Measure-driven Keyword-Query Expansion

Nikos Sarkas
University of Toronto
nsarkas@cs.toronto.edu

Nilesh Bansal
University of Toronto
nilesh@cs.toronto.edu

Gautam Das
University of Texas at Arlington
gdas@cse.uta.edu

Nick Koudas
University of Toronto
koudas@cs.toronto.edu

ABSTRACT

User generated content has been fueling an explosion in the amount of available textual data. In this context, it is also common for users to express, either explicitly (through numerical ratings) or implicitly, their views and opinions on products, events, etc. This wealth of textual information necessitates the development of novel searching and data exploration paradigms.

In this paper we propose a new searching model, similar in spirit to faceted search, that enables the progressive refinement of a keyword-query result. However, in contrast to faceted search which utilizes domain-specific and hard-to-extract document attributes, the refinement process is driven by suggesting interesting *expansions* of the original query with additional search terms. Our *query-driven and domain-neutral* approach employs surprising word co-occurrence patterns and (optionally) numerical user ratings in order to identify meaningful top-k query expansions and allow one to focus on a particularly interesting subset of the original result set.

The proposed functionality is supported by a framework that is computationally efficient and nimble in terms of storage requirements. Our solution is grounded on Convex Optimization principles that allow us to exploit the pruning opportunities offered by the natural top-k formulation of our problem. The performance benefits offered by our solution are verified using both synthetic data and large real data sets comprised of blog posts.

1. INTRODUCTION

The amount of available textual data has been growing at a staggering pace. Besides the adoption of digital text (at the expense of paper) as the primary means of exchanging and storing unstructured information, this phenomenon has also been fueled by the transformation of the Web into an interactive medium. Web users have transcended their role as simple consumers of information and now actively participate in the generation of online content. Blogs, micro-blogging services, wikis and social networks are just examples of an online revolution taking place in social media.

In this context, it is also common for users to express, either explicitly or implicitly, their views and opinions on products, events, etc. For example, online forums such as customer feedback portals offer unique opportunities for individuals to engage with sellers or other customers and provide their comments and experiences. These interactions are typically summarized by the assignment of a numerical or "star" rating to a product or the quality of a service. Numerous such applications exist, like Amazon's customer feedback and Epinions. Any major online retailer engages one way or another to consumer-generated feedback.

But even if ratings are not explicitly provided, sentiment analysis tools [15] can identify with a high degree of confidence the governing sentiment (negative, neutral or positive) expressed in a piece of text, which in turn can be translated into a numerical rating. This capability enables the extraction of ratings from less formal reviews, typically encountered in blogs. Extending this observation, such tools can be employed to identify the dominant sentiment not only towards products but also events and news stories. Virtually any text document can be associated with a rating signifying the author's attitude towards some event.

This trend is reminiscent of the explosion in the availability of structured data that was witnessed during the 1990s and led to the introduction of OLAP tools [7]. Similarly, given the vast text repositories being accumulated, there is a pressing need for techniques to efficiently and effectively navigate them.

Faceted search [3] is one example of a successful technique for effective textual data navigation. In this searching paradigm, each document is associated with a set of well-defined categorical attributes (meta-data) referred to as *facets*. The meta-data domains are usually organized in a hierarchy, much like the *dimension* attributes of an OLAP application. Then, the result of a vanilla keyword query is refined by "drilling-down" the facet hierarchies. This interactive process effectively places and gradually tightens constraints on the meta-data, allowing one to identify and focus on a fraction of the documents that satisfy a keyword query. This slice of the original result set possesses properties that are considered interesting, expressed as constraints on the document meta-data.

One major drawback of the faceted search model is its reliance on domain-specific and hard-to-extract document attributes to facilitate data navigation. This limitation renders the approach inapplicable to document domains that exhibit high variance in their content, like blog posts or review collections of arbitrary items. Instead, we would like to suggest ways to refine the original search result in a *query-driven, domain-neutral* manner that is indifferent to the presence of document meta-data, other perhaps than the omnipresent user ratings.

To realize things concrete, consider a search for "Canon SD700" on a popular consumer electronics site. We would like to be able to identify on the fly product features, e.g., "lens" or "SLR capability", which are discussed in the reviews. This capability would be extremely helpful, especially for less prominent products with more obscure features, and would enable the refinement of the original query result to reviews that discuss a certain feature of interest.

In addition, we are interested in incorporating user feedback in the refinement process. Besides simply identifying product features, we would like to locate the ones, e.g., the camera's "lens" in our example, which are mentioned in reviews for which users have provided *high* on average ratings. Similarly, we should be able to automatically locate other features, for instance the camera's "SLR capability", which are discussed in reviews with *low* on average ratings. Finally, another helpful possibility is identifying features mentioned in reviews with *consistent*, unanimous ratings, independently of whether they are actually good, bad or neutral.

Such functionality is quite powerful; it provides goal-oriented navigation of the reviews, as we can interactively identify the product features (keywords) mentioned by satisfied consumers (high ratings), dissatisfied consumers (low ratings) or consumers that have reached a consensus (consistent ratings) and use them to refine the initial query result and drill down to examine the relevant reviews.

In this spirit, we propose a new data analysis and exploration model that enables the progressive refinement of a keyword-query result set. However, in contrast to faceted search which utilizes domain-specific and hard-to-extract document attributes, the refinement process is driven by suggesting interesting *expansions* of the original query with additional search terms extracted from the text collection. We refer to this iterative exploratory process as *Measure-driven Query Expansion*. More specifically,

- We introduce three principled scoring functions to quantitatively evaluate in a meaningful manner the interestingness of a candidate query expansion. Our first scoring function utilizes surprising word co-occurrence patterns to single out interesting expansions (e.g., expansions corresponding to product features discussed in reviews). Our second and third functions incorporate the available user ratings in order to identify expansions that define clusters of documents with either extreme ratings (e.g., product attributes mentioned in highly positive or negative on average reviews) or consistent ratings (e.g., features present in unanimous reviews).
- The query expansion functionality is supported by a unified, computationally efficient framework for identifying the k most interesting query expansions. Our solution is grounded on Convex Optimization principles that allow us to exploit the pruning opportunities offered by the natural top-k formulation of our problem.
- We verify the performance benefits of the solution using both synthetic data and large real data sets comprised of blog posts.

The remainder of the paper is organized as follows: In Section 2 we survey related work. Section 3 formally introduces the query expansion problem, while Section 4 describes our baseline implementation. Section 5 introduces our improved solution, whose superiority is experimentally verified in Section 6. Lastly, Section 7 offers our conclusions.

2. RELATED WORK

The availability of raw data on a massive scale requires the development of novel techniques for supporting interactive exploratory tasks. This necessity became evident with the proliferation of structured data and led to the development of OLAP tools [7].

An approach that resembles the OLAP cube, but for unstructured textual data, is the *faceted search* model [3] which extends the plain keyword search model. The documents are associated with orthogonal attributes with hierarchical domains, referred to as *facets*. These hierarchies are navigated in order to refine the result of a keyword query and focus on a subset that satisfies constraints on the attribute values. The facet domains and their hierarchical organization can be either set manually by an expert or automatically extracted from the document collection on indexing time [9]. These document attributes tend to be highly domain-specific. This is reasonable as very generic attributes would not offer significant opportunities for identifying interesting refinements. On the downside, the need for domain-specific attributes reduces the utility of faceted search in more general document domains.

Recent work [19] on improving faceted search proposed the use of "dynamic" facets extracted from the content of the documents: a result set can be further refined using frequent phrases appearing in the documents comprising the result. This approach is similar in spirit to the query expansion functionality that we propose, as it does not rely (exclusively) on the document meta-data to drive the refinement process. Nevertheless, the proposed query expansion technique suggests refinements in a manner that is more principled and elaborate (we suggest refinements by *optimizing* three diverse measures of interestingness, two of which utilize omnipresent user ratings) and scalable (we utilize pairs of tokens found in documents; [19] stores and manipulates phrases of length up to 5).

Most major search engines provide query expansion functionality in the form of query auto-completion. However, the suggested expansions are ranked in a *query-independent* and *data-agnostic manner*, such as based on their popularity in query logs [2].

Two of the scoring functions that we propose to drive the query expansion process rely on the presence of explicit or derived (through sentiment analysis [15]) numerical ratings. The Live.com search engine performs sentiment analysis on product reviews and identifies the reviewer's sentiment towards certain, *predefined* product features (e.g., ease of use, battery life) and cites the number of positive comments for each feature. However, our use of user ratings/sentiment is both more general and more elaborate, integrating sentiment into general query processing.

The computational framework that we developed in order to support the query expansion functionality leverages two powerful mathematical techniques: the Ellipsoid Method and Maximum Entropy reconstruction of a probability distribution.

The Ellipsoid Method is widely known in the context of Linear Programming that deals with the optimization of linear functions in the presence of linear constraints. Nevertheless, it is a more generic technique that can be used to solve Convex Optimization problems [5]. [4] surveys the history, operation and applications of the technique. The Principle of Maximum Entropy [8] is widely applied for fully reconstructing a probability distribution when only partial information about it is observed. The principle maintains that since we have no plausible reason to bias the distribution towards a certain form, the reconstructed distribution should be as uniform and "uninformative" as possible, subject to the observed constraints. The technique has been successfully applied for purposes similar to our own before [14, 16, 12, 13].

3. MEASURE-DRIVEN QUERY EXPANSION

Consider a collection of documents denoted by $\mathcal{D}$ and a set of words $\mathcal{W}$ that can appear in the documents of our collection. The composition of $\mathcal{W}$ depends on the application context and will not affect our subsequent discussion. As an example, it can simply be the full set or a subset of the words comprising the documents, the contents of a dictionary, or a well-specified set of words relevant to the application.

DEFINITION 1. *A* word-set *F is a set of r distinct words from*

$\mathcal{W}$, i.e., $F \in$ Powerset$(\mathcal{W})$ and $|F| = r$.

DEFINITION 2. *A collection of word-sets $\mathcal{F}_r(w_1, \ldots, w_l), l < r$ is comprised of all word-sets F with $F \in$ Powerset$(\mathcal{W})$ and $|F| = r$ and $w_1, \ldots, w_l \in F$.*

Thus, a word-set is a set of distinct words from $\mathcal{W}$, while collection $\mathcal{F}_r(w_1, \ldots, w_l)$ consists of all word-sets of size r, subject to the constraint that they must always contain words $w_1, \ldots, w_l$. The following example clarifies the definitions and illustrates how they relate to our goal of suggesting interesting query expansions.

