

Finding buying guides with a Web carnivore

Reiner Kraft

Department of Computer Science, UC Santa Cruz
1156 High St., Santa Cruz, CA 95064, US
rekraft@soe.ucsc.edu

Raymie Stata

Department of Computer Science, UC Santa Cruz
1156 High St., Santa Cruz, CA 95064, US
raymie@cs.ucsc.edu

Abstract

Research on buying behavior indicates that buying guides perform an important role in the overall buying process. However, while many buying guides can be found on the Web, finding those guides is difficult to impossible for the average consumer. Web search engines typically index many buying guides on many topics, but simple queries do not often return these results. Given this, we built a Web carnivore that finds buying guides on behalf of consumers. Web carnivores leverage the crawling, scrubbing, indexing and ranking activities of Web search engines (the "herbivores") to provide more specific services. Ours finds buying guides by issuing machine-generated queries to Google and filtering the results.

This paper describes our system and quantitatively compares it to a basic search engine. Our system almost always returns more buying guides, often twice as many. Our user study also suggests that we return *better* buying guides. Finding buying guides is an instance of the more general problem of *genre search*; this paper points out novel aspects of our system that are applicable to a variety of genre-search problems.

Keywords

Searching and ranking, data mining

1. Introduction

Silverman et al. [1] report that, although consumer purchasing on the Web is increasing, consumers still abort 23% of sales transactions that they initiate, four out of five because of search-related reasons. Further, they report that many more potential transactions are never even initiated due to search-related failings. The current process of finding information relevant to purchasing decisions is simply too complicated for Internet-naive people. These findings motivated us to explore new kinds of search systems that help consumers at various stages of the buying process.

Guttman et al. [2] provides an overview of marketing research related to consumer buying behavior. This research describes a multi-activity decision-making process in which *product brokering* is an important, early activity. The goal of product brokering is to learn the high-level features and characteristics of a product category (vs. a specific product model) and how those features and characteristics relate to the buyer's personal needs and constraints. *Buying guides* provide just this information. Good buying guides for digital cameras, for example, describe features such as "pixel count" and "optical zoom" and relate those features to personal goals such as "sending prints to grandmother" and to constraints such as budget.

The Web contains many good buying guides on a bewildering range of product categories. The Web also contains lots of useful shopping pages that are *not* buying guides. For example, product reviews describe the features and benefits of one or a few particular products. Comparison shopping sites compare features and prices of typically quite a few particular products. Although this information is valuable, it

typically does not address the needs of product brokering. Unfortunately, this paucity of information makes it difficult to impossible to find buying guides when such guides are needed.

As part of our larger program on decision-support systems (DSS) for consumer e-commerce, we have built a *buying-guide finder* (BGF) to aid in the important activity of product brokering. The BGF takes as input a *product category* (a.k.a. *topic*), such as "digital camera" or "washing machine", and returns buying guides for the given product category. Our user-study of this system indicates that it almost always returns more buying guides than does naive Web searching, often twice as many. This study also suggests that we return *better* buying guides.

We implemented our BGF as an *Web carnivore*. Etzioni coined this colorful phrase in [6]. In this analogy, Web pages are at the bottom of the Web information food chain. Search engines are the *herbivores* of the food chain, grazing on Web pages and regurgitating them as searchable indices. Carnivores sit at the top of the food chain, intelligently hunting and feasting on the herbivores. The carnivore approach leverages the significant and continuous effort required to maintain a world-class search engine (crawling, scrubbing, de-spamming, parsing, indexing, and ranking). In a search context, the carnivore approach is applicable when standard Web search engines are known to contain the documents of interest but do not return them in response to naive queries.

Our basic approach is to expand the user's product-category phrase into a sequence of queries we feed to a standard Web search engine. We then filter the metadata returned by the search engine (e.g., title, snippet) to identify buying guides in the result stream. Central to both query-expansion and filtering are *genre terms*, terms related to the concept of "buying guide" and orthogonal to the particular topic. The work described in this paper focuses on the identification and use of such genre terms. We believe this aspect of our work is applicable to other instances of the *genre search problem*.

