Getting Work Done on the Web: Supporting Transactional Queries

Yunyao Li‡ Rajasekar Krishnamurthy† Shivakumar Vaithyanathan† H. V. Jagadish‡*

‡University of Michigan
Ann Arbor, MI 48109
yunyaol,jag@umich.edu

†IBM Almaden Research Center
San Jose, CA 95120
rajase,shiv@almaden.ibm.com

ABSTRACT

Many searches on the web have a transactional intent. We argue that pages satisfying transactional needs can be distinguished from the more common pages that have some information and links, but cannot be used to execute a transaction. Based on this hypothesis, we provide a recipe for constructing a transaction annotator. By constructing an annotator with one corpus and then demonstrating its classification performance on another, we establish its robustness. Finally, we show experimentally that a search procedure that exploits such pre-annotation greatly outperforms traditional search for retrieving transactional pages.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation

Keywords

Information Extraction, Intranet Search, Transactional Search

1. INTRODUCTION

A user performing a web search may be interested in obtaining information regarding a topic of interest (informational search), in finding a specific web-page (navigational search), or performing a desired action, such as filing a claim, purchasing a ticket, or downloading a software package (transactional search). Search engines based on classical information retrieval assume that web searches are motivated by users' *information* needs. Models based on this traditional assumption have been found inadequate to serve "the need behind the query" in navigational and transactional search [8, 10, 18]. Meanwhile, it has been observed [3, 17] that an increasing fraction of all web search queries have a transactional intent. It is this class of queries that we address in this paper.

Consider, for example, an employee at the University of Michigan seeking to file a "property damage report." There is a unique university wide form available online. However, a search on the university's web site using the keyword query *property damage report* does not have a link to this form in the first 20 hits. Most links returned by the search are to pages that discuss property damage, and many are specific to particular departments. From several of these results it may be possible to navigate to the desired form, but the path is not obvious. Our goal in this paper is to provide better results for such a transactional search.

An obvious problem, when we consider an example such as the one above is that most web pages are informational and not transactional. Given a set of keywords, it is likely that there are many more non-transactional pages that include the given keywords than actual transactional pages. In fact, for many transactional searches, such as the one in the example above, there is precisely one web page of interest. The question is how to return this page with a high score.

Given our thesis that only a small fraction of pages in the repository are transactional, if we have a means to identify this set of pages (with high probability), then we can make sure to return only such pages (with high score) in response to a transactional search query. This is precisely what we set out to do, identifying a number of cues on a web page that suggest that it is transactional. It turns out that from the effort to identify transactional pages, we can get more than just whether the page is transactional. In particular, we can identify information regarding the nature of transaction supported by the page, and terms that are associated with this transaction. Our experimental results show that identifying such information is a critical component for supporting transactional search.

In traditional IR, there is a preparatory phase, when documents are inserted into a collection and indices are created (or updated), and an operational phase, when search queries are efficiently evaluated. For transactional queries, we do some additional work in the preparatory phase — specifically, we recognize documents that are likely to be relevant to transactional queries, and annotate them suitably. We call such documents *transactional pages*. The set of all transactional pages is a subset of the complete document collection. These documents can then be processed in different ways (Sec. 3) to create *transactional collection* for transactional search.

*Supported in part by NSF grant IIS 0219513

```
procedure $\textbf{\textit{TransactionAnnotator}}(\mathcal{C},\mathbf{W})$
$\mathcal{O}$: Set of transactional objects
$\mathcal{A}$: Set of actions

1. $\mathcal{O}$= $ObjectIdentifier_{\mathcal{C}}(W)$
2. $\mathcal{A}$= $ActionIdentifier_{\mathcal{C}}(W)$
3. if ($PageClassifier_{\mathcal{C}}(W,\mathcal{O},\mathcal{A})$)
4.     then return ($\mathcal{O}$,$\mathcal{A}$)
```

Figure 1: Template for Transaction Annotator

The recognition of transactional pages is performed by a *transaction annotator*, whose principles of operation are given in Sec. 2. Specifically, we identify all transactions supported by a page.[1] At the core is a classification algorithm which identifies web-pages that act as gateways to forms and applications (i.e., transactions). While the notion of identifying such gateways is not new [9], we present a novel templatized procedure aimed at increasing precision. Transaction annotators have no counterparts in traditional IR, and have to be constructed from scratch by us.

Thereafter, we would like to leverage standard techniques as far as possible. Specifically, in our implementation, we use the popular open-source Lucene [16] infrastructure. There are several design decisions to be made in this regard: whether to index all terms or only transactional terms, whether to index only the transactional collection or other documents also, and so forth. We consider such issues in Sec. 3.

An experimental assessment of the proposed techniques is given in Sec. 4. In Sec. 5 we discuss both reasons for the good performance observed by our system as well as the limits of applicability. The final sections present a discussion of related work and then conclusions.

