466880
Naive Bayes	0.420866	0.475514	0.44819
ME	0.414184	0.455482	0.434833
ADTree	0.358095	0.422987	0.390541
Linear Regression	0.190609	0.279472	0.235041

Table 3. Ranking results using feature co-occurrence patterns.

Figure 1 and table 2 show the Kendall tau-b coefficient between human subjects' ranking results and the trained classifiers' ranking results of the test set when using individual features (112); Figure 2 and table 3 show the Kendall tau-b coefficient when using feature co-occurrence patterns (813).

As we can see from the above figures, when using individual features, Linear Regression achieved the best result, Adapted Naive Bayes performed the worst, Naive Bayes, ME and ADTree were in the middle; however, when using feature co-occurrence patterns, the order almost reversed, i.e., Adapted Naive Bayes performed the best and Linear Regression the worst. Detailed analysis of the result is beyond the scope of this paper. The best model gained by using feature co-ocurrence patterns (i.e., Adapted Naive Bayes classifier) and by using individual features (i.e., Linear Regression classification model) will be used for further experiments on the two-stage architecture.

4 Retrieving Relevant Documents for How-To Questions

In this section we will describe the experiments on retrieving relevant documents for how-to questions by applying different approaches mentioned in the introduction section.

4.1 Experiment Setup

We randomly chose 60 how-to questions from the query logs of the FAQ finder system (Burke et al., 1997). Three judges went through these questions and agreed on 10 *procedural questions*[5].

We searched Google and downloaded 40 top ranked documents for each question, which were then mixed with 1000 web documents from the SPIRIT collection to compile a test set. The two-stage architecture is as shown in figure 3. In the first stage, we sent only the content of the goal to a state-of-the-art IR model to retrieve 30 documents from the test set, which were reranked in the second stage according to the degree of procedurality by a trained document classifier.

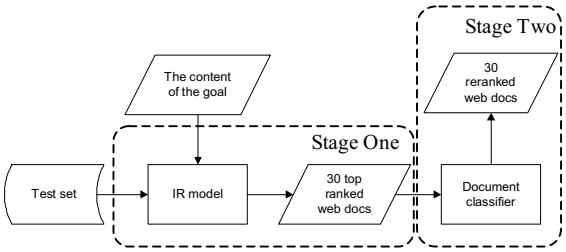

Figure 3. A two-stage architecture.

We also tried to test how well query expansion could help in retrieving procedural documents, following a process as shown in figure 4. First, key words in the content of goal were used to query an IR model to retrieve an initial set of relevant documents, those of which that do not contain the phrase 'how to' were then removed. The remaining top ten documents were used to generate 40 searching terms, which were applied in the second round to retrieve documents. Finally the 30 top ranked documents were returned as relevant documents.

[5] We distinguish questions asking for a series of steps (i.e., procedural questions) from those of which the answer could be a list of useful hints, e.g., 'how to make money'.

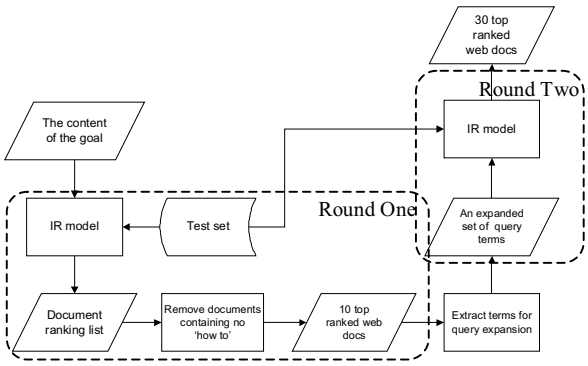

Figure 4. An alternative architecture using query expansion.

4.2 IR Model

For the above-mentioned IR model, we used the BM25 and PL2 algorithms from the Terrier IR platform[6].

The BM25 algorithm is one variety of the probabilistic schema presented in (Robertson et al. 1993). It has gained much success in TREC competitions and has been adopted by many other TREC participants.

The PL2 algorithm, as most other IR models implemented in the Terrier IR platform, is based on the Divergence From Randomness (DFR) framework. Amati and Rijsbergen (2004) provide a detailed explanation of this framework and a set of term-weighting formulae derived by applying different models of randomness and different ways to normalize the weight of a term according to the document length and according to a notion called *information gain*. They test these different formulae in the experiments on retrieving relevant documents for various sets of TREC topics and show that they achieve comparable result with the BM25 algorithm.

We also used the Bo1 algorithm from the same package to select terms for query expansion. Refer to (Plachouras et al., 2004) for details about this algorithm.