EXAMPLE 1. *Let $\mathcal{D}$ be a set of documents and $\mathcal{W}$ the set of words appearing in $\mathcal{D}$, after the removal of common and stop words. A query that retrieves all the documents in $\mathcal{D}$ containing word w_1 is issued. Let us denote the result of this query as $\mathcal{D}_{w_1}$. At this point, we suggest a small number k of potential expansions of the original query by two additional keywords (the size of the expansion is a parameter). The candidate expansions are the word-sets belonging to $\mathcal{F}_3(w_1)$ (sets containing 3 words, one of which is definitely w_1). Therefore, our goal is to suggest k* expanded queries *(word-sets) from $\mathcal{F}_3(w_1)$ that can be used to refine the initial search result in a manner that is interesting and meaningful.*

As the above example illustrates, at each step the functionality of suggesting ways to *expand* the keyword query $Q = w_1, \ldots, w_l$ and *refine* the current set of results in an interesting manner can be formulated as the selection of k word-sets from a collection $\mathcal{F}_r(w_1, \ldots, w_l)$.

MEASURE-DRIVEN QUERY EXPANSION: *Consider a document collection $\mathcal{D}$ and a keyword query $Q = w_1, \ldots, w_l$ on $\mathcal{D}$. Let $\mathcal{D}_Q$ be the set of documents in $\mathcal{D}$ that satisfy the query. Q can either be the first query submitted to the system, or a refined query that was already proposed. Then, the problem of* query expansion *is to suggest k word-sets of size r from $\mathcal{F}_r(w_1, \ldots, w_l)$ that extend Q and can be used to focus on a particularly interesting subset of $\mathcal{D}_Q$.*

Notice that the ability to perform this operation implies the use of conjunctive query semantics, i.e., a document needs to contain *all* search terms in order to be considered a valid result.

So far in our discussion we have purposefully avoided mentioning what would constitute a query expansion that yields an "interesting" refinement of the initial result. In order to be able to single out k expansions to a keyword query, we need to define quantitative measures of interestingness. In what follows, we offer examples of interesting and meaningful query refinements that we subsequently formalize into concrete problems that need to be addressed.

3.1 Defining interesting expansions

EXAMPLE 2. *Consider a search for "digital camera" on a collection of product reviews. In the documents that contain these query terms and comprise the result set, we expect to encounter terms such as "zoom", "lens" or "SLR" frequently. The reason is that these terms are highly relevant to digital cameras and are therefore used in the corresponding reviews. Thus, the probability of encountering such terms in the result set is much higher than that of encountering them in general text, unrelated to digital cameras.*

We formalize this intuition with the notion of *surprise* [18, 6, 10]. Let $p(w_i)$ be the probability of word w_i appearing in a document of the collection and $p(w_1, \ldots, w_r)$ be the probability of words $w_1, \ldots, w_r$ *co-occurring* in a document[1]. If words $w_1, \ldots, w_r$ were unrelated and were used in documents *independently* of one another, we would expect that $p(w_1, \ldots, w_r) = p(w_1) \cdots p(w_r)$. Therefore, we use a simple measure to quantify by how much the observed word co-occurrences deviate from the independence assumption. For a word-set $F = w_1, \ldots, w_r$, we define

$$\text{Surprise}(F) = \frac{p(w_1, \ldots, w_r)}{p(w_1) \cdots p(w_r)}$$

[1]If considered appropriate, more restrictive notions of co-occurrence can also be used, e.g., the words appearing within the same paragraph in a document.

We argue that when considering a number of possible query expansions $\mathcal{F}_r(w_1, \ldots, w_l)$, word-sets with high surprise values constitute ideal suggestions: we identify coherent clusters of documents within the original result set that are connected by a common underlying theme, as defined by the co-occurring words.

The use of surprise (unexpectedness) as a measure of interestingness has also been vindicated in the data mining literature [18, 6, 10]. Additionally, the definition of surprise that we consider is simple yet intuitive and has been successfully employed [6, 10].

EXAMPLE 3. *Consider a collection comprised of 250k documents and query "table, tennis". Suppose that there exist 5k documents containing "table", 2k documents containing "tennis" and 1k documents containing both words "table, tennis". We easily compute that* Surprise*(table,tennis)=25.*

Let us compare the surprise value of two possible expansions: with term "car" (10k occurrences) and term "paddle" (1k occurrences). Suppose (reasonably) that "car" is not particularly related to "table, tennis" and therefore co-occurs independently with these words. Then, there exist 40 documents in the collection that contain all three words "table, tennis, car" (Figure 1). We compute that Surprise*(table,tennis,car)=25. While this expansion has a surprise value greater than 1, this is due to the correlation between "table" and "tennis".*

Now, consider the expansion with "paddle" and assume that 500 of the 1000 documents containing "table, tennis" also contain "paddle" ("table, tennis, paddle"). We compute that Surprise*(table,tennis,paddle)=3125. As this example illustrates, enhancing queries with highly relevant terms results in expansions with considerably higher surprise values than enhancing them with irrelevant ones.*

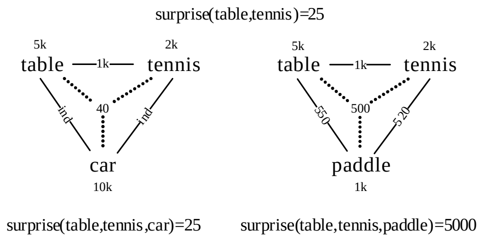
surprise(table,tennis)=25
5k
2k
table
1k
tennis
ind
40
ind
car
10k
5k
2k
table
1k
tennis
550
500
520
paddle
1k
surprise(table,tennis,car)=25
surprise(table,tennis,paddle)=5000

Figure 1: Query expansion based on the Surprise measure.

The maximum-likelihood estimates of the probabilities required to compute the surprise value of a word-set are derived from the textual data of the document collection $\mathcal{D}$ under consideration. We use $c(F) = c(w_1, \ldots, w_r)$ to denote the number of documents in a collection $\mathcal{D}$ that contain all r words of F. In the same spirit, we denote by $c(w_i)$ the number of documents that contain word w_i and $c(\bullet)$ the total number of documents in the collection. Then, we can estimate $p(w_1, \ldots, w_r) = c(w_1, \ldots, w_r)/c(\bullet)$ and $p(w_i) = c(w_i)/c(\bullet)$. Using these estimates, we can write

$$\text{Surprise}(F) = \frac{c(F)/c(\bullet)}{c(w_1)/c(\bullet) \cdots c(w_r)/c(\bullet)} \quad (1)$$

Therefore, one of the problems we need to address in order to suggest meaningful query expansions is, PROBLEM 1. *Consider a collection $\mathcal{D}$ of documents and a word-set $Q = w_1, \ldots, w_l$. We wish to determine the k word-sets $F \in \mathcal{F}_r(w_1, \ldots, w_l)$ with the maximum value of Surprise(F).*

Our first function does not assume the presence of meta-data in addition to the textual content of the documents. As a matter of fact, the query refinement solution based on the notion of surprise can be applied to any document corpus. Nevertheless, for our two subsequent formulations of interestingness, we make the assumption that every document is associated with a numerical "rating" from a small domain of values $s_1, \ldots, s_b$. As we discussed in Section 1, this could be the user-supplied rating to the product being reviewed in the document, or even a measure of how positive or negative is the sentiment expressed in the document [15].

The presence of numerical ratings associated with the documents points to two natural and meaningful ways of suggesting query expansions. Notice that every possible expansion $F \in \mathcal{F}_r(w_1, \ldots, w_l)$ of the initial query is associated with a subset of our document collection denoted as $\mathcal{D}_F$. Then, the *mean rating* and the *variance of the ratings* of the documents in $\mathcal{D}_F$ can be used to quantify the interestingness of the query expansion considered.

EXAMPLE 4. *Assume that our document collection is comprised of electronic gadget reviews, associated with the "star" rating that the reviewer assigned to the device. Then, if the original query is "Canon SD700", we can strengthen it with additional terms related to product features, so that the expanded query leads to a cluster of reviews with high on average "star" ratings, e.g., "Canon SD700 lens zoom". Such expansions would be highly interesting and aid users to quickly identify the product attributes for which other consumers are satisfied. Another alternative is to offer suggestions that would lead to groups of reviews with consistent ratings (low variance), thus facilitating the location of features for which a consensus on their quality has emerged.*

More formally, for a word set $F = w_1, \ldots, w_r$, let $\mathcal{D}_F$ be the documents in $\mathcal{D}$ that contain all words in F. Furthermore, let $c(F|s_i)$ be the number of documents in $\mathcal{D}_F$ that are rated with s_i. Then, the average rating of the documents in $\mathcal{D}_F$ is

$$\text{Average Rating: } Avg(F) = \sum_{i=1}^{b} s_i c(F|s_i) / \sum_{i=1}^{b} c(F|s_i) \qquad (2)$$

The variance of the ratings in $\mathcal{D}_F$ is equal to

$$\text{Var. of Ratings: } Var(F) = \sum_{i=1}^{b} s_i^2 c(F|s_i) / \sum_{i=1}^{b} c(F|s_i) - Avg(F)^2 \qquad (3)$$

Having demonstrated how to compute the mean value and the variance of the ratings associated with the result of a query expansion, we can formally state the two additional problems that we need to address.

PROBLEM 2. *Consider a collection $\mathcal{D}$ of documents rated with numerical values $s_1, \ldots, s_b$, and a word-set $Q = w_1, \ldots, w_l$. We wish to determine the k word-sets $F \in \mathcal{F}_r(w_1, \ldots, w_l)$ with either the* minimum *or the* maximum *value of $Avg(F)$.*

PROBLEM 3. *Consider a collection $\mathcal{D}$ of documents rated with numerical values $s_1, \ldots, s_b$, and a word-set $Q = w_1, \ldots, w_l$. We wish to determine the k word-sets $F \in \mathcal{F}_r(w_1, \ldots, w_l)$ with the* minimum *value of $Var(F)$.*

Hence, the problem of suggesting a few meaningful and interesting query expansions is formulated as three separate top-k problems (Problems 1, 2 and 3). Addressing Problem 1 produces the k word-sets/expansions with the highest surprise values (e.g., product features related to the query), Problem 2 expansions leading to documents with extreme ratings (e.g., features mentioned in highly positive on average reviews) and Problem 3 expansions leading to documents with consistent ratings (e.g., features mentioned in unanimous reviews).