In this paper, we focus solely on transactional queries; in a complete system, it may be necessary to determine whether a given user query is transactional by analyzing it. Such query classification is not included in the scope of this paper, so as not to confound issues of classification with issues of retrieval for the class (transaction queries) of interest. Instead, we refer the reader to techniques for automatic user goal identification that have been proposed recently [9, 10, 11, 14]

2. TRANSACTION ANNOTATOR

The transaction annotator serves two purposes: (a) classifies each web-page as being either transactional or not; and (b) returns those specific sections that support the transaction(s) (called *transactional features*). While multiple choices for a classifier exist (rule-based, machine learning etc.), our problem has several distinguishing characteristics that motivate our choice. Foremost, the interest is in very high-precision classification. Consequently the lack of a large training set combined with the fact that this is a single-class, classification problem makes transactional annotation a less appealing task for machine learning approaches [22]. Furthermore, the requirement of extracting relevant portions of the document necessitates the careful engineering of features. For the above stated reasons the advantages of a highly-optimized, hand-crafted, rule-based classifier clearly outweighs the obvious overhead involved in building such an annotator. In this section we provide an algorithm template that captures the essential elements of a transaction annotator. We then focus on two classes of transactions – software downloads (*SD*) and form-entry (*FE*), and describe the specifics of instantiating the template.

[1]Note that an individual web-page may support more than a single transaction.

```
procedure $\textbf{\textit{ObjectIdentifier}}_{\mathcal{C}}(W,\mathcal{RE}_P,\mathcal{RE}_N)$
$\mathcal{RE}_P = \{<re_i,f_i>\}$ is a set of positive patterns
$\mathcal{G}_N = \{G_i, f_i>\}$ is a set of negative patterns
$\mathcal{E}_{RE}$ is the pattern matching evaluation function
$\mathcal{E}_G$ is the gazetteer matching evaluation function
$\mathcal{BE}$ is a boolean expression for class $\mathcal{C}$
$\mathcal{R}$ is the set of objects identified

1. $\mathcal{CO}$= $CandidateObjectIdentifier_{\mathcal{C}}(W)$
2. foreach $o \in \mathcal{CO}$
3.     $\mathcal{CO}_P(o) = \{< re_j, f_j >\in \mathcal{RE}_P | \mathcal{E}_{RE}(o, re_j, f_j)$is true$\}$
4.     $\mathcal{CO}_N(o) = \{< G_j, f_j >\in \mathcal{G}_N | \mathcal{E}_G(o, G_j, f_j)$is true$\}$
5.     if $\mathcal{BE}(\mathcal{CO}_P(o), \mathcal{CO}_N(o))$
6.         then add $(o)$ to $\mathcal{R}$.
7. return $ConsolidateObjects_{\mathcal{C}}(\mathcal{R})$
```

Figure 2: Algorithm to Identify Transactional Objects

```
procedure $\textbf{\textit{ActionIdentifier}}_{\mathcal{C}}(W,\mathcal{RE}_P,\mathcal{RE}_N)$
$\mathcal{G}_A = \{g_i\}$ is a gazetteer of terms
$\mathcal{CA}$= set of actions identified in W
$\mathcal{E}$is the gazetteer matching evaluation function

1. $\mathcal{CA}$= $CandidateActionIdentifier_{\mathcal{C}}(W)$
2. foreach $a \in \mathcal{CA}$
3. if ($\mathcal{E}$(a,$\mathcal{G}_A$,f) is true)
4.     then discard $a$.
5. return $\mathcal{CA}$.
```

Figure 3: Algorithm to Identify Actions

Figure 1 shows the template for our transaction annotator. The first two steps identify *transactional objects* and *actions*, respectively. The former outputs the actual object of the transaction – e.g., the name of the software for *SD* while the latter identifies the action to be performed – e.g., downloadable links. Both steps rely primarily on checking for the existence of positive patterns and verifying the absence of negative patterns (Figure 2 and 3). Specifically, positive pattern matches are carefully constructed regular expression patterns and gazetteer lookups while negative pattern matches are regular expressions based on the gazetteer.

Consider, for example, the classifier that identifies *SD*. Firstly, candidate software names are extracted by $CandidateObjectIdentifier_{\mathcal{C}}$ (Figure 2, line 1) by looking for patterns resembling software names with version numbers (e.g., *Acrobat Reader 7.1*). Some of these will be false positives such as "*Chapter 1.1*". For each candidate object, *ObjectIdentifier* evaluates patterns comprising features in portions of the web page that are pertinent to the candidate object. Each pattern comprises a regular expression re and a feature f. For *SD* the only feature of interest is the *object-text* – i.e., the text that describes the software name (e.g., *Acrobat Reader* and *Chapter*). An example positive pattern for object-text requires that the first letter be capitalized. It is important to note that complex transactions (such as *FE*) contain a richer set of features. False positives such as "*Chapter 1.1*" will be pruned as a negative pattern using a gazetteer. A boolean expression $\mathcal{BE}$, over this set of positive ($\mathcal{CO}_P(o)$) and negative ($\mathcal{CO}_N(o)$) pattern matches, decides whether a candidate object (o) is relevant. The boolean expression we currently use is simple: it returns true if and only if $\mathcal{CO}_P(o)$ is non-empty and $\mathcal{CO}_N(o)$ is empty. Finally, the relevant objects are consolidated and returned by *ConsolidateObjects*$_{\mathcal{C}}$ (Figure 2, line 7). For example, *Adobe Acrobat Reader* and *Acrobat Reader* will be consolidated into a single object.

