

# Refining Search Queries from Examples Using Boolean Expressions and Latent Semantic Analysis

David Johnson, Vishv Malhotra,  
Peter Vamplew and Sunanda Patro  
School of Computing, University of Tasmania  
Private Bag 100, Hobart Tasmania, Australia  
dgjohnso@utas.edu.au, Vishv.Malhotra@utas.edu.au,,  
Peter.Vamplew@utas.edu.au, spatro@postoffice.utas.edu.au

## ABSTRACT

*This paper describes an algorithm whereby an initial, naïve user query to a search engine can be subsequently refined to improve both its recall and precision. This is achieved by manually classifying the documents retrieved by the original query into relevant and irrelevant categories, and then finding additional Boolean terms which successfully discriminate between these categories. Latent semantic analysis is used to weight the choice of these extra search terms to make the resulting queries more intuitive to users.*

## 1. INTRODUCTION

Search engines provide an invaluable service to users searching for information on the Web. A well designed search query exhibits both good recall and precision; retrieving a large number of documents relevant to the user's requirements, whilst minimising the number of irrelevant documents presented to the user.

However constructing such a query manually is a non-trivial process —the user needs to select search terms which are general enough to ensure that the majority of relevant documents are matched, whilst still specific enough to eliminate most irrelevant documents. The most effective queries may require combinations of search terms which are unlikely to be evident to the user. This paper proposes and tests an algorithm which automatically synthesises such queries on the basis of the user's classification of the documents retrieved by an initial, naïve query devised by the user. It is shown that through this process even simple, single-word queries can be used to generate complex queries which exhibit good recall and precision characteristics.

The goal of aiding users in constructing effective search queries is not unique to our work. Oyama et al [1] have suggested the use of pre-defined "spice" terms as a means for narrowing initial searches specified by the user. The choice of spice terms is based on a general categorisation of the topic of the search. For example, in searching for recipes using beef as an ingredient, an initial search using the keyword *beef* will result in a relatively low percentage of relevant documents being returned. Augmenting this search with the spice term (*ingredients & !season & !description*) | *tablespoon* results in a far more

effective search. This approach requires that a range of suitable spice terms have pre-identified, and that the user search for suitable spice terms to be added to their initial query terms. In contrast the approach taken in this paper is to use the initial query to retrieve a set of documents which are then classified as irrelevant or relevant by the user. These documents are then used as examples to dynamically construct appropriate "spice" terms for the query.

This approach of using examples to refine a query has been widely used in image retrieval systems. The survey article by Kherfi et al [2] lists a number of search engines which use feature matching against examples to retrieve suitable images from a database. The Atlas WISE system developed by Kherfi [3] uses both positive and negative example images to further refine the retrieval process. These image retrieval algorithms typically rely on extracting relevant features from the images, and then explicitly comparing each image in the database against each example image. In contrast our algorithm attempts to learn a classifier which differentiates between the relevant and irrelevant examples. This expression is then used to augment the original query — in this way no direct comparisons are made against the example documents.

Many possibilities exist for the structure of the classifier learnt from the example documents. For example: neural networks, probabilistic classifiers, decision trees classifiers, support vector classifiers. However most of these techniques can not readily be used to create search terms for contemporary search engines. The algorithm presented here derives a classifier in the form of a Boolean expression, as this will be directly applicable to most search engines, as long as certain constraints on the form of the expression are satisfied.

The structure of the paper is as follows. In Section 2 we introduce terminology used in the paper. Section 3 provides the basic algorithm for query construction. The following section provides a more focused description of those items of the algorithm that we have improved to generate refined queries. Section 5 lists two case studies to illustrate the advantages of the changes described in Section 4. The discussion in Section 6 critically reviews these results. The paper is concluded in Section 7 with some general remarks about the approach and future directions.