The [next section](#) of this paper describes our BGF system, including a number of variations with which we have experimented. [Section 3](#) describes an evaluation comparing the results of a standard Web search engine to our BGF. [Section 4](#) describes related work, [Section 5](#) describes future work, and [Section 6](#) concludes.

2. Approach

Our approach is based on a manual technique that many expert Web users employ when searching the Web for documents within a genre. When manually performing genre searches, one occasionally obtains good results by simply typing the product category directly into a regular search engine. Also, sometimes a specialized search engine is available (e.g., the Movie Review Query Engine [11]). However, when these simple approaches fail, our anecdotal investigation indicates that many expert Web users utilize the following refinement approach:

- *Genre expansion*. The first step is to add genre-related terms to the query. In the case of buying guides, for example, such terms include *buying guide* and *choosing*. Genre expansion is effective for mining search engines for documents within a genre.
- *Topic expansion*. For some topics, better results are obtained by trying alternative terms for the product category.
- *Iteration*. Genre and topic expansion are applied iteratively, sometimes exploring local optimizations (modifying the previous query in a small way to refine the results), sometimes exploring global optimizations (entering a different query to find a new collection of results).
- *Filtering*. During this iterative process, the Web searcher identifies and records good results. When enough results are found, the search stops.

Our BGF (buying-guide finder) automates this technique. Our system repeatedly executes *trials*, each trial consisting of three steps: building a query (during which we employ genre and topic expansion), issuing the query to a search engine, and harvesting results (where we employ filtering). Each trial yields some number of results. The system runs trials until the desired number of results are obtained.

Our system uses [Google](#) as its underlying search engine. Google is well-suited to our task because it has broad coverage, good ranking, and a nice API [13] that allows programmatic access. At a high-level, our

results are independent of Google and could be applied with any search engine. However, as will be seen, certain aspects of the Google API do impact the details of our system.

The substance of our system rests in building the query and harvesting results. The rest of this section discusses these issues in detail. Before describing the details, [Section 2.1](#) discusses the testing infrastructure we created to guide our development. [Section 2.2](#) discusses our approach to query formulation. [Section 2.3](#) discusses our method of filtering. [Section 2.4](#) describes how we select terms and term-vectors referred to in [Sections 2.2](#) and [Section 2.3](#).

2.1 Binoculars: our benchmark topic

Before describing the details of our system, we first describe the infrastructure we created for tuning those details. It became clear early that, in the details of what we were building, we were facing a huge number of design decisions each of which could have a significant impact on the performance of our system. On the one hand, it was impractical for us to do user-based evaluations of the many combinations we were facing. On the other hand, we wanted a mechanism for using data to test our decisions.

We created such a mechanism by manually building a reasonably exhaustive *benchmark set* of buying guides for a single *benchmark topic* ("binoculars"). When exploring our design space, we used traditional precision *and* recall metrics to measure the impact of various alternatives. Later in this section, when we say that one approach works "better" than others, such claims are based on measurements against this benchmark set. During development, to avoid over-tuning to our benchmark topic, we very occasionally performed additional evaluations against a handful of *secondary benchmark topics* ("digital cameras" and "cars"). In these secondary evaluations, we simply measured precision of the first tens of results by manually inspecting for buying guides.

As just suggested, this approach to tuning could over-tune our system to the benchmark topic. It could also introduce a biased notion of what is a "buying guide." The evaluation in [Section 3](#) suggests that neither of these issues became a problem for us.

We selected "binoculars" as our primary benchmark topic for the following reasons. First, we wanted a topic that was somewhat obscure but which had non-trivial representation in the Web. Google indicated that it has around 1.5M documents containing "binoculars" (as of June '03), a small number but not a tiny one. Second, we wanted a non-electronic product category, under the theory that buying guides for electronic categories are more easily found on the Web (the evaluation suggests this may not be the case).

Given the topic "binoculars," we manually searched Google for buying guides. We examined over 3,000 (unique) results from over a 1,000 queries (carefully crafted to be biased towards buying guides) and identified a *benchmark set* of around 200 buying guides. Based on the diminished returns we observed late in our searching process, we believe this benchmark set to be reasonably exhaustive for our topic.