Similarly, *ActionIdentifier* (Figure 3) begins with the identification of several candidate actions. Again, through the use of several hand-crafted regular expressions and gazetteer lookups the candidate list is pruned to get a final list of actions and corresponding features.

At this point *PageClassifier*$_{\mathcal{C}}$ (Figure 1) classifies web pages based on the transaction object and actions on each web page: any web page that contains at least one transactional object *and* an action associated with the object is classified as a transactional page.

2.1 Instantiating and Optimizing the Classifier

We have described above an overall template for building transaction annotators. However, several decisions such as choice of transactional features, regular expression patterns and gazetteer entries are to be made dependent on the class of transactions being considered. For *SD* and *FE* used in this paper, the transaction annotator was constructed and optimized using a subset of a million documents from the IBM corporate intranet. *These were then used unchanged in our experimental evaluation on an entirely different dataset (the University of Michigan intranet).*

Feature engineering (identifying transactional features) and defining regular-expressions and gazetteers was accomplished using a manual iterative process (using the IBM intranet data). It is worthwhile mentioning that there is a complex interaction between the choice of features and regular expressions/gazetteers. The final set of features included hyperlinks, anchor-texts and html tags along with more specific features such as window of text around candidate objects and actions. In Table 1 we present several example patterns (regular expressions and gazetteers)[2] used by the transaction annotator for *SD*. Similarly in Table 2 we present example patterns used for *FE*. The first two columns describe where in the algorithm the patterns are used. The third column lists some example regular expressions or gazetteer entries as the case may be. The fourth column lists the feature on which the regular expression (or gazetteer as the case may be) is evaluated. For example, the first row describes an example pattern to identify candidate transactional objects. The regular expression is evaluated over the document text. This template can be utilized to identify other classes of transaction annotators.

3. TRANSACTIONAL COLLECTION AND OPTIMIZATION

The results of the transaction annotator described above is a set of transactional pages, each with associated set of *transactional features*. In this section, we discuss some choices with respect to how these pages are processed to create a transactional collection that is indexed by the search engine. We consider these design issues at multiple levels, in turn.

1. **Collection-level – Document Filtering:** Each transactional page must contain at least one transaction object. We could evaluate transactional queries solely against the transactional collection so obtained.

2. **Document-level – Term Filtering:** Filtering at a document-level is accomplished by retaining only portions of the document that have been identified as containing transactional features. Each transactional page is likely to contain many terms, only a small number of which are actually associated with the transaction. We could choose to index only terms that appear in the transactional features, and ignore the rest.

3. **Term-level – Synonym Expansion:** Transactional queries typically have a general form of <`action`> <`object`>. In many cases, the action has multiple synonyms and there is the possibility of a mismatch between the term appearing in the user query and that appearing in the web-page. (E.g. "obtain" rather than "download" some software package). The object on the other hand, being the name of an entity, is less likely to be obfuscated by the user. To address this potential mismatch we create the transactional collection by performing synonym expansion on all verbs appearing in the transactional features. Note that performing synonym expansion over the entire document collection will dramatically increase the size of the index. Restricting it to only verbs in the transactional collection, as in our case, prevent this problem (Sec. 4.4.3).

4. EXPERIMENTAL EVALUATION

In this section, we report experiments conducted to evaluate the effectiveness of transactional search and compare the processing choices discussed in Sec. 3.

4.1 Dataset and Queries

Ideally, we would have liked to have used a standard corpus and query set for our evaluation. Unfortunately, we found no suitable standard document collection available for transactional search, let alone a query set. As such, we had to create our own[3].

In our experiments, we used a collection of intranet Web pages obtained by using Wget [20]: given a single start point (URL of the entry page of a research university), the software recursively collected textual documents with a small set of MIME types (e.g., html, php) within the domain of `umich.edu` in November 2005. This Web page collection includes 434,211 documents with a total size of 6.49GB.

A set of 15 transactional search tasks was derived from an informal survey conducted among administrative staff and graduate students in the university: 10 of the tasks are to find a particular form(s); the rest are for software downloads. The list of tasks was then given to 26 subjects, who are all either current (graduate) students or recent graduates. Each subject was asked to write at least one keyword query for each search task. We obtained an average of 1.48 queries

[2]For ease of exposition, the example patterns presented in Tables 1 and 2 are simplified versions of the actual patterns used.