4.3 Result

We tested eight systems, which could be organized into two sets. The first set uses BM25 algorithm as the basic IR model and the second set uses PL2 as the basic IR model. Each set includes four systems: a baseline system that returns the result of the first stage in the two-stage architecture, one system that uses query expansion technique following the architecture in figure 4 and two systems that apply the two-stage architecture (one uses the Adapted Naive Bayes classifier and another one uses the Linear Regression classification model).

The mean average precision (MAP) [7] of different retrieval systems is shown in table 4 and figure 5.

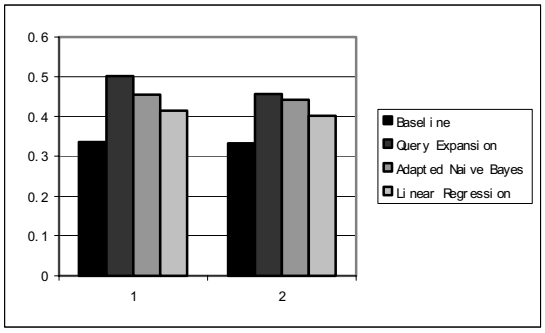

Figure 5. MAPs of different systems: 1 refers to using BM25 as the IR model, 2 refers to using PL2 as the IR model.

	Model	MAP
Set1	BM25 (Baseline)	0.33692
	BM25 + Query Expansion	0.50162
	BM25 + Adapted Naive Bayes	0.45605
	BM25 + Linear Regression	0.41597
Set2	PL2 (Baseline)	0.33265
	PL2 + Query Expansion	0.45821
	PL2 + Adapted Naive Bayes	0.44263
	PL2 + Linear Regression	0.40218

Table 4. Results of different systems.

We can see that in both sets: (1) systems that adopts the two-stage architecture performed better than the baseline system but worse than the system that applies query expansion technique; (2) the system that uses Adapted Naive Bayes classifier in the second stage gained better result than the one that uses Linear Regression classification model. We performed a pairwise t-test to test the significance of the difference between the results of the two systems with an integrated Adapted Naive Bayes classifier and of the two baseline systems. Each data set contained 20 figures, with each figure representing the average precision of the retrieving result for one question. The difference is significant ($p=0.02$). We also performed a pairwise t-test to test the significance of the difference between the two systems with an integrated Adapted Naive Bayes classifier and of

[6] http://ir.dcs.gla.ac.uk/terrier/index.html

[7] The average precision of a single question is the mean of the precision scores after each relevant document is retrieved. The mean average precision is the mean of the average precisions of a collection of questions.

the two systems using query expansion techniques. The difference is not significant (p=0.66).

4.4 Discussion

Contrary to our expectation, the result of the experiments showed that the two-stage approach did not perform better than simply applying a query expansion technique to generate an expanded list of querying terms. An explanation can be sought from the following two aspects (each of which corresponds to one of the two problems mentioned in the first section).

First, we expected that many documents that contain procedures do not contain the word 'procedure' or the phrase 'how to'. Therefore, a system based on key word matching would not be able to identify such documents. However, we found that such words or phrases, although not included in the body of the text, often occur in the title of the document.

Another problem we pointed out before is that the phrase 'how to' occurs in a much broader context than keywords in the content of the goal, therefore, it would bring a lot of irrelevant documents when used together with the content of goal for document retrieval. However, in our experiment, we used the content of the goal to retrieve document first and then removed those containing no phrase 'how to' (refer to figure 4). This is actually also a two-stage approach in itself.

Despite the experiment result, a well-known defect of query expansion is that it is only effective if relevant documents are similar to each other while the two-stage approach does not have this limitation. For example, for retrieving documents about 'how to cook herring', query expansion is only able to retrieve typical recipes while our two-stage approach is able to detect an exotic method as long as it is described as a sequence of steps.

5 Summary and Future Work

In this paper, we suggested that a how-to question could be seen as consisting of two parts: the specific goal and the general nature of the desired information (i.e., procedural). We proposed a two-stage architecture to retrieve documents that meet the requirement of both parts. We compared the two-stage architecture with other approaches: one only uses the content of the goal to retrieve documents (baseline system) and another one uses an expanded set of query terms obtained by automatic query expansion techniques. The result has shown that the two-stage architecture performed better than the base line system but did not show superiority over query expansion techniques. We provide an explanation in section 4.4.

As suggested in section 1, many information requests are formulated as consisting of two parts. As a future work, we will test the two-stage architecture for retrieving answers for other kind of questions.

References

Amati, Gianni and Cornelis J. van Rijsbergen. 2002. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Transactions on Information Systems,* 20 (4): 357-389.