Notice also that Problems 1,2 and 3 specify two input parameters in addition to the query Q to be expanded: the length of the expansion r and the number of required expansions k. The techniques that we subsequently develop can handle arbitrary values of those parameters. Appropriate values for r and k depend on the application. However, in practice the number of suggested expansions will normally be a fixed small constant (e.g. $k = 10$, see examples in [3]). Likewise, a query will be expanded by 1 or 2 additional terms, i.e., $r = l + 1$ or $r = l + 2$, where l is the length of query Q.

4. IMPLEMENTING QUERY EXPANSION

Let us concentrate on Problem 1 that involves identifying the k word-sets $F \in \mathcal{F}_r(w_1, \ldots, w_l)$ that maximize expression (1). The problem can be solved by computing the surprise value of every candidate word-set and identifying the top-k ones. We argue that the main challenge in solving the problem in that manner is calculating the surprise value of a candidate word-set.

The difficulty arises from our need to determine the value of $c(F)$, i.e., the number of documents that contain all words in F. Of course, expression (1) requires the number of occurrences in the corpus for single words, as well as the size of the corpus, i.e., counts $c(w_i)$ and $c(\bullet)$ respectively. However, for all practical purposes, these counts can be easily computed and manipulated. In order to compute a word-set's surprise value we need to focus our attention on determining the value of $c(F)$. This observation is also valid for Problems 2 and 3: in this case the challenge is to determine counts $c(F|s_i)$, i.e., the number of word co-occurrences conditioned on the numerical rating of the documents, which is a problem equally hard to determining $c(F)$.

In what follows, we argue that the naive approaches of *fully materializing* and retrieving on-demand (Section 4.1) all possible word co-occurrences $c(F)$ is infeasible for large document collections, while performing *no materialization* (Section 4.2) at all is extremely inefficient. Instead, we propose an alternative approach that is based on *estimating* co-occurrences $c(F)$ by utilizing materialized, lower-order co-occurrences of the words comprising F (Section 4.3).

4.1 Full Materialization

Suppose that we allow query expansions up to size $r = 5$. Then, the full materialization approach would need to generate, store and manipulate all two, three, four and five-way word co-occurrences. However, this is infeasible even for moderately large collections.

Let us demonstrate this using a simple example and concentrate on the computation part for the occurrences of word-sets of size four. The pre-computation of these occurrences would involve processing the collection one document at a time, generating all four-word combinations present in the document and temporarily storing them. Then, that data would need to be aggregated and stored. If on average a document contains 200 distinct words, each document would generate 65 million four-word tuples. If the corpus contains 10 million documents, we would need to generate and aggregate 650 billion tuples. As this trivial exercise demonstrates, the combinatorial explosion in the amount of generated data renders the explicit handling of high-order co-occurrences impossible.

4.2 No Materialization

While materializing all high-order word co-occurrences is impossible for large document collections, materializing no information at all would be extremely inefficient. As an example, consider a two-word query that we wish to expand with two additional words. Since we have no knowledge of four-way word co-occurrences, in order to evaluate the candidate expansions we would need to compute them on the fly. That would involve performing random I/O in order to retrieve all documents that satisfy the original query and process them in order to compute all two-way word co-occurrences in the documents (since two words out of the required four are fixed). It is evident that the I/O and CPU cost of this operation is prohibitively high. It would only make sense if the original result was comprised of a handful of documents, but in that case, the refinement of such a result wouldn't be necessary.

4.3 Partial Materialization

The proposed implementation of the query expansion functionality lies in-between the two aforementioned extremes, offering a solution that is both feasible (unlike full materialization) and efficient (unlike no materialization at all). To accomplish this, we propose the materialization of low-order word co-occurrences and their use in the subsequent *estimation* of higher-order word co-occurrences. This process involves the computation and storage of the occurrences of word-sets up to size l, for a reasonable value of l, and their use in the estimation of the occurrences of arbitrary size word-sets.

Based on this high-level idea, algorithm DIRECT (Algorithm 1) presents a unified framework for addressing problems 1, 2 and 3. Given a query Q, we need to suggest k expansions of size r that maximize either one of the three scoring functions. In order to do so, we iterate over all candidate word-sets $F \in \mathcal{F}_r(Q)$. For every candidate word-set F, we use the low-order co-occurrences (up to size l) of the words comprising F and *estimate* the number of documents $c(F)$ that contain all the words in F. For scoring functions (2) and (3) that require the co-occurrence values conditioned on the document rating, we derive a separate estimate for every rating value. Finally, the estimated high order co-occurrences are used to evaluate the interestingness of the candidate expansion and its value is compared against the current list of top-k expansions.

Algorithm 1 DIRECT

```
Input: Query $Q$, expansion size $r$, result size $k$

TopK $= \emptyset$
Iterator.init($Q$, $r$)

while Iterator.hasMore() do
    $\langle F$, Counts$_F \rangle$ = Iterator.getNext()

    for $i = 1$ to $b$ do
        $c(F|s_i)$ = Estimate($\langle$Counts$_F\rangle$)

    Score($F$) = Compute($c(F|s_1), \ldots, c(F|s_b)$)

    if Score($F$) $>$ TopK.threshold then
        Topk.update($F$)

return TopK
```

A natural question that arises at this point is why, unlike many other top-k query evaluation problems, we need to examine every candidate word-set in $\mathcal{F}_r(Q)$. Indeed, there exists a wealth of techniques that support the early termination of the top-k computation, before examining the entire space of candidates [11] and without sacrificing correctness. However, these algorithms require the scoring function to be monotone. It has been established that the co-occurrence estimation process does not exhibit monotonicity properties that can be exploited. Discussion related to the non-monotonicity of measures like the one adopted herein is available elsewhere [10, 6].

In order to realize algorithm DIRECT, we need to address in an efficient manner two challenges: (a) the progressive generation of candidate expansions and the retrieval of the corresponding low-order word co-occurrences and (b) their use in the estimation of the desired high-order co-occurrences.

4.3.1 Generation of candidate word-sets

The solution suggested pre-computes and manipulates the occurrences of word-sets up to size l. For most applications, the use of *two-word co-occurrences* presents the most reasonable alternative. Co-occurrences of higher order can be utilized at the expense of space and, most importantly, time. For the scale of the applications we envision, materializing co-occurrences of length higher than two is probably infeasible.

Two-word co-occurrences can be computed and stored efficiently as described in [1]. This involves the computation of a sorted list consisting of triplets $(w_i, w_j, \langle c(w_i, w_j)\rangle_s)$. Every such triplet contains the number of co-occurrences $\langle c(w_i, w_j)\rangle_s$ of words w_i and w_j for all document ratings s . A special tuple $(w_i, w_i, \langle c(w_i, w_i)\rangle_s)$ stores the occurrences of word w_i. If two words in $\mathcal{W}$ do not co-occur we simply don't store the corresponding tuple.

Then, one can use the tuples in the list of two-word co-occurrences and "chain" together pairs of words in order to progressively generate all word-sets of a collection $\mathcal{F}_r(Q)$, while at the same time retrieving the corresponding one-word and two-word counts. With some careful indexing and engineering this can be achieved without generating a candidate word-set more than once and with a minimum amount of I/O. Thus, we can efficiently implement the iterator utilized by algorithm DIRECT, which progressively retrieves word-set candidates and their low-order word co-occurrences.

Although we suggest the use of two-word co-occurrences and base the remainder of our presentation on this assumption, all of our techniques can be easily adapted to handle the use of higher-than-two word co-occurrences.

4.3.2 Estimation of Word Co-occurrences

Having established a methodology for efficiently generating the candidate word-sets and retrieving the corresponding single-word and two-word counts ($c(w_i)$, $c(w_i, w_j)$), we need to focus on how to utilize them in order to *estimate* higher-order co-occurrences $c(w_1, \ldots, w_r)$. The estimation approach that we use is based on the widely accepted Principle of Maximum Entropy [8] and has been successfully employed before [14, 16, 12, 13].

A basic observation is that a given word-set $F = w_1, \ldots, w_r$ defines a probabilistic experiment and consequently a probability distribution over the possible outcomes of the experiment: Given a document $D \in \mathcal{D}$, we identify which of the words w_i of F are contained in D. We associate with each word w_i a binary random variable W_i, such that $W_i = 1$ if $w_i \in D$ and $W_i = 0$ otherwise. Therefore, the experiment has $n = 2^r$ possible outcomes that are described by the joint probability distribution $p(W_1, \ldots, W_r)$.

If we had knowledge of that joint probability distribution we could easily estimate the number of co-occurrences $c(w_1, \ldots, w_r)$ *using its expected value:* $c(w_1, \ldots, w_r) = p(1, \ldots, 1)c(\bullet)$, *where* $c(\bullet)$ *is the number of documents in* $\mathcal{D}$. But although we do not know the distribution, we are not completely ignorant either: the pairwise co-occurrences and single-word occurrences at our disposal provide us with some knowledge about $p(W_1, \ldots, W_r)$.

EXAMPLE 5. *In order to ease notation, let us concentrate on a word-set* $F = a, b, c$ *of size* $r = 3$ *that defines an experiment with* $n = 8$ *possible outcomes and is described by the joint distribution* $p(A, B, C)$. *Our fractional knowledge about this distribution is in the form of simple linear constraints that we can derive from the pre-computed co-occurrences. For example, we can estimate that* $p(A = 1, B = 1) = c(a, b)/c(\bullet)$. *But* $p(A = 1, B = 1) = p(1, 1, 0) + p(1, 1, 1)$. *In the same manner* $p(A = 1) =$

$c(a)/c(\bullet) = p(1,0,0) + p(1,0,1) + p(1,1,0) + p(1,1,1)$.

Let us introduce some notation that will allow us to describe succinctly our knowledge about the joint distribution p. Each of the $n = 2^r$ possible outcomes of the experiment described by p is associated with a probability value. Recall that each outcome is described by a tuple $(W_1, \ldots, W_r)$, where variable W_i is either 0 or 1, signifying the existence or not of word w_i in the document. Then, let p_1 be the probability of outcome $(0, \ldots, 0, 0)$, p_2 of outcome $(0, \ldots, 0, 1)$, p_3 of $(0, \ldots, 1, 0)$ and so on and so forth so that p_n is the probability of outcome $(1, \ldots, 1)$. Therefore, the discrete probability distribution can be described by a vector $\mathbf{p} = (p_1, \ldots, p_n)^T$. Element p_n is used to provide the high-order co-occurrence as $c(w_1, \ldots, w_r) = p_n c(\bullet)$.