## 2. TERMINOLOGY

A *literal* is an expression that is either a single term or its negation. A *minterm* is a sequence of literals combined by the Boolean operator AND ( $\&$ ). In this paper we will use the Google convention to treat AND ( $\&$ ) as the lowest precedence and implied (not explicitly written) operator. A sequence of minterms combined by a Boolean OR ( $\mid$ ) operation is the disjunctive normal form (DNF) of the Boolean expression. A *maxterm* is a sequence of literals combined by Boolean OR operations. A Boolean expression is in conjunctive normal form (CNF) if it is made of maxterms combined by operator AND. For further information about Boolean expressions the reader may wish to see [4] as an excellent reference.

A *search query* is a suitably compacted representation of a Boolean expression with at least one positive literal in each minterm. To help presentation of the algorithms in the later sections, we define a selection operation  $\sigma$ : For a set of textual documents,  $D$ , and a search query,  $Q$ , expression  $D \sigma Q$  will be used to denote a search by query  $Q$  over set  $D$ . An operational interpretation of expression  $D \sigma Q$  is as follows:

Case: $Q$ is term	$\{doc \mid doc \in D \text{ and term occurs in document } doc\}$
Case: $Q$ is $\neg$ term	$\{doc \mid doc \in D \text{ and term does not occurs in document } doc\}$
Case: $Q$ is $(R \ \& \ S)$	$(D \ \sigma \ R) \ \sigma \ S$
Case: $Q$ is $(R \ \mid \ S)$	$(D \ \sigma \ R) \ \cup \ (D \ \sigma \ S)$

The current implementation of the query synthesis algorithm requires the initial search query for retrieving example relevant and irrelevant documents to be a minterm comprising of positive literals. In the rest of this paper, the terms in the initial query minterm are assumed to occur in each document. The real documents may be missing some terms, as some search engines use text around the links to a document to index the linked document.

## 3. CONSTRUCTING QUERIES FROM EXAMPLES

The query synthesis process begins with an initial imprecise query to a search engine, generated by the user. This query returns links to web resources. The user downloads these resources and indicates for each resource if it is of interest or not. In what follows, we assume all retrieved resources to be text documents (in this case it may be possible to classify a resource based on the short précis returned by the search engine, rather than downloading the entire document). The categorised sets of documents will be referred to as: *Relevant* and *Irrelevant*.

The next stage is to apply dimension reduction techniques to remove superfluous terms to make the processing more efficient. The steps in this process are

removal of stop terms, stemming of the terms, removal of terms that occur in very few documents, and removal of terms that occur in virtually every document [5, 6]. Each document in the sets *Relevant* and *Irrelevant* is now an array of terms suitable for use in generating boolean queries.

The search query synthesis begins with the construction of a CNF Boolean expression that selects all relevant documents and rejects irrelevant documents. This expression can be constructed by repeatedly constructing maxterms to cover all relevant documents whilst rejecting many of the irrelevant documents. Sanchez et al [7] have reported an algorithm for maxterm construction by endeavouring to reject the irrelevant documents that have not been rejected by the previously constructed maxterms. Thus, after each successive stage, the conjunction of maxterms rejects more irrelevant documents. We have made two important changes to the reported algorithm to make it suitable for synthesising search queries:

1. We restrict maxterms to contain only positive literals. We have not yet fully investigated the impact and implications of negated terms to the synthesis of search queries; instead it is taken into account through the application of Latent Semantic Analysis (see Section 4.2).
2. Each term (individual words in the example documents) is prioritised to reflect its potential as the next candidate term in the maxterm under construction. This ranking process is discussed further in Section 4.

The algorithm for constructing a CNF Boolean expression is shown below. The algorithm terminates successfully when the conjunction of maxterms rejects every irrelevant document. If all terms in a relevant document are also in an irrelevant document then it is not possible to define a maxterm that would simultaneously select all relevant documents and reject all irrelevant documents. When this happens the final Boolean expression will not reject all irrelevant documents - the algorithm can be forced to terminate by removing these problematic irrelevant documents from the *Irrelevant* set. The price for this action is a small reduction in the precision of the synthesised query.