It should be noted that the process of manually creating the benchmark set gave us good insight into the types of queries do and do not tend to turn up buying guides. It also pointed out certain problems (e.g., "genre drift") that we needed to address. Thus, we benefited unexpectedly from generating this benchmark set early in the project.

2.2 Generating queries: query templates

On the surface, our approach to generating queries seems easy: simply combine some topic terms and some genre terms. However, a number of details conspire to make it harder than it first seems:

- **Term generation.** We need to generate a *candidate pool*, that is, a set of *candidate terms* from which we randomly pick to generate queries. We divide this pool into *topic candidates* and *genre candidates*, and in fact further subdivide the latter into *verb*, *noun* and *adjective* genre candidates.
- **Query formulation.** Once we have a good candidate pool, we need a process for turning it into a sequence of queries.

We return to the problem of term generation in [Section 2.4](#). The rest of this subsection discusses query formulation.

Naively giving the entire candidate pool to a search engine as a single query does not yield good results. Rather, our experience indicates that queries need to be formulated more surgically, e.g., combining a single topic term with a genre verb and genre noun. Further, Web search engines typically have rich query languages. For example, Google treats terms early in the query as more "important" than latter words, it is sensitive to the proximity of words within a query, and it offers a number operators such as `intitle:` [13]. We found that utilizing such features can greatly improve yields.

These factors caused us to create a *query template language* to more quickly explore the huge space of strategies for query formulation. A *query template* is a pattern for generating a family of queries. Roughly speaking, our template language works like this:

- Our four candidate pools are represented by the keywords `TOPIC`, `GENRE_NOUN`, `GENRE_VERB` and `GENRE_ADJECTIVE`.
- A simple template is simply a sequence of literal text and 'placeholders'. For example, the template `"intitle:TOPIC GENRE_NOUN"` generates a query consisting of `"intitle:"` followed by a topic term followed by a genre-noun.
- If expanded exhaustively, even a small template and candidate pool can generate a huge number of queries. However, our high-level strategy is to generate just a few queries per template, then move on to another template in a sequence of templates. Thus, we need a mechanism for controlling the number of expansions that occur.

This is done through decorations placed on the placeholders. Thus, for example, `TOPIC=2` indicates that only two expansions should be tried. The template `"TOPIC=3 GENRE_VERB=2"` would generate six queries. The expansion is done in a pseudo-random fashion such that the same template against the same candidate pool will generate the same expanded queries.

The above description is meant only to give a rough overview of our rich template language. It is beyond the scope of this paper to describe it entirely.

As already mentioned, our strategy is to generate only a few queries (5-10) from a given template, harvest the results from these, and then move on to another template in a larger sequence. The system evaluated in [Section 3](#) used ten hand-crafted templates. We generated these templates by both reflecting on our experience in creating the benchmark set and also by using the benchmark set to test a large set of alternatives. In general, templates that start with topic terms followed by genre terms seem to work best. At the same time, we slightly favored more genre terms and fewer topic terms. We found that, on the genre side, mixing parts of speech yields better results, e.g., "choosing guide" works better than "choosing selecting". This observation led us to creating multiple classes of genre terms. Finally, we found that smaller, simpler templates yield better results. Thus, when ranking templates, we try the simple ones first and move to the more complicated ones if the simple ones do not yield enough results.

2.3 Harvesting results: genre screening

Recall that our overall algorithm looks something like the following:

```
R := { };
for each query template T in our template sequence:
  for each query Q generated by T:
    r := submit Q to the search engine;
    R := R UNION filter(r);
    if (|R| > goal), we're done, otherwise continue;
```

As the above pseudocode suggests, we have found that filtering the output using a *genre screener* yields higher-quality results (this finding is in keeping with [14]).

Genre screening is based on term vectors. We combine terms associated with a result -- including its title terms, URL, and "snippet" terms -- into a *result term vector* which we then compare against a *genre screening vector*. (Note that the inclusion of a fair number of genre terms in the query yields a fair number of genre terms in the snippets of good documents.) Our process for generating this screening vector is described in the next subsection. We compare these vectors by computing their cosine.