As we discussed in Example 5, each two-word co-occurrence count c_i provides us with some knowledge about the distribution in the form of a linear constraint $\mathbf{a}_i^T \mathbf{p} = c_i$. $\mathbf{a}_i$ is a vector with elements that are either 1 or 0, depending on the p_i's that participate in the constraint. This is also true for the single word occurrences, as well as the fact that the probabilities must sum up to 1. In total, we have at our disposal $m = 1+r+r(r-1)/2$ *independent* linear constraints: $r(r-1)/2$ from the two-word co-occurrences, r from the single-word occurrences and 1 from the fact that probabilities must sum up to 1. Therefore, our knowledge of the probability distribution can be represented concisely in matrix form as $A_{m \times n}\mathbf{p} = \mathbf{c}$, $\mathbf{p} \geq 0$, where each row of A and the corresponding element of $\mathbf{c}$ correspond to a different constraint.

EXAMPLE 6. *Let $F = a, b, c$ be a word-set. Then $r = 3$, $n = 8$ and $m = 7$. We can describe our knowledge of the distribution $\mathbf{p} \geq 0$ defined by F in matrix form $A\mathbf{p} = \mathbf{c}$, i.e.,*

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \\ p_8 \end{bmatrix} = \begin{bmatrix} c(a,b)/c(\bullet) \\ c(a,c)/c(\bullet) \\ c(b,c)/c(\bullet) \\ c(a)/c(\bullet) \\ c(b)/c(\bullet) \\ c(c)/c(\bullet) \\ c(\bullet)/c(\bullet) \end{bmatrix}$$

The constraints can also be viewed as a system of linear equations. However, the system $A\mathbf{p} = \mathbf{c}$ is under-defined, as there are less equations (constraints) than variables (p_i's). Therefore, this information by itself does not suffice to uniquely determine the joint probability distribution $\mathbf{p}$. It is important to note that we could inject additional constraints by utilizing information like the number of documents in $\mathcal{D}$ that contain word w_i, but not word w_j. The number of such documents is simply $c(w_i, \bar{w_j}) = c(w_i) - c(w_i, w_j)$. However, all such additional constraints can be derived by the original constraints defined by A and $\mathbf{c}$, therefore no supplementary knowledge can be gained in that manner.

When only partial information about a distribution is observed (such as in our case) the well-known information-theoretic *Principle of Maximum Entropy* [8] is widely applied in order to fully recover it [14, 16, 12, 13]. The principle maintains that since we have no plausible reason to bias the distribution towards a certain form, the reconstructed distribution should be as symmetric and "uninformative", i.e., as close to uniform $(1/n, \ldots, 1/n)$ as possible, subject to the observed constraints. In that manner, no additional information other than the observed constraints is injected.

More formally, the information entropy of discrete distribution $\mathbf{p}$ is defined as $H(\mathbf{p}) = -\sum_{i=1}^{n} p_i \log p_i$. The *unique* distribution $\mathbf{p}^*$ that maximizes $H(\mathbf{p})$ subject to $A\mathbf{p} = \mathbf{c}$ and $\mathbf{p} \geq 0$ is the *maximum entropy distribution* and satisfies the aforementioned desirable properties. Having computed the maximum entropy distribution $\mathbf{p}^*$, we estimate the desired high-order co-occurrence as $c(w_1, \ldots, w_r) = p_n^* c(\bullet)$.

EXAMPLE 7. *Let us revisit Example 3 (Section 3.1) where we compare the surprise value of two possible expansions for query "table, tennis": with irrelevant term "car" and highly relevant term "paddle". Using the two-way word co-occurrences depicted in Figure 1 (Section 3.1) and the Maximum Entropy Principle, we estimate that there exist 40 documents containing all three terms "table, tennis, car" (true value is 40) and 462 containing "table, tennis, paddle" (true value is 500). While the reconstruction process does not perfectly recover the original distribution, its accuracy is compatible with our goal of computing top-k expansions: we estimate that* Surprise*(table,tennis,car)=25 (true value is 25),* Surprise*(table,tennis,paddle)=2888 (true value is 3125), i.e., we are able to distinguish beyond doubt between interesting and non-interesting candidate expansions.*

Entropy maximization is a *convex optimization* problem [5]. Although there exists a variety of optimization techniques for addressing convex problems, the special structure of the entropy-maximization task, its importance and the frequency with which it is encountered in practice, has to led to the development of a specialized optimization technique known as *Iterative Proportional Fitting* (IPF) [8]. The IPF algorithm is an extremely simple iterative technique that does not rely on the heavyweight machinery typically employed by the generic convex optimization techniques and exhibits many desirable properties. In what follows, we offer a brief description of the algorithm and highlight some of its properties. More details are available elsewhere [8].

Initially, vector $\mathbf{p}$ is populated with arbitrary values. The choice of the starting point does not directly affect the speed of the algorithm, while the starting values do not even need to satisfy the constraints $A\mathbf{p} = \mathbf{c}$. Then, the algorithm iterates over the linear equality constraints (the rows of matrix A) and scales by an equal amount the variables of $\mathbf{p}$ participating in the constraint, so that the constraint is satisfied. This simple process, converges monotonically to the maximum entropy distribution.

A last point to note is that, due to the form of the entropy function $H(\mathbf{p})$, if we scale the right hand side of the problem constraints by a scalar $a > 0$, i.e., $A\mathbf{p} = a\mathbf{c}$, then the optimal solution will also be scaled by a, i.e., the optimal solution will be $a\mathbf{p}^*$. Therefore, we can scale the right hand side of the constraints by $c(\bullet)$ so that we directly use the low-order occurrence counts in the solution of problem (Example 6) and get the expected number of co-occurrences $c(w_1, \ldots, w_r) = p_n^* c(\bullet)$ directly from the value of p_n^*. The IPF procedure is also unaffected by this scaling.

5. WORKING WITH BOUNDS

The query expansion framework implemented by algorithm DIRECT (Algorithm 1) incrementally generates all candidate query expansions and for each candidate F it solves an entropy maximization problem to estimate the co-occurrence count $c(F)$ from lower-order co-occurrences. Hence, the bulk of the computational overhead can be attributed to the maximum-entropy-based estimation step. In this section, we focus our attention on reducing this overhead. Let us begin by making two important observations that will guide us towards an improved solution.

- First, the IPF procedure, or any other optimization algorithm for that matter, "over-solves" the co-occurrence estimation problem, in the sense that it completely determines the maximum entropy distribution $\mathbf{p}^*$, relevant to the candidate expansion under consideration. However, recall that we only utilize a single element of $\mathbf{p}^*$, namely p_n^*, which provides the required co-occurrence estimate (Section 4.3.2). The remaining $n-1$ values of the optimal solution vector $\mathbf{p}^*$ are of no value to our application.

- Second, besides requiring a single element from the maximum entropy distribution $\mathbf{p}^*$, we do not always require its exact value: in most cases a bound around p_n^* would work equally well. Remember that we only need to determine the top-k most interesting query expansions. Therefore, a bound on the estimated co-occurrence count, *which translates into a bound on the score of the expansion considered*, might be sufficient for *pruning* the candidate: if the upper bound on the score of the candidate is less than the scores of the top-k expansions that we have computed so far, we do not need to evaluate its exact score as it can never make it to the top-k.

Hence, we require much less than what the IPF technique, or any other optimization algorithm provides: we only need bounds on the value of p_n^* (high-order co-occurrence) instead of the exact solution $\mathbf{p}^*$ of the entropy maximization problem.

In order to exploit this opportunity we develop ELLIMAX, a novel *iterative* optimization technique. ELLIMAX is capable of computing the exact value of p_n^*, but does so by deriving *progressively tighter bounds* around it. As we elaborate in Section 5.1, each iteration of the ELLIMAX technique results in a tighter bound around p_n^*. This is a property that neither IPF, nor any other optimization algorithm possesses.

The unique properties of the ELLIMAX technique are leveraged by algorithm BOUND (Algorithm 2), an improved framework for computing the top-k candidate expansions. Algorithm BOUND processes candidate expansions one at a time, as algorithm DIRECT does. However, it utilizes the ELLIMAX technique to progressively bound the co-occurrences of candidate expansion F and consequently its score. The algorithm stops processing candidate F as soon the upper bound on its score becomes less than the score of the expansions currently in the top-k heap. In the case that a candidate cannot be pruned since it needs to enter the top-k heap, the ELLIMAX technique is invoked until the bound on its score tightens enough to be considered a singular value.

The advantage offered by algorithm BOUND over algorithm DIRECT presented before is its ability to prune candidate expansions that cannot appear in the top-k result, *without incurring the full cost of computing their exact score*. In most cases, only a handful of ELLIMAX iterations should be sufficient for eliminating a candidate from further consideration.

Algorithm 2 BOUND

```
Input: Query $Q$, expansion size $r$, result size $k$

TopK $= \emptyset$
Iterator.init($Q$, $r$)

while Iterator.hasMore() do
    $\langle F$, Counts$_F \rangle =$ Iterator.getNext()
    Score$_{\min}(F) = -\infty$, Score$_{\max}(F) = +\infty$

    while Score$_{\max}(F) -$ Score$_{\min}(F) > \epsilon$ do
        for $i = 1$ to $b$ do
            Tighten $[c_{\min}(F|s_i), c_{\max}(F|s_i)]$ using ELLIMAX

        Tighten [Score$_{\min}(F)$, Score$_{\max}(F)$] using $[c_{\min}(F|s_i), c_{\max}(F|s_i)]$

        if Score$_{\max}(F) <$ TopK.threshold then
            Break

    Topk.update($F$)

return TopK
```

Before we proceed with the presentation of the ELLIMAX technique, let us briefly verify that a bound on the estimated number of word co-occurrences is actually translated into a bound on the interestingness of the candidate expansion, for all three scoring functions that we consider.

- Surprise (1): It is not hard to see that a bound on the estimated number of co-occurrences $a \leq c(F) \leq b$ bounds surprise between $\frac{a/c(\bullet)}{c(w_1)/c(\bullet)\cdots c(w_r)/c(\bullet)} \leq \text{Surprise}(F) \leq \frac{b/c(\bullet)}{c(w_1)/c(\bullet)\cdots c(w_r)/c(\bullet)}$.