It should be noted that the expressions produced by these algorithms may not be directly suitable for submission to a search engine. Many search engines place restrictions on the form or complexity of queries — for example Google restricts queries to ten terms. An algorithm for simplifying queries into a suitable form has been proposed in [8]. In this work we found that a suitable query could readily be generated by simply repeatedly re-applying the maxterm building algorithm — the random element within `Build_Maxterm` ensures variation between runs of the algorithm. In cases where no query below the limit of 10 terms was produced, we used the simple but very

### Build\_CNF\_BE

Input: Relevant, Irrelevant, Original Query.

Output: BoolExp

Description: BoolExp selects all documents in set *Relevant* and rejects virtually all documents in set *Irrelevant*.

```
TIR := Irrelevant;
i := 0;

while (TIR != {}) {
    i := i+1;
    Maxtermi := Build_Maxterm(Relevant, TIR);
    If (no Maxtermi possible) break;
    TIR := Temp σ Maxtermi;
}

BoolExp := (Initial Query) & Maxterm1 & Maxterm2 & ...;
```

### Build\_Maxterm

Input: Relevant, TIR

Output: maxterm

Description: maxterm selects every document in set *Relevant* and rejects some document in set TIR

```
TR := Relevant;
Candidate_terms_sorted := sort_candidates(set_of_terms_in(TR));
j := 0;
while (TR != {}) {
    j := j+1;
    Termj := Randomly_select_one_from_top_n(Candidate_terms_sorted);
    TR := TR - (TR σ Termj);
    Candidate_terms_sorted := sort_candidates(set_of_terms_in(TR));
}
maxterm := (Term1 | Term2 | ... | Termj);
```

Algorithm 1: Algorithm to construct Boolean expressions from examples.

effective approach of eliminating those terms which selected the least documents from the *Relevant* set.

## 4. PRIORITISING TERMS

The sorting of potential terms for addition to the current maxterm within the *Build\_Maxterm* algorithm is based on two factors related to that term. An initial ranking is produced based on the term's ability to discriminate between the relevant and irrelevant documents. This ranking is then modified based on characteristics of the term identified by the application of Latent Semantic Analysis to the documents in the Relevant and Irrelevant sets.

### 4.1. INITIAL PRIORITISATION

The potential of a candidate term is determined by the number of new relevant and irrelevant documents it selects. The potential for term  $t$  during construction of  $(i+1)^{\text{st}}$  maxterm is computed as follows (maxterm<sup>p</sup> is the partially constructed  $(i+1)^{\text{st}}$  maxterm):

```
TR = Relevant - (Relevant σ maxtermp);
TRt = TR σ t;
TIR = Irrelevant σ (maxterm1 & maxterm2 & ... & maxtermi);
TIRt = TIR σ t;
```

$$Potential(t) = \frac{(|TR_t|)(|TIR| - |TIR_t|)}{(|TR| - |TR_t| + 1)(|TIR_t| + 1)}.$$

### 4.2. LATENT SEMANTIC ANALYSIS

Landauer et al [9] have described LSA as

*a theory and method for extracting and representing the contextual-usage meaning of words by statistical computations applied to a large corpus of text. The underlying idea is that the aggregate of all the word contexts in which a given word does and does not appear provides a set of mutual constraints that largely determines the similarity of meaning of words and sets of words to each other.*

LSA uses no humanly constructed dictionaries or knowledge bases, yet has been shown to overlap with human scores on some language-based judgment tasks [10]. It shows particular promise in dealing with two of the problems encountered in information retrieval – *synonymy* (where different words have the same meaning) and *polysemy* (where the same word may have different meanings in different contexts).