[3]Available at http://www.eecs.umich.edu/db/transactionalSearch

Table 1: Example patterns for *SD*

		Regular Expressions/Gazetteer Entries	Features
ObjectIdentifier	*CandidateObjectIdentifier*	[a-zA-Z]{2,} [Vv]?[0-9]([0-9.])+	document text
	(Positive)Regular Expr.($\mathcal{RE}_P$)	[A-Z][a-zA-Z]+	text of candidate objects
	(Negative)Gazetteer($\mathcal{G}_N$)	{chapter, section, version, since, ...}	text of candidate objects
ActionIdentifier	*CandidateActionIdentifier*	• [Tt]o download \w+ (click (on)?|go to)	•window of text around hyperlinks
		• download [a-zA-Z]+ from	•window of text around hyperlinks
		• .(exe|zip|tar(.gz)?|jar)	•hyperlinks in web-page
		• ˆ [Dd]ownload	•title of web-page
	(Negative) Gazetteer $\mathcal{G}_A$	{tutorial, instruction, form, survey, ...}	window of text around hyperlinks

Table 2: Example patterns for *FE*

		Regular Expressions/Gazetteer Entries	Features
ObjectIdentifier	*CandidateObjectIdentifier*	(.)+	hyperlinks in the web-page
	(Positive) Regular Expr.($\mathcal{RE}_P$)	• click here to	•window of text around hyperlink
		• following form is (used|related) to	•window of text around hyperlink
		• \w+ form$	•anchor-text
		• \w+ form$	•heading appearing before hyperlink
	(Negative) Gazetteer ($\mathcal{G}_N$)	{toolkit,presentation,cookbook,...}	window of text around hyperlink
ActionIdentifier	*CandidateActionIdentifier* (Positive) Gazetteer	{cancel,claim,register,nomination,...}	• title of web-page
			• window of text around hyperlinks
			• headings in web-page

Table 3: Example tasks (sample user queries)

Task 1	*Get recent trip reimbursed*
	(travel reimburse, reimbursement form)
Task 3	*Get the official letter of recommendation form*
	(recommendation letter form, candidate evaluation)
Task 6	*Alter some of the courses you have registered*
	(modify course registration, add/drop course)
Task 12	*Obtain Remedy client for your windows machine*
	(Remedy client windows, Remedy client download)
Task 14	*Obtain the virus scan software available locally*
	(download virus scan, virus scan software)

Table 4: Query statistics: mean and (std. deviation)

Avg. Number of Unique Queries/Task (SD)	26.3 (6.34)
Avg. Number of Words/Query (SD)	2.95 (1.14)

per task from each subject. After removing duplicates, a total of 394 unique queries were collected for the 15 tasks. Example tasks are listed in Table 3 and statistics on the query collection is reported in Table 4.

4.2 Experimental Set Up

We used Apache Lucene [16], a high-performance, full-featured text search engine library, to index and search the following data collections:

- **S-DOC**: This is the original dataset described in Sec. 4.1.
- **S-TDC**: Based on the existence of transactional objects, each document in S-DOC was classified as being a transactional page or not. We separately indexed the collection of transactional pages. Note that this collection is a strict subset of the pages in S-DOC.
- **S-ANT-NE**: This collection is created by writing all the transaction features (for both *SD* and *FE*) on the same document into a single file. The identifier associated with each file is the original document.
- **S-ANT**: This collection is generated similar to S-ANT-NE, but with term-level synonym expansion. We used WordNet [21] as our general thesaurus to expand the verbs in the transactional features.

4.3 Evaluation Metrics

Multiple different metrics, including precision, recall, and F-measure, can be used to evaluate the quality of the result returned by a search. In the case of a transactional search, it is most often the case that the user is only interested in one way to perform the transaction — that there are other additional ways matters less. In other words, the user is likely to care the most about the top ranked relevant match returned.

In light of the above, results of most experiments are reported in terms of the mean reciprocal rank (MRR) measure. For each unique query of each task, the reciprocal value ($1/n$) of the rank (n) of the highest ranked correct result is obtained. This value is then averaged over all the queries corresponding to the same task. The reciprocal rank of a query is set to 0 if no correct result is found in the first 100 pages returned.

In our study, correct answers are web pages that can support the desired transaction tasks. For example, a correct answer for "*download Remedy Client*" must be a web page from which the software *Remedy Client* can be downloaded directly. As such, there is little subjectivity in determining relevance.

For completeness, we also present recall measurements, but only for the main experiment in Section 4.4.1.

4.4 Experimental Results

4.4.1 *Transactional Search*

As a first test, we wish to judge how effective our system is for transactional search. The MRR for each task over S-DOC and S-ANT is reported in Figure 4. As can be seen, search based on S-ANT almost always outperforms that based on S-DOC. For nearly two third of the tasks, S-ANT achieves higher than 0.5 in the MRR, while S-DOC achieves similar performance for only 3 of them. In particular, for 5 of the tasks (Task 1,2,4,5,10), MRR of S-ANT is significant higher than that of S-DOC. For tasks 1,2,4, and 5, the MRR of S-ANT is higher than 0.5, indicating that S-ANT on average returned a correct answer in its top two results. In contrast, the MRR of S-DOC for the same set of tasks is lower than 0.05. For Task 10, S-ANT performed slightly worse, with its MRR being right below 0.3; however, it still significantly outperforms S-DOC, whose MRR is merely 0.009.

Based on the results reported in Figure 4, the search tasks can essentially be divided into two categories (i) tasks where S-ANT significantly outperforms S-DOC (Task 1, 2, 4, 5, 10), and (ii) tasks where S-ANT achieves better MRR than S-DOC by a smaller margin (such as Task 3, 6). Figure 6 presents reciprocal rank of S-ANT and S-DOC on individual queries for selective tasks in these two categories. For tasks in the first category (Figure 6(a), (b)), S-ANT performs extremely well over individual queries: it returned correct results in the first rank for more than half of the queries, and in up to fifth rank for more than three fourth of the queries. In contrast, S-DOC never returned correct results in the first rank for any query, and only returned correct results in up to tenth rank for less than one tenth of the queries for the same tasks.