- Average Rating (2): Let assume that we have obtained bounds $a_i \leq c(F|s_i) \leq b_i$. Additionally, let us also assume that ratings s_i are positive. Then, in order to get an upper bound on $Avg(F)$ we need to set the numerator to its largest possible value and the denominator to its smallest possible. Reasoning in the same manner for the lower bound, we obtain $\frac{\sum_i s_i a_i}{\sum_i b_i} \leq Avg(F) \leq \frac{\sum_i s_i b_i}{\sum_i a_i}$. A similar process can provide us with bounds when some of the s_i's are negative.

- Variance of Ratings (3): The variance equation is comprised of two terms. We can compute bounds on the first term using the process we just described, while the second is simply $Avg(F)^2$, for which we demonstrated how to derive bounds.

5.1 Progressive Bounding of Co-occurrences

The iterative ELLIMAX technique that we develop for providing progressively tighter bounds around the estimated number of high-order co-occurrences p_n^* is based on the principles underlying the operation of the Ellipsoid algorithm for solving Convex Optimization problems. We briefly survey these topics in Section 5.1.1, as they are vital for understanding of the ELLIMAX technique, substantiated in Section 5.1.2.

5.1.1 Convex optimization and the Ellipsoid method

The entropy maximization problem, whose optimal solution $\mathbf{p}^*$ provides the estimate p_n^* of the desired high-order co-occurrences, is a *convex optimization*[2] problem [5].

$$\min -H(\mathbf{p}), \;\; A\mathbf{p} = \mathbf{c}, \;\; \mathbf{p} \geq 0 \tag{4}$$

DEFINITION 3. *A set* $D \in \mathbb{R}^n$ *is convex if* $\forall x, y \in D$ *and* $0 \leq \theta \leq 1$, $\theta x + (1-\theta)y \in D$.

Less formally, a set D is convex if any line segment connecting two of its points lies entirely in D. Therefore, convex domains are continuous subsets of $\mathbb{R}^n$ without any "cavities".

DEFINITION 4. *A function* $f : D \to \mathbb{R}$, $D \subseteq \mathbb{R}^n$ *is* convex *if its domain* D *is a convex set and* $\forall x, y \in D$ *and* $0 \leq \theta \leq 1$, $f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y)$.

It is not hard to demonstrate that both the optimization function $-H(\mathbf{p})$ and the *feasible area* of the problem, defined by constraints $A\mathbf{p} = \mathbf{c}, \mathbf{p} \geq 0$ are *convex*. A desirable property of convex optimization problems is the following.

THEOREM 1. *[5] Any locally optimal point of a convex optimization problem is also globally optimal.*

A corollary of this important property is that most convex optimization problems have a unique optimal solution. This is one of the reasons that convex optimization is a tractable problem, since it allows the development of efficient, greedy iterative techniques (descent and interior-point) that progressively move towards the optimal solution. Nevertheless, these algorithms are *oblivious* of their current distance to the optimum.

There exists, however, a different class of optimization techniques known as localization methods that progressively bound the

[2]We formulate the entropy maximization problem as a minimization problem in order to conform with the established optimization terminology.

optimal solution within a shrinking container. When the container becomes small enough, a point inside it is used to approximate the solution. Most prominent among this class of algorithms is the *Ellipsoid Method*, which utilizes an ellipsoidal container to bound the optimal solution.

The Ellipsoid Method can be used to solve convex optimization problems of the form $\min f(\mathbf{p})$ subject to $A\mathbf{p} \leq \mathbf{b}$, where f is a convex function. At a high level, it utilizes an ellipsoid in order to contain the problem's optimal solution. An ellipsoid $\mathcal{E}$ is described by means of a matrix P and its center $\mathbf{o}$, so that the points inside it satisfy $\mathcal{E} = \{\mathbf{p} : (\mathbf{p} - \mathbf{o})^T P^{-1}(\mathbf{p} - \mathbf{o}) \leq 1\}$.

The algorithm commences with an ellipsoid $\mathcal{E}_0$ that contains the entire feasible region, as defined by the problem constraints $A\mathbf{p} \leq \mathbf{b}$. Then, at each iteration t, it queries an *oracle* which provides the algorithm with a hyperplane passing through the current ellipsoid's center $\mathbf{o}_t$. The hyperplane is described by a vector $\mathbf{h}_t$ perpendicular to the hyperplane. Using this representation, the points $\mathbf{p}$ on the hyperplane satisfy $\mathbf{h}_t^T(\mathbf{p} - \mathbf{o}_t) = 0$. The guarantee we are offered by the oracle is that the optimal solution $\mathbf{p}^*$ is located on the positive side of the hyperplane, i.e., $\mathbf{h}_t^T(\mathbf{p}^* - \mathbf{o}_t) \geq 0$. Having obtained this *separating hyperplane*, the algorithm computes the unique *minimum volume ellipsoid* $\mathcal{E}_{t+1}$ which contains the half of the current ellipsoid $\mathcal{E}_t$ that lies on the positive side of the hyperplane. Notice that the invariant maintained by this procedure is that the current ellipsoid $\mathcal{E}_t$ *always* contains the optimal solution $\mathbf{p}^*$. When the ellipsoid becomes small enough, we can use its center as an adequate approximation to the optimal solution.

Although the iterations of the ellipsoid algorithm might seem heavyweight, they are actually efficient and come with a theoretical guarantee concerning the amount of shrinkage they accomplish. In case the cost function f is differentiable, as is the case for the entropy function, the separating hyperplane is simply minus the *gradient* of the function at the ellipsoid center, i.e., $\mathbf{h}_t = -\nabla f(\mathbf{o}_t)$ [5]. In the event the ellipsoid center lies outside the feasible region, any violated constraint (rows from $A\mathbf{p} \leq \mathbf{b}$) can serve as a separating hyperplane. Having obtained the separating hyperplane, determining the next ellipsoid involves a few simple matrix-vector multiplications involving matrix P_t and vectors $\mathbf{h}_t$, $\mathbf{o}_t$. The total cost of an iteration is $O(n^2)$ (n is the problem dimensionality) and reduces the containing ellipsoid's volume by at least $e^{-\frac{1}{2(n+1)}}$ [4].

5.1.2 *The* ELLIMAX *technique*

The ellipsoid method offers a unique advantage not provided by IPF or any other convex optimization technique. Namely *the progressively shrinking ellipsoid can be utilized to derive bounds on any element* $\mathbf{p}_i^*$ *of the optimal solution*. However, this process is far from straightforward. There are a number of *significant challenges* that need to be addressed *efficiently* in order to substantiate the ELLIMAX optimization technique to used by algorithm BOUND.

- Remove equality constraints: As we discussed in our overview of the ellipsoid method for convex optimization, it is applicable in the presence of inequality constraints of the form $A\mathbf{p} \leq \mathbf{b}$. However, our optimization problem (4) contains equality constraints that need to be efficiently removed.
- Update bounds around p_n^*: We need to work out the details of how to translate the ellipsoidal bound around the optimal solution $\mathbf{p}^*$ into a one-dimensional bound for p_n^*.
- Identify a small starting ellipsoid: The ellipsoid method requires a starting ellipsoid that completely covers the feasible region. Since our motivation for utilizing the ellipsoid method is to derive a tight bound around the optimum as fast as possible, it is crucial that we initiate the computation with the smallest ellipsoid possible, subject to the constraint that its determination should be rapid.

In the remainder of the Section, we focus on providing efficient solutions for these tasks. The efficiency of the solutions is also experimentally verified in Section 6.4.1.

Removing the equality constraints and moving to the λ-space

The ellipsoid method cannot handle equality the equality constraints of the entropy maximization problem ($A\mathbf{p} = \mathbf{c}$) because such constraints cannot provide a separating hyperplane in the case the ellipsoid's center violates one of them. Therefore, the problem needs to be transformed into an equivalent one that does not feature equality constraints. In order to perform this transformation we utilize linear algebra tools [20].

DEFINITION 5. *The* null space *of matrix* $A_{m\times n}$*, denoted by* $\mathcal{N}(A)$*, is the space of vectors* $\mathbf{r}$ *that satisfy* $A\mathbf{r} = 0$*. The null space is a* $(n - m)$*-dimensional subspace of* $\mathbb{R}^n$*.*

LEMMA 1. *A vector* $\mathbf{p}$ *with* $A\mathbf{p} = \mathbf{c}$ *can be described as the sum of two vectors* $\mathbf{p} = \mathbf{q} + \mathbf{r}$*, where* $\mathbf{q}$ *is any vector that satisfies* $A\mathbf{q} = \mathbf{c}$ *and* $\mathbf{r}$ *is a vector that lies in the null space of* A*.*

The null space of A, like any vector space, can be described by an *orthonormal basis*, i.e., a set of orthogonal, unit vectors. Such a basis can be computed using one of a number of available techniques, like the Singular Value Decomposition. The basis consists of $g = n - m$, n-dimensional vectors.

LEMMA 2. *Let* $\mathbf{e}_1, \ldots, \mathbf{e}_g$ *with* $g = n - m$ *be an orthonormal basis for* $\mathcal{N}(A)$*. Then,* $A\mathbf{r} = 0 \Leftrightarrow \mathbf{r} = \sum_{i=1}^{g} \lambda_i \mathbf{e}_i$*, with* $\lambda_i \in \mathbb{R}$*. To ease notation, let* $U = [\mathbf{e}_1 \ldots \mathbf{e}_g]$ *be a matrix whose columns are the basis vectors of* $\mathcal{N}(A)$*. Then,* $A\mathbf{r} = 0 \Leftrightarrow \mathbf{r} = U\boldsymbol{\lambda}$*.*

The aforementioned lemmas allow us to eliminate the constraints $A\mathbf{p} = \mathbf{c}$ by simply enforcing $\mathbf{p} = \mathbf{q} + U\boldsymbol{\lambda}$. Observe that a vector $\boldsymbol{\lambda} \in \mathbb{R}^g$ fully defines a vector $\mathbf{p} \in \mathbb{R}^n$. Then, we can substitute $\mathbf{p}$ in the cost function with $\mathbf{q} + U\boldsymbol{\lambda}$ and express it as a function of $\boldsymbol{\lambda}$. We will denote the entropy function expressed as a function of $\boldsymbol{\lambda}$ with $H_\lambda(\boldsymbol{\lambda})$. It is easy to show that $-H_\lambda(\boldsymbol{\lambda})$ is also convex. Additionally, the constraint $\mathbf{p} \geq 0$ becomes $U\boldsymbol{\lambda} \geq -\mathbf{q}$. Putting it all together, the optimization problem that we need to address is:

$$\min -H_\lambda(\boldsymbol{\lambda}),\ U\boldsymbol{\lambda} \geq -\mathbf{q} \tag{5}$$

Problem (5) is equivalent to problem (4), but (a) it does not contain equality constraints and (b) it is of smaller dimensionality $g = n - m$, since $\mathbf{p} \in \mathbb{R}^n$ while $\boldsymbol{\lambda} \in \mathbb{R}^{n-m}$. *We will say that the original problem (4) lies in the* p*-space, while the transformed problem (5) lies in the* λ*-space.*

The remaining question is whether this transformation can be computed efficiently. The answer is positive and this is due to a simple observation: *matrix* A *is always the same for all instances of problem (4)*. Although the constraints are different for every instance, what varies is the values of vector $\mathbf{c}$. This is intuitive, as the only change from instance to instance are the low-order co-occurrence counts populating vector $\mathbf{c}$ and not the way that variables p_i are related, which is described by matrix A (Section 4.3.2).