The linear algebra factorization technique of singular value decomposition (SVD) is used to produce a reduced dimensionality representation of the word occurrence / context matrix. The term/context matrix  $X$

## Singular Value Decomposition

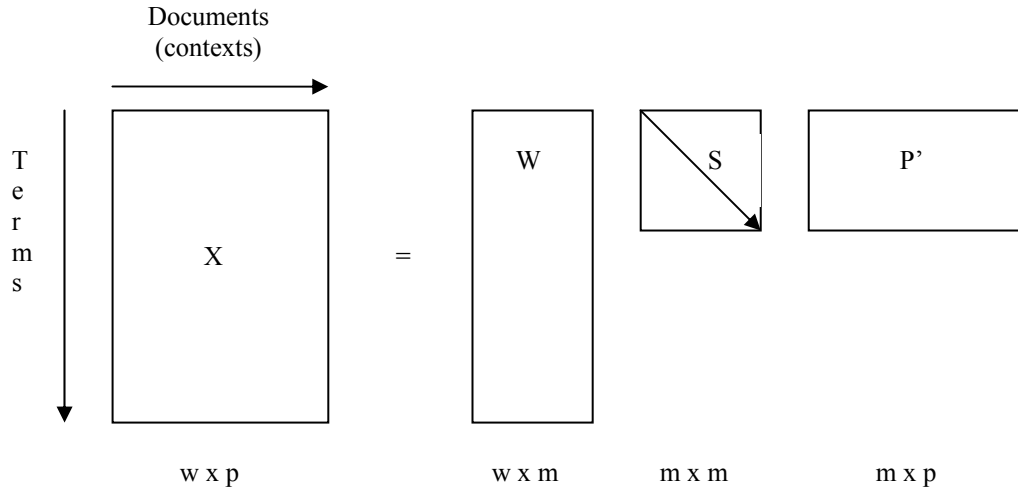


Figure 1: The singular value decomposition process as used in Latent Semantic Analysis

is first decomposed into the product  $WSP'$ . The diagonal matrix  $S$  (in which all entries are zero, except the diagonal from top left to bottom right) provides a weighting of the “importance” of each dimension in representing the original matrix  $X$ . By deleting all but the first  $n$  elements in  $S$  ( $n = 40 - 150$  has been shown to generally be most effective, with smaller values being more effective for smaller collections) and expanding  $WSP'$ , we obtain a dimensionally reduced approximation to  $X$ . Each row in the resulting matrix still represents a term, but each column represents an abstract “concept” and each entry gives a measure of how closely the word is associated with the “concept”.

The matrix obtained by the LSA process is often used to measure the similarity of documents to each other by calculating their closeness in this “concept space”, or their closeness to a query can be determined by modeling the query as a pseudo-document and finding the closest documents to it.

Another interpretation of the LSA matrix is in determining the “similarity relationship” between words [9] – if two words are found to appear with a similar weight in relation to each “concept” in the LSA matrix, then they have a “similarity relationship” – they may not have similar meanings, but they are often used together in discussing similar concepts. This relationship can be evaluated using the Spearman rank correlation coefficient ( $r$ ), calculated as:

$$r = \frac{n \sum xy - \sum x \sum y}{\sqrt{(n \sum x^2 - (\sum x)^2)(n \sum y^2 - (\sum y)^2)}}$$

where  $x$  and  $y$  are the weightings for two different terms over the range of “concepts” in the LSA matrix. It is this interpretation, together with the term frequency by inverse document frequency ( $tf*idf$ ) weight that we have used in this work to obtain a weighting to help identify the most important words relating to the concepts in the relevant and irrelevant document sets using the function `Calculate_LSA_Weighting`.

### Calculate\_LSA\_Weighting (Relevant)

Input: *Relevant*,  $tf*idf$  for all terms in relevant, LSA matrix.  
Output: LSA importance ranking for each term in *Relevant*.

```

For each term in Relevant {
    weight(term) = tf *idf(term);
    for each other term in Relevant {
        calculate r wrt this term;
    }
    for 10 terms with highest r {
        weight(related_term) =
            weight(related_term) +
            r * (tf*idf(term));
    }
    return weight(terms)
}

```

Algorithm 2: Algorithm to compute LSA weights.