For tasks in the second category, the advantage of S-ANT over S-DOC is still obvious but smaller (Figure 6(c), (d)). S-ANT returned correct results in the first and second rank for about half of the queries but failed to return correct results in the top ten rank for others. Meanwhile, S-DOC returned correct results in up to tenth rank for nearly one third of the queries, and even returned correct results in the first rank for about one tenth of them.

Note that for Task 9, both S-ANT and S-DOC perform badly, with S-ANT performing worse than S-DOC. This result is not surprising. Task 9 asks for a form used to request telephone repair service, but there is no such form in our dataset (At the University of Michigan, telephone repair is requested by telephoning a repair number, not by filling in a web form). For this task, web pages with the correct contact information for the service were counted as correct results. One such page was included in S-ANT due to a different transaction being identified on that page. Recall, however, that transaction annotators only keep features related to transactions. Since there is no transaction object related to this task, keyword search for the task on S-ANT is essentially a search over less related document content to the task, explaining the worse results.

While it is straightforward to see how many relevant answers there are in the result set, it is hard to know how many answers there are in the document collection as a whole. As such, it is difficult to measure recall. Instead, for each task we approximate the total number of relevant web pages by obtaining the union of the correct answers in the top 100 results for S-DOC and S-ANT[4]. The recall of S-DOC and S-ANT at the cutoff rank 100 (Recall@100) is reported in Figure 5. As can be seen, even though S-ANT (12.6MB) is only a tiny fraction of S-DOC (6.49GB), it actually retrieved more correct results[5] in its top 100 results for two third of the tasks, and only returned fewer correct results for three tasks.

4.4.2 Term Filtering vs. Document Filtering

We also wish to compare the effectiveness of transactional collection generated via term filtering and document filtering. The result of the study between S-ANT (term filtering) and S-TDC (document filtering) is shown in Figure 7. As can be seen, S-ANT performs better than S-TDC in 13 out of 15 tasks. This result shows that when the transaction annotator does a good job in identifying relevant features, retaining only these features in the transactional collection improves search performance, as opposed to retaining the entire document.

[4]Specifically, the two most frequent keyword queries for each task was used to get the 100 top results.

[5]There frequently are multiple pages linking to the same form, and each of the page is considered a correct result in our study.

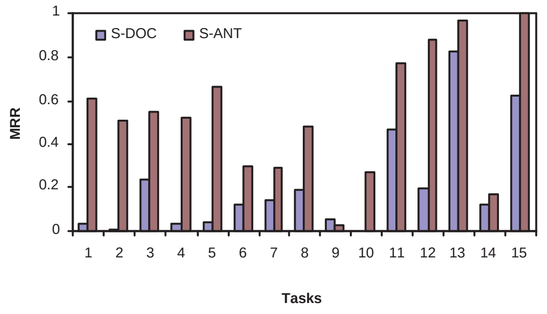


Figure 4: MRR for S-DOC and S-ANT.

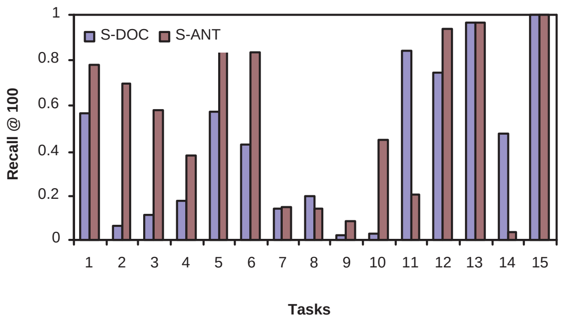


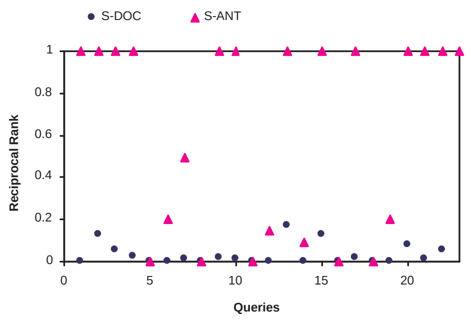
Figure 5: Recall@100 on S-DOC and S-ANT.

The cases where unrelated content may help involve queries that are less well defined, such as those for Task 6 and 14. For example, none of the keyword queries specified for Task 14 contains the actual software name "*VirusScan*"; rather, more general terms such as "*anti-virus*" and "*virus scan*" are used. In future work, we intend to investigate techniques to obtain useful context for such cases.