We observed that "genre drift" in earlier versions of our system. For example, queries often took us to price-comparison pages (which, as discussed in the Introduction, are not what we mean by "buying guides"). To counter such drift, we added a *genre discrimination vector*. This vector contains words (such as *comparison*) which are *negatively* correlated with the results that we desire.

Thus, our total filtering process consists of computing a genre screening and discrimination score using these two vectors, combining them in a linear fashion that gives extra weight to discrimination, and then selecting against a rather high threshold. We take at most two results from each query, which yields a higher-quality result-set overall. By over-weighting the discriminator and thresholding on the high side, we are being conservative. However, filtering is done in a context in which many candidates are being generated. (Again, reminiscent of [14].)

2.4 Selecting terms: PMI-IR

We need to select terms for a number of reasons: we need terms for a number candidate pools, plus terms for the genre screening vector and genre discrimination vector. This section discusses selection of these terms.

We have not spent much time on selection of terms for the topic pool. All we currently do is apply simple stemming operations to the category phrase supplied by the user.

To produce genre-related terms, we first tried the simple approach: looking up synonyms in various thesauri (e.g., Wordnet). However, this approach did not work: it suggested many bad terms and failed to suggest many good ones. So we turned to PMI-IR [7], an unsupervised learning algorithm for recognizing synonyms. PMI-IR measures the similarity of pairs of words by observing co-occurrences, assuming that co-occurrences are not statistically independent. Overall, PMI-IR shows promising results in determining whether two terms are synonyms.

Our overall term-selection process is complicated and includes a number of manual steps. Here are the highlights of what we do:

- *PMI-IR seeds*. We have hand-crafted two seed vectors for the PMI-IR algorithm. These vectors are small, containing fewer than five terms each. In the PMI literature, the terms in these vectors are called *problems words*. One vector contains seeds for the screening vector (e.g., "buying guide") and the other seeds for the discrimination vector (e.g., "comparative").
- *PMI-IR database*. We have implemented a wrapper around the Google API that runs the PMI-IR scoring algorithm against the results returned by a query. This algorithm takes the above "problem" vectors as inputs. Each run of this algorithm adds new scores to our *PMI-IR database*, a disk file in which we keep scores returned by the PMI-IR algorithm. These databases have grown to over 3,000 terms each. The update algorithm takes a long time to run, so we do not run it very often.
- *Genre vectors*. We compute the screening and discrimination vectors from the PMI-IR database. Basically, we take the top-scoring 100 terms from each database; however, we do introduce an element of "hand tuning" in the process. These vectors are recomputed only very occasionally, much fewer times than the PMI-IR database gets updated.
- *Genre term pools*. All of the genre pools (noun, verb, and adjective) are maintained by hand. These are quite small, currently containing 14 terms in total. When we recompute the genre screening vector, we keep an eye out for terms we might add to the genre pools, but in practice we rarely change these pools.

3. Evaluation

We ran a simple user study to test both the effectiveness and the generality of our system. Remember that we tuned our algorithms by testing recall against a "benchmark" product category (Section 2.1). While this technique allowed quick refinements, it is possible that these refinements work only for the benchmark categories and do not work generally. Thus, in addition to testing effectiveness, we wanted to test that our system generalized over a range of categories.

Our evaluation process was simple: we used a group of seven evaluators to measure the precision-at-ten of our system and a reasonable "baseline." Precision-at-ten ("P@10") measures the number of relevant documents contained in the top-ten ranking documents returned by a search engine ("10%" means one of those top-ten documents were relevant). It is used widely for evaluating Web search engines because it corresponds closely to what a Web searcher experiences (ten results on a "results page"). It is typically assumed that recall is not an issue within the top-ten results; that is, the corpus contains well over ten relevant results, and thus the problem is one of precision, not recall [15].