Therefore, the null space of A and consequently matrix U can be pre-computed using any one of available techniques [20]. We can also use pre-computation to assist us in determining an appropriate vector $\mathbf{q}$, such that $A\mathbf{q} = \mathbf{c}$. A solution to this under-defined system of equations can be computed by means of either the QR or LU decomposition of A [20]. As with the null space, the decomposition of A can be pre-computed.

Translating the ellipsoidal bound in the λ-space to a linear bound in the p-space

At each iteration, the ellipsoid method provides us with an updated ellipsoid $\mathcal{E} = \{\boldsymbol{\lambda} : (\boldsymbol{\lambda} - \mathbf{o})^T P^{-1}(\boldsymbol{\lambda} - \mathbf{o}) \leq 1\}$. The challenge we need to address is how to translate this ellipsoidal bound in the λ-space into a linear bound for variable p_n in the p-space. The following theorem demonstrates how this is done.

THEOREM 2. *Let $\mathcal{E} = \{\boldsymbol{\lambda} : (\boldsymbol{\lambda} - \mathbf{o})^T P^{-1}(\boldsymbol{\lambda} - \mathbf{o}) \leq 1\}$ be the bounding ellipsoid in the λ-space. Let us also define vector $\mathbf{d} = (e_{1n}, \dots, e_{gn})^T$. Then, variable p_n in the p-space lies in*

$$p_n \in [q_n + \mathbf{d}^T\mathbf{o} - \sqrt{\mathbf{d}^T P \mathbf{d}}, q_n + \mathbf{d}^T\mathbf{o} + \sqrt{\mathbf{d}^T P \mathbf{d}}]$$

PROOF. We can verify that $p_n = q_n + \mathbf{d}^T\mathbf{o} + \mathbf{d}^T\mathbf{m}$, with $\mathbf{m}^T P^{-1}\mathbf{m} \leq 1$, using the p-space to λ-space mapping. Due to the positive-definiteness of P, there exists real, invertible matrix V such that $P = VV^T$. Thus, the constraint on $\mathbf{m}$ becomes $(V^{-1}\mathbf{m})^T\ (V^{-1}\mathbf{m}) \leq 1$. We set $\mathbf{v} = V^{-1}\mathbf{m}$ and have that $p_n = q_n + \mathbf{d}^T\mathbf{o} + (V^T\mathbf{d})^T\mathbf{v}$, with $\mathbf{v}^T\mathbf{v} \leq 1$. In other words, $\mathbf{v}$ is a vector of length at most 1 and p_n is maximized when product $(V^T\mathbf{d})^T\mathbf{v}$ is maximized. In order to accomplish this, vector $\mathbf{v}$ must be aligned with vector $V^T\mathbf{d}$ and its length must be set to its maximum value, i.e., 1. Hence, the value of $\mathbf{v}$ maximizing p_n is $\mathbf{v} = V^T\mathbf{d}/||V^T\mathbf{d}||$. By substituting (and using the fact that $\sqrt{\mathbf{x}^T\mathbf{x}} = ||\mathbf{x}||$) we obtain p_n's maximum value. $\square$

Based on the aforementioned result, the translation of the ellipsoidal bound in the λ-space to a linear bound for variable p_n in the p-space can be computed analytically by employing a few efficient vector-vector and matrix-vector multiplications.

Identifying a compact starting ellipsoid

Identifying a compact starting ellipsoid is an integral part of the ELLIMAX technique. Nevertheless, determining such an ellipsoid presents a performance (computation cost)/efficiency (ellipsoid size) trade-off. For example, we can formulate the problem of identifying the minimum volume ellipsoid (known as Löwner-John ellipsoid) covering the feasible region of problem (5) as a convex optimization problem [5]. However, the resulting problem is harder than the problem we need to solve. Therefore, it does not make sense to determine the best possible starting ellipsoid at such cost.

In what follows, we present an *analytical* procedure for identifying a compact starting ellipsoid. The procedure is efficient, as it involves a handful of lightweight operations, and is comprised of three steps: (a) identifying an axis-aligned bounding box around the feasible region in the p-space, (b) using the box in the p-space to derive an axis-aligned bounding box in the λ-space and (c) covering the latter bounding box with the smallest possible ellipsoid.

Let us concentrate on the first step of the procedure. The feasible region in the p-space is described by the linear constraints $A\mathbf{p} = \mathbf{c}$ and $\mathbf{p} \geq 0$. Every row of A along with the corresponding element of $\mathbf{c}$, define a linear constraint that sums some of the p_i's so that they equal a value c (Example 6). Then, since $\mathbf{p} \geq 0$, the elements of $\mathbf{p}$ that participate in the constraint cannot be greater than c, since that would require some element in the constraint to be negative.

Therefore, by iterating once over all the constraints in $A\mathbf{p} = \mathbf{c}$ we can get an upper bound for every element p_i. At that point we can make another pass over the constraints in order to determine a lower bound tighter than 0: by setting all variables but one to the maximum value that they can assume, we can use the linear equality to derive a lower bound for our free variable. By repeating this process for all constraints and variables we identify lower bounds for all variables p_i.

EXAMPLE 8. *Suppose that we only have 2 constraints $p_1 + p_2 = 2$ and $p_2 + p_3 = 4$. From the first constraint we can derive that $p_1 \leq 2$ and $p_2 \leq 2$, while the second one provides $p_2 \leq 4$ (for which we already acquired a better bound) and $p_3 \leq 4$. We then use these upper bounds to derive the lower bounds $p_1 \geq 2 - p_{2,max} \Rightarrow p_1 \geq 0$, $p_2 \geq 2 - p_{1,max} \Rightarrow p_2 \geq 0$ and $p_3 \geq 4 - p_{2,max} \Rightarrow p_3 \geq 2$.*

We now need to translate the derived bounds $a_i \leq p_i \leq b_i$ into bounds in the λ-space. Let $\mathbf{v}_i = (0, \dots, \underset{i}{1}, \dots, 0)^T$ be a basis vector of the p-space. Then,

$$\mathbf{p} = \mathbf{q} + \sum_{i=1}^{g} \lambda_i \mathbf{e}_i \text{ and } \mathbf{p} = \sum_{i=1}^{n} p_i \mathbf{v}_i \Rightarrow \mathbf{q} + \sum_{i=1}^{g} \lambda_i \mathbf{e}_i = \sum_{i=1}^{n} p_i \mathbf{v}_i$$

Let us multiply both sides of the equation with vector $\mathbf{e}_k$, keeping in mind that vectors $\mathbf{e}_i$ form an orthonormal basis.

$$\mathbf{q} + \sum_{i=1}^{g} \lambda_i \mathbf{e}_i = \sum_{i=1}^{n} p_i \mathbf{v}_i \Rightarrow \lambda_k = -\mathbf{q}^T\mathbf{e}_k + \sum_{i=1}^{n} p_i(\mathbf{v}_i^T\mathbf{e}_k)$$

Getting a bound on λ_k is now straightforward. Each constraint $a_i \leq p_i \leq b_i$ is multiplied by $(\mathbf{v}_i^T\mathbf{e}_k)$, so that

$$a_i(\mathbf{v}_i^T\mathbf{e}_k) \leq p_i(\mathbf{v}_i^T\mathbf{e}_k) \leq b_i(\mathbf{v}_i^T\mathbf{e}_k), \text{ if } (\mathbf{v}_i^T\mathbf{e}_k) > 0$$
$$b_i(\mathbf{v}_i^T\mathbf{e}_k) \leq p_i(\mathbf{v}_i^T\mathbf{e}_k) \leq a_i(\mathbf{v}_i^T\mathbf{e}_k), \text{ if } (\mathbf{v}_i^T\mathbf{e}_k) < 0$$

Adding these constraints up and further adding $-\mathbf{q}^T\mathbf{e}_k$ to both the lower and upper bound, we get a bound on λ_k.

Lastly, the minimum volume ellipsoid $\mathcal{V}$ covering an axis-aligned box can also be analytically determined. Intuitively, $\mathcal{V}$ is an axis-aligned ellipsoid whose center coincides with the box's. The length of $\mathcal{V}$'s axis parallel to the λ_i-axis is equal to $\sqrt{g}$ times the box's extent across that axis.

THEOREM 3. *Consider a g-dimensional, axis-aligned box so that $\lambda_i^{\min} \leq \lambda_i \leq \lambda_i^{\max}$. Then, the minimum volume ellipsoid $\mathcal{V} = \{\boldsymbol{\lambda} : (\boldsymbol{\lambda} - \mathbf{o})^T P^{-1}(\boldsymbol{\lambda} - \mathbf{o}) \leq 1\}$ covering the box is*

$$\mathbf{o} = (\frac{\lambda_1^{\min} + \lambda_1^{\max}}{2}, \dots, \frac{\lambda_g^{\min} + \lambda_g^{\max}}{2})^T$$
$$P_{ii} = g\frac{(\lambda_i^{\max} - \lambda_i^{\min})^2}{4} \quad \textit{and} \quad P_{ij} = 0$$

6. EXPERIMENTAL EVALUATION

6.1 Summary of Contributions

Before we proceed with our evaluation, let us recapitulate our contributions and the algorithms introduced and subsequently evaluated. In Section 3 we formulated the problem of refining a keyword query result by suggesting interesting and meaningful query expansions. We introduced three scoring functions to quantitatively evaluate the interestingness of candidate query expansions and single out the k expansions with maximum (or minimum) score values (Problems 1, 2 and 3).