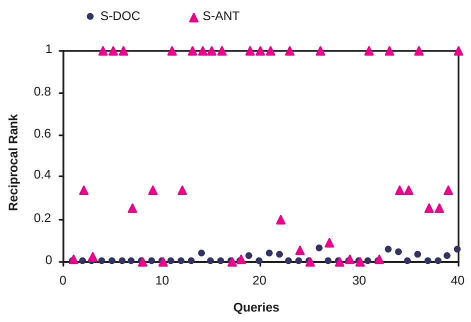
4.4.3 Synonym Expansion

Finally, to evaluate the effectiveness of synonym expansion for transactional features, we compare the performance of S-ANT and S-ANT-NE in Figure 8(a) for queries containing a verb[6]. As can be seen, for such queries in FE tasks (Task 1-10), the advantage of synonym expansion is evident, while for SD tasks, synonym expansion does not provide better results. For SD tasks, almost all user queries with a verb use the same word "download", which is contained by all transactional features of SD pages. S-ANT performs worse than S-ANT-NE for Task 9 and 14 due to the abnormalities discussed previously: (i) there is no form for Task 9 in the dataset; and (ii) no query specified for Task 14 contains the actual software name. When we consider all the user queries for each task (including those without a verb), the comparison of S-ANT and S-ANT-NE is shown in Fig-

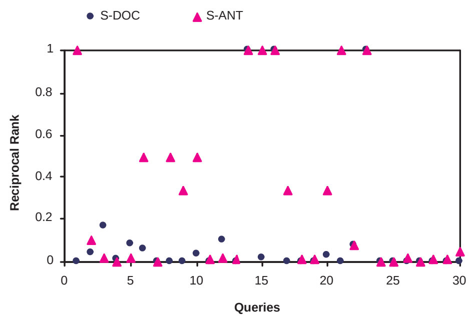
[6]Our query set includes 154 such queries, out of a total of 394 unique queries.



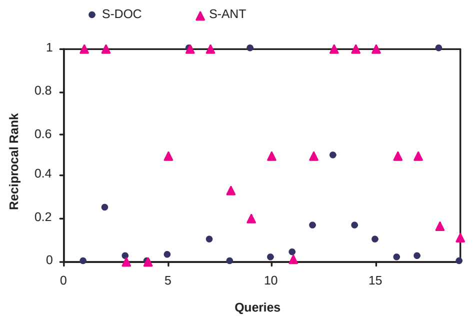
(a) Task 1



(b) Task 2



(c) Task 3



(d) Task 6

Figure 6: Reciprocal rank on individual queries.

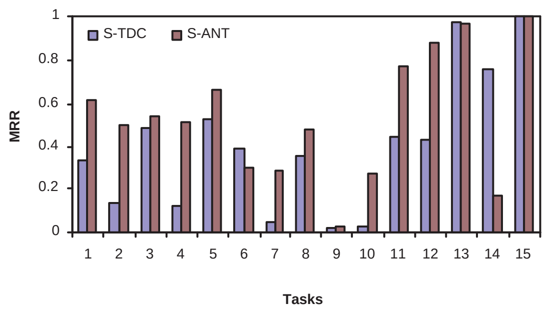


Figure 7: MRR for S-TDC and S-ANT.

ure 8(b). Among the FE tasks, tasks 2, 6, 7 and 10 still show considerable improvement of S-ANT over S-ANT-NE, while the difference is marginal for the rest. This can be explained by the fact that a larger fraction of user queries for these four tasks contain a verb, as opposed to the other tasks.

The cost of synonym expansion in terms of index size is 89.3%, with the size of Lucene index increases from 1.77MB for S-ANT-NE to 3.35MB for S-ANT.

5. DISCUSSION

It is clear from the experimental results above that pre-identifying and annotating transactional pages leads to a vastly superior performance in transactional search. In this section, we consider why that might be the case.

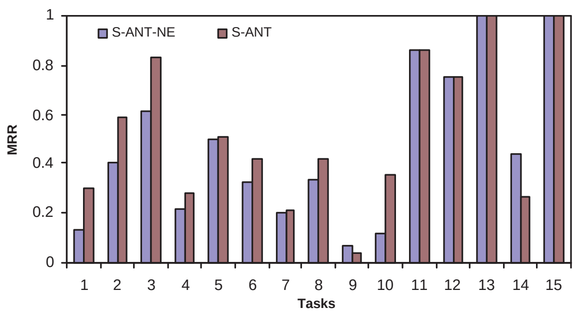
5.1 Collection Size

We begin by noting that for the experimental dataset, as expected, the size of the transactional collection was 434,211 pages, only a very small fraction of the total collection size of 22,967 pages are transactional. A fundamental question to ask is whether the transactional collection as identified by our classifier is indeed the correct set. Could there be errors in classification?

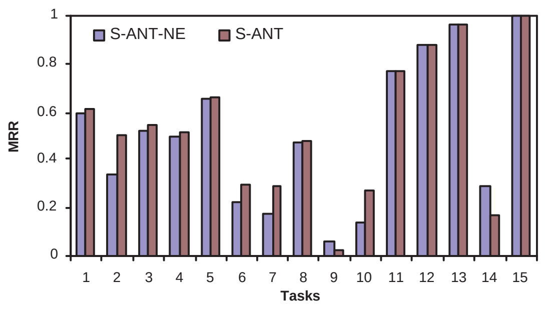
We manually took a random sample of 100 documents from the transactional collection and found that each of these was indeed transactional. We then took a random sample of 500 documents from the full document collection. Of these, we found that all documents that were identified as transactional, indeed were transactional, but not vice versa. Specifically, we found 30 transactional pages in total, of which 23 were included in the transactional collection by the classifier, and 7 had been missed. In other words, our transaction annotator is identifying *roughly 3/4* of the transactional pages. This gives us more confidence in our superior search performance as it indicates that a fair-sized fraction of the transactional pages from S-DOC are present in S-ANT.