Our baseline system (a.k.a. BASE) was to submit the product-category phrase (defined below) augmented by "Naive genre expansion" to the [Google](#) search engine. "Naive genre expansion" means we attached the phrase "buying guide" to the category phrase. We added this phrase because the category phrase by itself typically fails to turn up *any* buying guides, which makes for an unreasonably poor baseline to compare our own system against.

We feel it is important to highlight this point. The baseline system is *not* a standard search engine: these perform so poorly that they are not worth measuring. Our improvements, then, are relative to a higher baseline.

We measured the performance of BASE and BGF on ten different product categories. For each category, a "category phrase" was chosen (see [Table 1](#)). In the case of BASE, this phrase was augmented with the phrase "buying guide" and submitted to the [Google](#) search engine. In the case of BGF, the category phrase was entered directly into the system.

Table 1. Product-category phrases.

Nickname	Category Phrase	Nickname	Category Phrase
BIN	binoculars	nb	notebooks
car	cars	pc	pressure cooker
dc	digital camera	saw	skilsaws
dvd	dvd players	vac	vacuum cleaners
mp3	mp3 players	wm	washing machines

Evaluating the results for ten categories is a fair amount of work. However, we wanted to measure a reasonably-large number of categories to test the generality of our approach. Five of the ten categories -- dvd, mp3, pc, saw, and wm -- were suggested by our evaluators *after* we froze our algorithm. We did this to further ensure that our study measured the generality of our approach. (It might be noted that two categories, dc and nb, were also suggested by our users, but these categories were secondary benchmarks for our tuning process and thus could be biased.)

The output of the two systems were combined randomly and duplicates eliminated. These combined lists were presented to the evaluators (on a category-by-category basis) for measurement. Our general instructions to the evaluators stated:

You are being asked make two judgments for us. First, you are being asked to judge whether or not each document is or is not a buying guide. Second, for those documents

you believe are buying guides, you are being asked to judge whether they are "good" or "bad."

The instructions to the evaluators went on to discuss in more detail what is a buying guide. We felt it was important that the evaluators judge buying guides in terms of fulfilling the information needs of the product-brokering process rather than in terms of specific characteristics. Thus, our instructions included paragraphs such as:

A buying guide is defined in terms of its intent. A buying guide is meant to help people at a certain point in the buying process. Imagine that you know nothing about digital cameras but you think you might want one. At the very beginning, you're less interested in the specific details of particular products and more interested in learning about the entire category. ... When looking at a document, ask yourself: Is this document useful given that know little about this category and I'm trying to learn about it?

In selecting evaluators, we tried to obtain diversity of backgrounds. Two were CS researchers, two were programmers (one in India), two were marketing professionals, and one was in the arts. The evaluators did not communicate amongst themselves regarding the evaluation. We computed our precision numbers on the basis of a simple majority amongst the evaluators. However, because of the open-ended nature of the judgments being made, we wanted to see consensus among the evaluators. We considered that a "consensus" had been reached when 2/3 of our evaluators agreed. For the simple "buying guide" judgment, consensus was reached on 85%; for "good buying guide", on 80%. Per-topic consensus numbers are given in the "cnss" row of [Table 2](#); these numbers do not suggest any anomalies or patterns. Overall, we were happy with the level of consensus reached amongst a diversified collection of evaluators.

Table 2. Results (precision and consensus).

Category	Simple			Good			Good	
	P@10/BASE	P@10/BGF	Cnss	P@10/BASE	P@10/BGF	Cnss	P@5/BASE	P@5/BGF
BIN	60%	80%	90%	40%	40%	76%	60%	40%
car	20%	60%	88%	0%	10%	85%	0%	20%
dc	40%	30%	81%	20%	0%	85%	0%	0%
dvd	20%	40%	86%	20%	30%	82%	20%	40%
mp3	40%	40%	86%	10%	10%	71%	0%	20%
nb	40%	40%	70%	10%	0%	81%	20%	0%
pc	10%	20%	100%	10%	20%	94%	20%	40%
saw	0%	60%	88%	0%	20%	75%	0%	20%
vac	20%	70%	83%	10%	60%	88%	20%	60%

wm	80%	70%	86%	30%	50%	68%	40%	80%
All	33%	51%	85%	15%	24%	80%	%	%

The results of the evaluation are given in [Table 2](#) and, graphically, in [Figure 2](#). We present the P@10 numbers for both the simple judgment ("is it a buying guide") and the "goodness" judgment ("is it a 'good' buying guide"). In the graphical figure, the simple judgment is represented by the vertical bars, while the goodness judgment is represented by the horizontal black lines across these bars.