In Section 4 we observed that the main challenge in evaluating the score of a candidate expansion F lies in computing the number of documents $c(F)$ containing all the words comprising F and justified the use of two-word co-occurrences in order to estimate $c(F)$. To this end, we introduced Algorithm DIRECT which incrementally generates all candidate query expansions and for each candidate F it (a) solves an entropy maximization problem (many in the case of Problems 2 and 3) to estimate $c(F)$ (Section 4.3.2) and (b) uses this estimate to compute F's score. Algorithm DIRECT uses the specialized IPF technique in order to solve the entropy maximization problem.

Algorithm BOUND (Section 5) improves upon DIRECT by exploiting the natural top-k formulation of the query expansion problem. By leveraging the novel ELLIMAX technique (neither IPF nor any other algorithm can be used) in solving the entropy maximization problem, it can progressively bound $c(F)$ and subsequently the score of a candidate expansion F and eliminate it as soon its upper bound is lower than the candidates in the top-k heap.

While we introduced ELLIMAX for use with algorithm BOUND and apply it on the entropy maximization problem, it should be noted that it is a *novel* and *generic* optimization technique and can be applied on *any convex optimization problem* in order to derive progressively tighter bounds around parts of its optimum.

6.2 Expansion Length and Problem Dimensionality

Let us now briefly discuss how the length r of the candidate query expansions affects the overall complexity of our problem. This length defines the size of the resulting entropy maximization problem which in turns contributes to the difficulty of our problem.

Estimating the occurrences for a word-set of size r requires the solution of a convex optimization (entropy maximization) problem involving $n = 2^r$ variables and $m = 1+r+r(r-1)/2$ constraints (Section 4.3.2). In the case of the ELLIMAX technique we introduced in Section 5.1, removal of the m equality constraints results in an optimization problem that lies in the g-dimensional λ-space, where $g = n-m$. Table 1 summarizes the values of these variables for three reasonable values of r.

	n	m	g
$r=3$	8	7	1
$r=4$	16	11	5
$r=5$	32	16	16

Table 1: Word-set size effect on problem size.

As the length of the candidate expansions increases, so does the complexity of the entropy maximization problem. This has an adverse effect on the running time of both the IPF (used by DIRECT) and the ELLIMAX (used by BOUND) algorithms. An interesting observation is that for $r = 3$, the ELLIMAX technique handles a 1-dimensional problem. Effectively, the feasible region of the convex optimization problem in the λ-space is a line segment. In this case, the ELLIMAX technique collapses to a bisection method that bounds the optimal solution by iteratively cutting the feasible line segment in half.

6.3 Experimental Setting

Both the IPF process and the ELLIMAX algorithm are optimization techniques that iteratively converge towards the problem's optimum. The methods terminate when they are able to provide an approximation to the optimal solution within a desired degree of accuracy. In our experiments, we set this accuracy to 10^{-6} in absolute terms, i.e., we declared convergence when $|p_n - p_n^*| < 10^{-6}$.

In order to guarantee that the convergence condition is met by the IPF, we required that two iterations fail to change the values of all variables p_i by more than 10^{-6} [14]. At this point, the IPF's most recent estimate for p_n^* is returned to Algorithm 1. The ELLIMAX technique derives progressively tighter bounds around the required optimal value p_n^*. Unless the method is terminated by algorithm BOUND, due to the algorithm being able to prune the candidate, the ELLIMAX technique stops when the bound around p_n^* becomes smaller than 10^{-6}. Then, the middle-point of the interval is used to approximate the true value of p_n^* with accuracy within 10^{-6}.

In our timing experiments we concentrated on the CPU time required by algorithms DIRECT and BOUND, since the I/O component is identical for both algorithms. The CPU time of algorithm DIRECT is consumed by IPF calls, while the CPU time of BOUND is consumed by ELLIMAX iterations. Our test platform was a 2.4Ghz Opteron processor with 16GB of memory, although both optimization techniques have miniscule memory requirements. The methods were implemented in C++, while the vector and matrix operations were supported by the ATLAS linear algebra library [17].

6.4 Experimental Results

6.4.1 Evaluation of the ELLIMAX Technique

The initialization of the ELLIMAX technique includes the removal of the equality constraints and the transition to the λ-space where the algorithm operates, as well as the computation of a compact starting ellipsoid. An iteration involves the computation of a cutting hyperplane, its use in updating the ellipsoidal container and the derivation of a linear bound for variable p_n^*. Each iteration guarantees a reduction of the ellipsoid's volume by at least a certain amount. Although this reduction cannot be directly translated into a reduction in the size of the bound around p_n^*, it provides us with some information about the effectiveness of each iteration. Table 2 summarizes this information.

	Initialization (μs)	Iteration (μs)	Vol. reduction %
$r=3$	1.8	0.5	50
$r=4$	4.6	2.6	8
$r=5$	12.00	7.9	3

Table 2: Performance of the ELLIMAX method.

As we can observe, both the initialization and iteration operations are extremely efficient. Nevertheless, as the problem size increases (expansion length r), two sources contribute in the method's performance degradation: (a) the iterations are more expensive and (b) more iterations are required in order to decrease the bound around p_n^* by an equal amount. Such trends are consistent as the value of r increases.

6.4.2 Synthetic data

For our first set of experiments, we applied both algorithm DIRECT and algorithm BOUND to a stream of 100k synthetically generated candidate expansions and measured the total CPU time (spent in IPF calls by DIRECT and ELLIMAX iterations by BOUND) required in order to identify the top-10 expansions. As we discussed, since the purpose of computing expansions is to present them to a user, only a handful of them need to be computed.

A candidate expansion is generated by assigning values to the low-order co-occurrences that describe it. Equivalently, for every candidate we need to assign values to the constraint vector $\mathbf{c}$ of its corresponding entropy maximization problem (4). For Problems 2 and 3, the required low-order co-occurrences conditioned on the document rating, are generated for each rating value independently.

Our first data generator, denoted by U, works as follows. Instead of directly populating vector $\mathbf{c}$, its values are generated indirectly by first producing the underlying probability distribution $\mathbf{p}$ that the maximum entropy process attempts to reconstruct. Recall that low-order co-occurrences appearing in vector $\mathbf{c}$ are related to the underlying distribution that we estimate through constraints $A\mathbf{p} = \mathbf{c}$. Then, since A is fixed and $\mathbf{p}$ known, we can generate vector $\mathbf{c}$. These constraints are the *only* information used by IPF and ELLIMAX in order to estimate p_n^*. The data distribution vector $\mathbf{p}$ is populated with uniformly distributed values from a specified range ($[5, 10000]$ in our experiments). We experimented with other skewed data distributions and the results were consistent with the ones we subsequently present.

During our experimentation we observed a dependence of the performance of the IPF technique (and consequently of algorithm DIRECT) on the degree of *pairwise correlation* between the words comprising a word-set. We quantify the correlation between words

w_i and w_j by employing ratios $c(w_i, w_j)\ /c(w_i)$ and $c(w_i, w_j)/c(w_j)$: the closer the value of these ratios is to 1, the more frequently words w_i and w_j co-occur in the document collection. As we will discuss in more detail, we observed that the convergence rate of IPF was adversely affected by the presence of strong correlations.

Our second data generator, denoted by C, synthetically produces co-occurrences with a varying degree of pairwise correlation. Its first step is to randomly generate two-word co-occurrences $c(w_i, w_j)$ from a uniform distribution over the interval $[100, 1000]$. These co-occurrences are subsequently used in the generation of the single word occurrences $c(w_i)$: for every word w_i, the ratio $\max_j c(w_i, w_j)$ $/c(w_i)$ is sampled uniformly from an interval $[a, b]$. Using this ratio we can derive a value for $c(w_i)$. Controlling the interval $[a, b]$ allows us to control the degree of pairwise correlation we inject. Therefore, we experimented with 5 data sets: $C0$, where we sample from the interval $[0.01, 0.99]$, $C1$ from the interval $[0.01, 0.25]$, $C2$ from $[0.25, 0.50]$, $C3$ from $[0.50, 0.75]$ and $C4$ from $[0.75, 0.99]$. Intuitively, data set $C0$ is comprised of candidates with a varying degree of correlation, while data sets $C1$ to $C4$ contain candidates that exhibit progressively stronger correlations.

The experimental results for Problem 1 are presented in Figure 2. The left bar chart depicts the results for expansions of size $r = 3$, while the right bar chart for expansions of size $r = 4$. At a high level, it is evident that algorithm BOUND clearly outperforms DIRECT by orders of magnitude. BOUND*'s superior performance can be attributed to the fact that once a few of highly-surprising candidates are encountered, the pruning of subsequent candidates is relatively easy, requiring only few* ELLIMAX *iterations*.

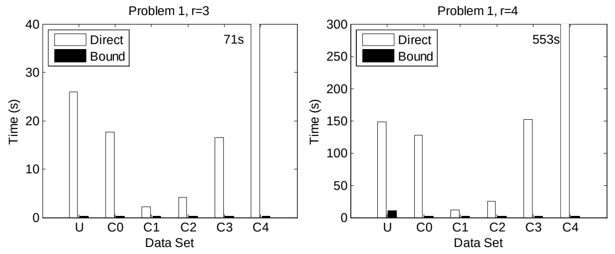

Figure 2: Problem 1 performance on synthetic data sets.

Additionally, the performance of DIRECT in Figure 2 verifies our previous observation, i.e., IPF's performance decreases as the degree of two-way correlation between the words increases (data sets $C1$ to $C4$). The absence of pairwise correlations suggests that the underlying text is mostly uniform. Given that the maximum entropy principle underlying the operation of IPF is essentially a uniformity assumption, this effect is understandable: the more the ground truth about the data distribution (pairwise co-occurrences) deviates from the technique's assumptions (uniformity), the slower its convergence.

We consider this behavior as a major drawback of the IPF technique and algorithm DIRECT since we are, by the definition of our problem, interested in discovering words that are highly correlated and define meaningful and significant clusters of documents. This trend in the performance of IPF is consistent throughout our experimental evaluation.