5.2 Probabilistic Model

For the most part in our experiments the retrieval engine, Lucene, has been treated as a black-box thus making it difficult to isolate the different aspects of transactional search and evaluate them. To understand the value of identifying transactional pages, we implemented a simple language-model retrieval engine where all assumptions can be made explicitly. A standard language model, in response to a



(a) Over queries containing a verb



(b) Over all queries

Figure 8: MRR for S-ANT and S-ANT-NE.

query, ranks documents based on[7]

$$P(Q|\Omega_d) \cdot P(d) \tag{1}$$

where Q is the keyword query, $P(Q|\Omega_d)$ is the likelihood of producing the query Q given Ω_d the language model for document d. $P(d)$ is the prior probability for document d often assumed to be uniform.

Since our goal is transactional search, apriori we have a bias towards pages that are transactional. It is precisely this belief that we encode in the computation of $P(d)$ where each page has a prior probability that is proportional to the number of transactional objects identified in Section 2. Similar reasoning for home page search can be found in [13].

$$P(d) = \frac{1 + O(d)}{N + O(N_d)} \tag{2}$$

where $O(d)$ is the number of transactional objects in document d, $O(N_d)$ is the total number of transactional objects identified in the collection and N is the total number of documents in the collection. Smoothing has been added to account for documents that have no transactional objects.

Ω_d was modeled as a multinomial and the following mixture probability was used to compute $P(Q|\Omega_d)$[8].

$$P(Q|\Omega_d) = \prod_{t \in Q} (\lambda * P(t|\Omega_d) + (1 - \lambda) * P(t)) \tag{3}$$

where t refers to an individual term; $P(t)$ is the collection level probability for term t computed as the ratio of the number of times term appears in the collection and the total number of terms in the collection. Term-level synonym expansion was not performed for the language model implementation.

The results obtained are shown in Figure 9. As can be seen, transactional prior help to increase search effectiveness in almost all tasks on document collection (S-DOC). These results are very pleasing and validate our intuition that pre-identifying transactional pages is important. Certainly, more powerful language models incorporating other assumptions (transactional terms and important parts of speech) is possible but beyond the scope of this paper. We intend to continue this line of research further to better understand transactional search.

[7]Strictly speaking we are interested in the posterior $P(d|Q)$ which is given by $\frac{P(Q|\Omega_d) \cdot P(d)}{P(q)}$ where $P(Q|d)$ has been replaced by the model for d.

[8]Some details are not mentioned here in the interests of space such as minimal feature selection using stop-word lists, removal of very frequent terms etc.

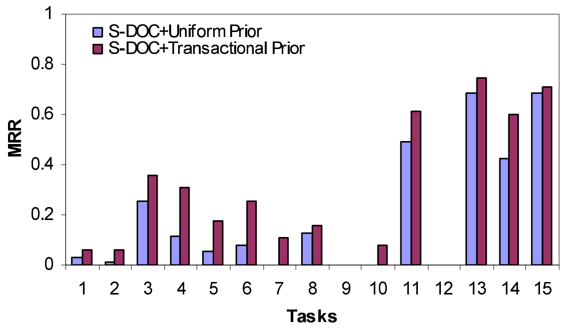


Figure 9: MRR for S-DOC with uniform prior and with transactional prior.

6. RELATED WORK

This research has its roots in web page search, web genre detection, and information extraction. The following section discuss related work in these fields.

Web Genre Detection: Automatic web genre identification (AWGI) has been recently studied as a key factor to improve search results [5, 12]. For example, genre classification could allow users to sort search results according to their immediate interests [12]. While current AWGI techniques could improve navigational search (e.g., home pages [5]), we are not aware of similar work for transaction search.

Web Page Search: Structural information on a Web page, such as URL, anchor text, and title, has been used to improve entry page search and link-based page classification with great success [4, 7, 9, 13, 19]. We have shown that structural information can also be exploited in transaction annotators to improve transactional search.

Most work on web page search is focused on Internet search. A notable exception is [6]. In this study, the authors identified a few fundamental differences between intranet and Internet. Among these differences, the following directly motivated our work: (i) "intranet documents are often created for simple dissemination of information;"(ii) "a large fraction of queries tend to have a small set of correct answers (often unique), and the unique answer pages do not usually have any special characteristic;" (iii) "intranets are spam-free." Our example annotators are also designed by taking such differences into consideration.