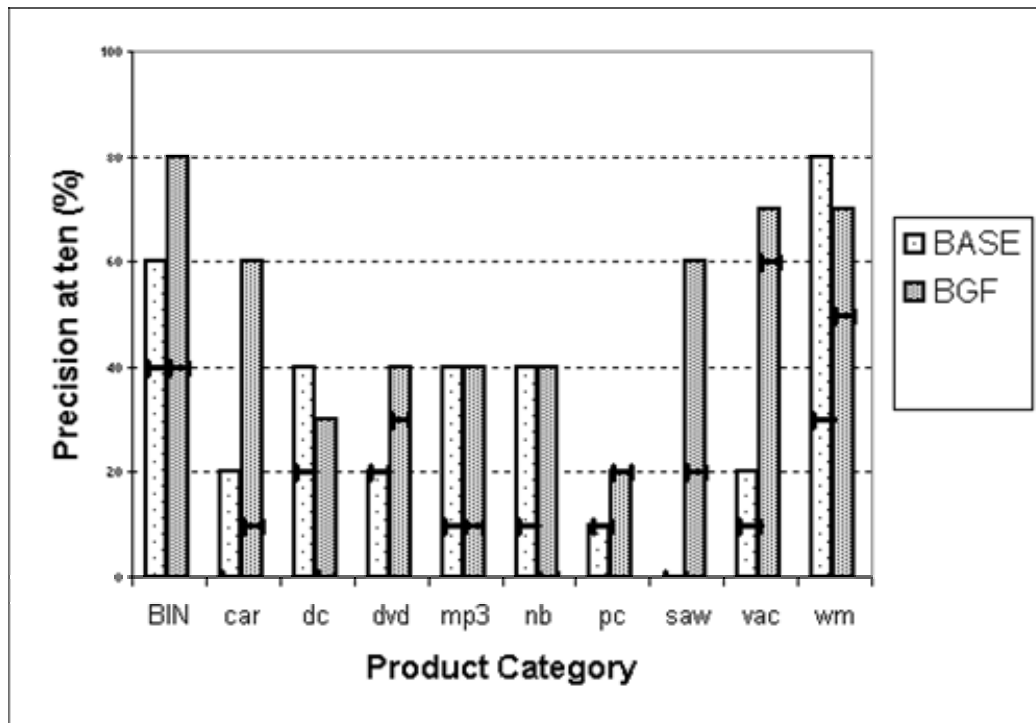


Figure 2. Results from user study.

In half of the product categories, our approach at least doubles the number of buying guides in the top-ten. In another category, our approach improves performance (on the simple test) by over 30%. In another two categories, performance stays the same, and in two categories performance degrades slightly. Overall, this study shows both effectiveness (especially given that our baseline is more than a simple search engine) and generality (especially given that some of our strongest improvements are for categories supplied by our users).

4. Related work

To our knowledge, there has been no work done related to genre classification of buying guides. Genre classification itself is an area of active research and quite different from topic classification. Finn et al. [3] identify genre as an important factor in retrieving useful documents. They investigate the performance of three different techniques and are primarily focused on domain transfer. Also, Karlgren [4] performs various experiments to investigate various word-based and text-based statistics for the purpose of improving retrieval results. In other work, Karlgren et al. [5] propose a genre scheme for Web pages and outline a search interface prototype that incorporates genre and content. Mostly, this work tries to do genre classification. Genre classification is basically the same as our genre screening, although we do not look at the document itself, treating the Metadata that is dynamically generated from the search engine as a proxy. The related work on genre classification could improve the recall of our screener.