Burke, Robin D., Kristian J. Hammond, Vladimir Kulyukin, Steven L. Lytinen, Noriko Tomuro, and Scott Schoenberg. 1997. Question answering from frequently-asked question files: experiences with the FAQ finder system. *AI Magazine*, 18(2): 57-66.

Clarke, Charles, Gordan Cormack, M. Laszlo, Thomas Lynam, and Egidio Terra. 1998. The impact of corpus size on question answering performance. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in IR*, Tampere, Finland.

Elworthy, David. 2000. Question answering using a large NLP system. In *Proceedings of the Ninth Text Retrieval Conference (TREC-9)*, pages 355-360.

Freund, Yoav and Llew Mason. 1999. The alternating decision tree learning algorithm. In *Proceeding of the Sixteenth International Conference on Machine Learning*, pages 124-133, Bled, Slovenia.

Hovy, Eduard, Laurie Gerber, Ulf Hermjakob, Michael Junk, and Chin-Yew Lin. 2001. Question Answering in Webclopedia. In *Proceedings of the Ninth Text Retrieval Conference (TREC-9)*, pages 655-664.

Kelly, Diane, Vanessa Murdock, Xiao-Jun Yuan, W. Bruce Croft, and Nicholas J. Belkin. 2002. Features of documents relevant to task- and fact-oriented questions. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management (CIKM '02)*, pages 645-647, McLean, VA.

Kendall, Maurice. 1979. *The Advanced Theory of Statistics.* Fourth Edition. Griffin, London.

Murdok, Vanessa and Bruce Croft. 2002. Task orientation in question answering. In *Proceedings of SIGIR '02*, pages 355-356, Tampere, Finland.

Picard, Justin. 1999. Finding content-bearing terms using term similarities. In *Proceedings of the Ninth Conference of the European Chapter of the Association for Computational Linguistics (EACL 1999)*, pages 241-244, Bergen, Norway.

Plachouras, Vassilis, Ben He, and Iadh Ounis. 2004. University of Glasgow at TREC2004: Experiments in web, robust and terabyte tracks with Terrier. In *Proceedings of the Thirteenth Text REtrieval Conference (TREC 2004)*.

Robertson, Stephen, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1993. Okapi at TREC-2. In *Proceedings of the Second Text Retrieval Conference (TREC-2),* pages 21-24.

Schwitter, Rolf, Fabio Rinaldi, and Simon Clematide. 2004. The importance of how-questions in technical domains. In *Proceedings of the Question-Answering workshop of TALN 04*, Fez, Morocco.

Stricker, Mathieu, Frantz Vichot, Gérard Dreyfus, and Francis Wolinski. 2000. Two steps feature selection and neural network classification for the TREC-8 routing. *CoRR cs. CL/0007016.*

Takechi, Mineki, Takenobu Tokunaga, Yuji Matsumoto, and Hozumi Tanaka. 2003. Feature selection in categorizing procedural expressions. In *Proceedings of the Sixth International Workshop on Information Retrieval with Asian Languages (IRAL2003)*, pages 49-56, Sapporo, Japan.

Tsur, Oren, Maarten de Rijke, and Khalil Sima'an. 2004. BioGrapher: biography questions as a restricted domain question answering task. In *Proceedings ACL 2004 Workshop on Question Answering in Restricted Domains*, Barcelona.

Weisstein, Eric. 1999. *Correlation Coefficient*. MathWorld--A Wolfram Web Resource. Available at: <URL: http://mathworld.wolfram.com/ CorrelationCoefficient.html> [Accessed 21 Oct 2005]

Witten, Ian and Eibe Frank. 2000. *Data Mining: Practical Machine Learning Tools with Java Implementations*, Morgan Kaufmann, San Mateo, CA.

Xu, Jinxi, Ana Licuanan, and Ralph Weischedel. 2003. TREC2003 QA at BBN: answering definitional questions. In *Proceedings of the Twelfth Text Retrieval Conference (TREC 2003)*, pages 98-106.

Yang, Yi-Ming. 1999. An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval* 1(1/2): 67-88.

Yang, Yi-Ming and Xin Liu. 1999. A re-examination of text categorization methods. In *Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99)*, pages 42-49, Berkeley, CA.

Yin, Ling and Richard Power. 2005. Investigating the structure of topic expressions: a corpus-based approach. In *Proceedings from the Corpus Linguistics Conference Series,* Vol.1, No.1, University of Birmingham, Birmingham.

Yin, Ling and Richard Power. 2006. Adapting the Naive Bayes classifier to rank procedural texts. In *Proceedings of the 28th European Conference on IR Research (ECIR 2006)*.