Search Goal Detection: Techniques based on classification and user behavior [9, 10, 11, 14] have been proposed to support automatic user goal identification in web search. These techniques are important for transactional search, as it is often necessary to identify transaction queries before search. The most relevant work to ours is [9], in which Kang proposed the notion of *Service Link* and used it to improve query type classification. The idea behind Service Link bears great similarity to that of our transaction annotator: both classify pages based on structural information within each page. One key difference is that [9] does not support careful extraction of transaction features. In the Internet, the context of [9], simple adjustment based on Service Link to the search results of commercial search engine seems serve the purpose well. However, as our comparison study has shown, such page level annotation is not adequate for intranets, which is far less search-engine-friendly. For such environments, supporting extraction of transactional features are necessary for better transactional search .

Information Extraction: Transaction annotators often heavily depend on information extraction techniques for page classification and transactional feature extraction. Depending on the type of transaction and/or dataset of interest, different information extraction techniques can be applied. For example, in object identification for software download pages (Sec. 2), finding software name on each web page is essentially a Named Entity Recognition (NER) task [2], which is to identify proper names in text, such as geographic names [1, 15]. Our current implementation of transaction annotator mostly relies on structural pattern matching. But more advanced information extraction techniques can be easily adopted to improve transaction annotators and benefit transactional search.

7. CONCLUSIONS

In this paper we introduced a methodology for transactional search that is based upon identifying and pre-annotating transactional pages. We developed a template-driven architecture for a rule-based transaction annotator that is capable of highly specific labeling of many distinct transaction types. We showed the robustness of our annotator design by constructing and optimizing it on one data set (from a corporate intranet) and reporting results of its use, with no modifications, on a different data set (from a university intranet). Even without dataset specific optimization, the new methodology was able to show dramatic improvement in search results produced.

Transactional search is very important, and is growing in importance on the web. On the extranet web, better serving transactional need has significant economic benefit potential associated with it. It is typically the case that a user is willing to spend \$500 to book a hotel on the web, including a \$50 commission to the web site, but is unwilling to spend even an amount smaller than \$50 for information to help choose the hotel. The business model for many web businesses is to provide information for free in the hope of making money from a transaction that eventually results. Not surprisingly, generic search engines (as far as we can tell) already special case selected transactional searches. For instance, a Google query naming a pair of cities will bring up a link to flight booking services as the top hit, before providing the standard list of matches. It is therefore in the interests of companies like Orbitz and Expedia to provide appropriate additional information to search engines, in order to enable them to support transactional requests better.

On the other hand, intranets are usually managed in a de-centralized fashion (e.g., departments in universities and geographical boundaries or product divisions in large multinational companies). Groups within the organization that support various transactions (such as filing a claim and downloading software) have little coordination while exposing these services. Since the economic imperatives for the intranet are simply not as strong as the extranet, it is unlikely that anyone will manually identify and manage webpages that serve transactional requests. Furthermore, only a small fraction of pages in the intranet are typically relevant to transactional search, and ordinary search is not good at finding such pages. Therefore we expect that techniques developed in this paper, while useful for all transactional web searches, will be of greatest value for intranet searches.

8. REFERENCES

[1] E. Amitay et al. Web-a-where: geotagging web content. In *SIGIR*, 2004.

[2] D. M. Bikel et al. An algorithm that learns what's in a name. *Machine Learning*, 34(1-3):211–231, 1999.

[3] A. Broder. A taxonomy of Web search. In *SIGIR Forum*, 2002.

[4] N. Craswell et al. Effective site finding using link anchor information. In *SIGIR*, 2001.

[5] A. Dillon and B. Gushrowski. Genres and the web - is the home page the first digital genre? *JASIS*, 51(2):202–205, 2000.

[6] R. Fagin et al. Searching the workplace Web. In *WWW*, 2003.

[7] J. Furnkranz. Exploiting structural information for text classificatio on advances in intelligent data analysis. In *IDA*, 1999.

[8] D. Hawking et al. Which search engine is best at finding online services? In *WWW*, 2000.

[9] I.-H. Kang. Transactional query identification in web search. In *AIRS*, 2005.

[10] I.-H. Kang and G. Kim. Query type classification for web document retrieval. In *SIGIR*, 2003.

[11] I.-H. Kang and G. C. Kim. Integration of multiple evidences based on a query type for web search. *Inf. Process. Manage.*, 40(3):459–478, 2004.

[12] B. Kessler et al. Automatic detection of text genre. In *ACL*, 1997.

[13] W. Kraaij et al. The importance of prior probabilities for entry page search. In *SIGIR*, 2002.

[14] U. Lee et al. Automatic identification of user goals in web search. In *WWW*, 2005.

[15] H. Li et al. InfoXtract location normalization: a hybrid approach to geographic references in information extraction. In *Workshop on the Analysis of Geographic References*, 2003.

[16] Lucene: http://lucene.apache.org/java/docs/.

[17] D. E. Rose and D. Levinson. Understanding user goals in web search. In *WWW*, 2004.

[18] T. Upstill et al. Buying bestsellers online: A case study in search & searchability. In *ADCS*, 2002.

[19] T. Westerveld et al. Retrieving web pages using content, links, urls and anchors. In *TREC*, 2001.

[20] Wget: http://www.gnu.org/software/wget/wget.html.

[21] Wordnet: http://wordnet.princeton.edu/.

[22] H. Zhu et al. Topic learning from few examples. In *PKDD*, 2003.