Turney [7] describes the PMI-IR algorithm, how it compares to LSA, and shows experimental results on its performance on the TOEFL test. In our approach we apply PMI-IR to find related words for genre or topic information. The algorithm so far has produced good results, although we are not using the somehow better scoring function that would require a NEAR operator, which Google doesn't support.

Liu et al. [8] are also trying to use search engines to mine topic-specific knowledge on the Web. Their goal is to help people learn in-depth knowledge of topic systematically on the Web. However, the difference to our approach that they propose techniques to first identify sub-topics or salient concepts of a topic, and then find and organize those informative pages (like a book) instead of focusing on genre. We could see that it might be interesting to first use our technique to harvest a large set of buying guides for a given topic and then use their technique to organize this collection like a big buying guide book.

Davison et al. [9] analyze search engine traffic by focusing on the queries-to-results graph generated by a search engine. Mining that graph can show interesting relations. For instance, related URLs can be found, or for a given URL, the list of search query that yielded that URL can be retrieved. It might be interesting to use the proposed technique to enhance our query candidate pool: For instance, if we know the URL of a good buying guide, what were the queries people used to find that guide. We could then analyze these queries to gain insights on how to further improve our techniques for topic and genre expansion.

Giles et al. [10] describe *CiteSeer*, an autonomous citation indexing system for research papers. It uses crawling as its primary source to discover and harvest new information. They also use Web search engines and heuristics to locate papers: This is done by using queries for documents that contain certain words (e.g., "publications", "papers"). The obtained search results are then used as a seed list for their crawler. Although they realized the potential of search engines as a source for discovering new information, there queries are manually composed and fixed. They do not use the techniques described in this paper (e.g., query sequences using genre expansion) to avoid crawling. Instead they use the search engine only as a shortcut for some potential leads that need to be explored further by a crawler. We are going one step further since we are eliminating the crawling step completely. Our assumption is that everything useful is already crawled somewhere by a Web search engine. Instead we want to leverage crawling, preprocessing, and indexing from Web search engines and let them do the necessary ground work that enables us to find desired information faster with less effort.

The Movie Review Query Engine [11] helps to find movie reviews quickly. It is doing this by managing a database of known sites where reviews are located that are downloaded regularly to keep it up to date. Although there are focusing on a specific genre `movie reviews` there are many differences compared to our work. They are using manually edited lists of known sites to locate new information, instead of using genre expansion techniques combined with query sequences that are run against search engines to collect useful genre specific information. Although they do not perform genre screening, since they can rely on the fact that the sites they know contains reviews.

The idea itself of using query expansion techniques to improve precision is not new. Mitra et al. [12] show that adding words to queries via blind feedback, without any user input, can improve precision of such queries. We are also using expansion techniques to improve precision, but the methods we use are quite different. Where query expansion in the user scenario is typically a query refinement process combined with some form of relevance feedback (or not) to eventually find the desired piece of information, we are constructing sequences of queries up front based on query templates. Our process is fully automatic and the user is not involved the providing relevance feedback. Also, we do not download documents for feedback. Our genre screener uses "cheap" Metadata that provides already strong hints to perform a genre classification. Overall our application represents a new way of using query expansion and may stimulate new research on new query expansion techniques based on query templates and adaptive query sequencing.

5. Future work

We have identified a number of avenues for future work.

We are most intrigued by utilizing inter-trial feedback. In observing expert human Web searchers, we observe that the results of one search influence the formation of the next. For example, good results might suggest small refinements, while bad results might suggest a whole new approach. We would like to

integrate this and other feedback strategies into our system. We believe our template language provides a good foundation for such work.

We have performed some initial experiments with more aggressive topic expansion. These early attempts have helped for some topics, but have hurt for others -- for too many others. We are looking for techniques that more consistently help. In this regard, our work so far suggests that *topic discriminators* or some other mechanism to reduce topic drift will be important.

The system as presented in this paper performs Web searches on directly on behalf of consumers. However, in the context of a larger DSS for consumer e-commerce, studies have suggested that buying guides should also appear in strategic places in catalogs and e-commerce directory schemas. In this context, the guides are found *a priori*, and there is more emphasis on recall as a metric. We have begun to apply our technology to this context and it seems to work well.