For Problems 2 and 3 we used three ratings: 0, 1 and 2. Our experimental results are depicted in Figures 3 and 4 respectively. For $r = 3$ the BOUND algorithm outperforms DIRECT by a large margin. For $r = 4$ the image is mixed, although the BOUND algorithm performs clearly better for all data sets other than $C1$ and $C2$. Recall that these two data sets exhibit exclusively very low correlations, a scenario which as we discussed is beneficial for IPF and algorithm DIRECT. In practice we expect hardly ever to encounter close to uniform (uncorrelated) text. Correlations are prevalent in real data sets and this points to the advantage of our proposal.

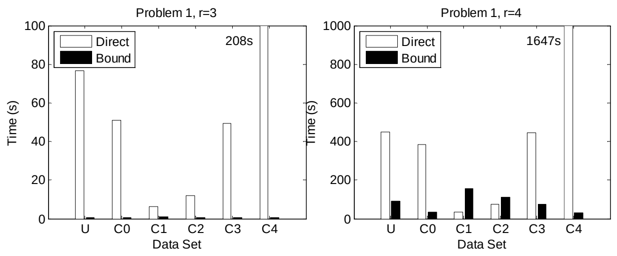

Figure 3: Problem 2 performance on synthetic data sets.

The reason for the, perhaps surprising, gap in the performance of the BOUND algorithm from $r = 3$ to $r = 4$ is the following. Due to the complexity of scoring functions (2) and (3), we need to derive relatively tight bounds on the estimated co-occurrences in order for them to be translated into a sufficient for pruning bound around the expansion's score. But, in order to achieve these bounds, BOUND for $r = 4$ must perform more, yet less efficient ELLIMAX iterations than for $r = 3$ (Section 6.4.1).

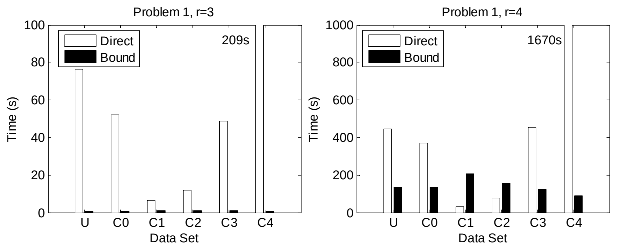

Figure 4: Problem 3 performance on synthetic data sets.

We also performed experiments to examine whether enabling the use of more rating values adversely affects the pruning performance of the BOUND. Figure 5 presents its performance for documents with 3, 5 and 10 possible ratings, under the U data set. The running time scales linearly, as desired: 3, 5 and 10 instances of the ELLIMAX technique need to run and provide bounds for counts $c(F|s_i)$ in parallel, therefore this linear scale-up is expected. A super-linear increase would imply a reduction in pruning efficiency, but this was not observed. The performance of IPF also scales linearly and is therefore omitted from the graphs. This result was consistent for all synthetic data sets.

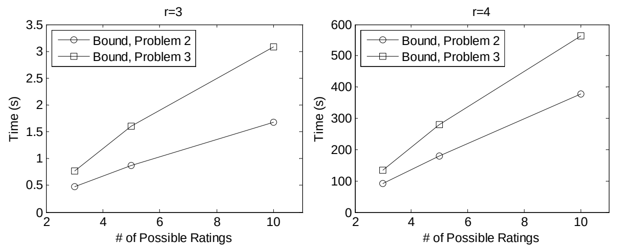

Figure 5: Performance vs number of possible ratings.

6.4.3 Real data

We also performed experiments using massive, real data sets comprised of blog posts. To evaluate the performance of the techniques on Problem 1, we used a sample consisting of 2.7 million posts, retrieved daily during February 2007. In order to reduce the search space of the algorithms and prune uninteresting co-occurrences, we only maintained counts $c(w_i, w_j)$ such that $c(w_i, w_j)/c(w_i) > 0.05$ and $c(w_i, w_j)/c(w_j) > 0.05$. We posed random single-keyword queries and present the average CPU time required by the techniques in Figure 6. As it is evident, the BOUND method has a clear advantage over DIRECT, offering significantly better performance.

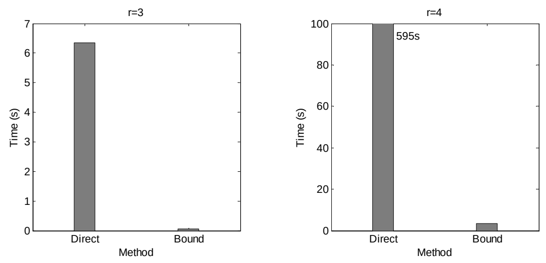

Figure 6: Problem 1 performance on real data.

In order to evaluate performance on Problems 2 and 3, we used a sample of 68 thousand posts from the day of 13/02/2008. We used a custom sentiment analysis tool based on [15] to associate each post with a rating (0 for negative, 1 for neutral and 2 for positive). As before, we removed uninteresting co-occurrences to reduce the search space in a similar manner and posed random single-word queries. Figure 7 presents the average CPU time required by the algorithms to solve Problem 2. The results for Problem 3 followed the same trend and are therefore omitted. As it is evident, the pruning opportunities exploited by the BOUND algorithm enable us to deliver superior performance to DIRECT.

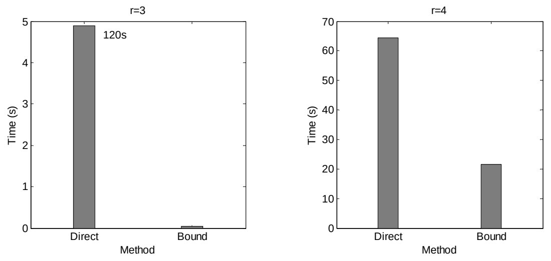

Figure 7: Problem 2 performance on real data.

6.4.4 *Qualitative results on real data*

Finally, we conclude our experimental evaluation with a brief case study regarding the query expansions suggested by our technique. We used blog post data for the day of 26/08/2008, processed by the same sentiment analysis tool as before. Only stopwords were removed. Below we present the top-3 expansions suggested using the scoring functions discussed in Section 3.1, for a small sample of product-related queries, in order to verify our claims from Sections 1 and 3 and demonstrate the utility of the query expansion framework. Notice that the expansions are indeed comprised of terms (mostly products and product features) relevant to the original query, e.g., "toner" (feature) for query "hp printer" (product), or "eee netbook" (product line) and "acer netbook" (competitor) for query "asus" (manufacturer).

The expansions suggested would be invaluable for interacting with the underlying document collection. For example, we query the *generic and diverse* blog post collection for the day of 26/08/2008 in order to unearth discussion, reviews and opinions about "hp printer(s)". Without further assistance from the query expansion framework, manual browsing of hundreds matching blog posts would be the sole option for discovering more useful information about the subject. The expansions allow us to rapidly identify and focus on a particularly interesting slice of the relevant posts. For instance, expansions with high surprise value identify a subset of posts discussing particular features of interest, e.g, "cartridges". Alternatively, we have the option to focus on posts with low average score, discussing a problematic aspect of the product, e.g., "graphics".

Query	Surprise	Max. Avg. Score	Min. Avg. Score
hp printer	cartridge laser ink	network mobile ebay	toner cartridge graphics
nikon digital	autofocus viewfinder cmos viewfinder cmos autofocus	slr standard slr tone slr stabil	canon click camera file manual images
asus	acer netbook eee netbook acer eee	laptop camera mobile model core mobile	eee install eee linux laptop pc

7. CONCLUSIONS

Motivated by the accumulation of vast text repositories and the limitations of existing techniques, we introduced a new data analysis and exploration model that enables the progressive refinement of a keyword-query result set. The process is driven by suggesting expansions of the original query with additional search terms and is supported by an efficient framework, grounded on Convex Optimization principles.

Acknowledgements: The work of Gautam Das was supported in part by the US National Science Foundation under grants 0845644 and 0812601, unrestricted gifts from Microsoft Research and Nokia Research, and start-up funds from the University of Texas at Arlington.

8. REFERENCES

[1] N. Bansal, F. Chiang, N. Koudas, and F. W. Tompa. Seeking stable clusters in the blogosphere. In *VLDB*, 2007.
[2] Z. Bar-Yossef and M. Gurevich. Mining search engine query logs via suggestion sampling. *PVLDB*, 1(1), 2008.
[3] O. Ben-Yitzhak, N. Golbandi, N. Har'El, R. Lempel, A. Neumann, S. Ofek-Koifman, D. Sheinwald, E. J. Shekita, B. Sznajder, and S. Yogev. Beyond basic faceted search. In *WSDM*, 2008.
[4] R. G. Bland, D. Goldfarb, and M. J. Todd. The ellipsoid method: A survey. *Operations Research*, 29(6), 1981.
[5] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 1st edition, 2004.
[6] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlations. In *SIGMOD Conference*, 1997.
[7] S. Chaudhuri and U. Dayal. An overview of data warehousing and olap technology. *SIGMOD Record*, 26(1), 1997.
[8] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2nd edition, 2006.
[9] W. Dakka, P. G. Ipeirotis, and K. R. Wood. Automatic construction of multifaceted browsing interfaces. In *CIKM*, 2005.
[10] W. DuMouchel and D. Pregibon. Empirical bayes screening for multi-item associations. In *KDD*, 2001.
[11] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *PODS*, 2001.
[12] C. Faloutsos, H. V. Jagadish, and N. Sidiropoulos. Recovering information from summary data. In *VLDB*, 1997.
[13] V. Markl, N. Megiddo, M. Kutsch, T. M. Tran, P. J. Haas, and U. Srivastava. Consistently estimating the selectivity of conjuncts of predicates. In *VLDB*, 2005.
[14] T. Palpanas, N. Koudas, and A. O. Mendelzon. Using datacube aggregates for approximate querying and deviation detection. *IEEE Trans. Knowl. Data Eng.*, 17(11), 2005.
[15] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. *CoRR*, cs.CL/0205070, 2002.
[16] D. Pavlov, H. Mannila, and P. Smyth. Probabilistic models for query approximation with large sparse binary data sets. In *UAI*, 2000.
[17] See homepage for details. Atlas homepage. http://math-atlas.sourceforge.net/.
[18] A. Silberschatz and A. Tuzhilin. On subjective measures of interestingness in knowledge discovery. In *KDD*, 1995.
[19] A. Simitsis, A. Baid, Y. Sismanis, and B. Reinwald. Multidimensional content exploration. *PVLDB*, 1(1), 2008.
[20] G. Strang. *Linear Algebra and its Applications*. Brooks Cole, 4th edition, 2005.