We would like to evaluate our system applied to genres other than buying guides. To do this, we would like to further automate the compilation of genre terms. While this process is largely automated, there a few manual steps that remain. Also, as part of looking at other genres, we would like to experiment with other approaches to genre classification in our filtering step.

Finally, an on-going aspect of this work is reducing the number of queries that get issued. Our initial code would often issue many hundred queries per topic. We have reduced this to under a hundred, but would like to reduce it even further (while *improving* precision).

6. Conclusions

This paper presented a Web carnivore for finding buying guides on the Web. Finding buying guides is an important part of consumer e-commerce, and its a problem ill-served by existing technology. Our evaluation indicated that our approach significantly improves the ability of consumers to find buying guides using simple product-category phrases.

As mentioned earlier, finding buying guides is an instance of the larger problem of genre search. While we have not yet experimented in this direction, we believe our ideas will translate well to the more general context. In particular, we believe the our contributions in query templates, genre expansion, genre filtering, and our unsupervised approach of genre-related terms will all be applicable to other genres.

7. References

- [1] Barry G. Silverman and Mintu Bachann and Khaled Al-Akharas. *Implications of Buyer Decision Theory for Design of eCommerce Websites*. June, 2001.
- [2] Robert H. Guttman and Pattie Maes. *Agent-Mediated Integrative Negotiation for Retail Electronic Commerce. Lecture Notes in Computer Science*, pages 70-90, 1999.
- [3] A. Finn, N. Kushmerick, and B. Smyth. *Genre classification and domain transfer for information filtering. Proc. 24th European Colloquium on Information Retrieval Research*, Glasgow, 2002, 353-362.
- [4] Jussi Karlgren. *Stylistic experiments in information retrieval*. In T. Strzalkowski, editor, *Natural Language Information Retrieval*, Kluwer, 1999.
- [5] Jussi Karlgren, Ivan Bretan, Johan Dewe, Anders Hallberg, and Niklas Wolkert. *Iterative information retrieval using fast clustering and usage-specific genres*. In *Eight DELOS workshop on User Interfaces in Digital Libraries*, pages 85-92, Stockholm, Sweden, 1998.
- [6] Oren Etzioni. *Moving Up the Information Food Chain: Deploying Softbots on the World Wide Web. Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pages 1322-1326, 1996.
- [7] Peter D. Turney. *Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. Lecture Notes in Computer Science*, 2001.
- [8] Bing Liu, Chee Wee Chin, Hwee Tou Ng. *Mining Topic-Specific Concepts and Definitions on the Web. Proceedings of the twelfth international World Wide Web conference (WWW-2003)*, 20-24 May 2003, Budapest, Hungary.

[9] B. D. Davison, D. G. Deschenes, and D. B. Lewanda. *Finding Relevant Website Queries*. Twelfth international World Wide Web conference (WWW-2003), 20-24 May 2003, Budapest, Hungary.

[10] C. Lee Giles and Kurt Bollacker and Steve Lawrence. *CiteSeer: An Automatic Citation Indexing System*. Digital Libraries 98 - The Third {ACM} Conference on Digital Libraries, ACM Press, Pittsburg, PA, pages 89-98, June 1998.

[11] Movie Review Query Engine. <http://www.mrqe.com>

[12] Mandar Mitra and Amit Singhal and Chris Buckley. *Improving Automatic Query Expansion*. Research and Development in Information Retrieval, pages 206-214, 1998.

[13] Tara Calishain and Rael Dornfest. *Google Hacks: 100 Industrial-Strength Tips and Tricks*. O'Reilly & Associates, 2003.

[14] Monika Henzinger, Bay-Wei Chang, Brian Milch, Sergey Brin. *Query-free news search*. Twelfth international World Wide Web conference (WWW-2003), 20-24 May 2003, Budapest, Hungary.

[15] Mei Kobayashi, Koichi Takeda. *Information retrieval on the Web*. ACM Computing Surveys, Vol. , No. 32, pp. 144-173, Feb. 2000.