The `Calculate_LSA_Weighting` function is then applied to set *Irrelevant* to obtain a similar rating of the importance of the words in the irrelevant documents. The two sets of weighting values are then normalized and combined using the formula  $\text{word weight} = (\text{word weight in relevant examples}) - (\text{word weight in irrelevant examples})$ . This value is then multiplied by the potential value as defined in 4.1 to give an overall weighting to the word in the `Build_Maxterm` algorithm.

## 5. EXAMPLE QUERIES AND RESULTS

To evaluate the success of this approach, some sample search-based tasks were identified. A query with a single term (a naïve query) was initially generated to retrieve links to documents which may be of interest. The first 150 links were examined and the retrieved documents were classified to form the sets *Relevant* and *Irrelevant*. These sets were then used to synthesise a new query, via the expression building process described in Section 3. The resulting queries were then re-submitted to the search engine, and the first 150 documents returned were again manually classified as

either relevant or irrelevant, to determine the algorithm's success in improving on the original naïve query. The query synthesis process was carried out twice: once with and once without the inclusion of the LSA weighting in the ranking of the potential terms. Google was used as the search engine for these experiments.

### 5.1. EXAMPLE 1: ELEPHANTS

The "information need" scenario for this trial was the task of locating information suitable for a high school project on elephants - e.g. habits, diet, distribution, life-cycle, species, and conservation. The initial, naïve query consisted of the single word "elephant". This query returned 43 relevant documents and 96 irrelevant documents (11 documents from the first 150 links could not be downloaded).

#### 5.1.1. WITHOUT LSA WEIGHTING

The query generated without LSA weighting was:

```
elephant AND (katy OR snorkeling
OR forests OR lived OR calves OR
poachers OR maximus OR threaten
OR chemistry OR electric OR tel
OR kudu)
```

This query contained 13 terms, and failed to eliminate just one of the irrelevant documents. To reduce the complexity of the query to the 10 term limit imposed by Google, the terms which selected the least number of relevant documents were dropped yielding the following query:

```
elephant AND (forests OR lived OR
calves OR poachers OR maximus OR
threaten OR electric OR tel OR
kudu)
```

When this query was submitted to Google the first 150 links returned produced 114 relevant documents and 36 irrelevant documents - a significant improvement on the original naïve query.

#### 5.1.2. WITH LSA WEIGHTING

With LSA weighting included in the ranking of maxterms, the query produced was:

```
elephant AND (ornithology OR lung
OR climate OR habitats OR
grasslands OR quantities OR
africana OR threaten OR chemistry
OR insects OR electric OR kudu)
```

Again this query selected just one of the irrelevant set, and consisted of 13 terms. As before the terms selecting the least number of relevant documents were eliminated to yield:

```
elephant AND (climate OR habitats
OR grasslands OR quantities OR
africana OR threaten OR insects
OR electric OR kudu)
```

When submitted to Google, this query produced 122 relevant documents, and just 28 irrelevant documents.

The synthesised query reported in 5.1.1 produced a 62% reduction in irrelevant retrievals compared to the naïve query. The LSA weighting has further removed another 22% of irrelevant results from the retrieved links.

### 5.2. EXAMPLE 2: MUSHROOMS

The scenario for this trial was the gathering of information on the growth, life-cycle, and structure of mushrooms. In particular it was assumed the user was not interested in recipes, magic mushrooms, or identifying wild edible mushrooms. The initial, naïve query consisting of the single word "mushrooms" produced 18 relevant documents and 123 irrelevant documents (9 documents could not be retrieved).

#### 5.2.1. WITHOUT LSA WEIGHTING

The query generated without LSA weighting was:

```
mushrooms AND (cylindrical OR
ancestors OR hyphae OR cellulose
OR issued OR hydrogen OR
developing OR putting)
```

This query was successful in not selecting any documents from the Irrelevant set. It contained just 9 terms, and therefore was suitable for submission to Google without modification. It returned 86 relevant documents and 64 irrelevant documents.

#### 5.2.2. WITH LSA WEIGHTING

With LSA weighting included in the ranking of maxterms, the query produced was:

```
mushrooms AND (ascospores OR
hyphae OR itis OR peroxide OR
discharge OR developing OR pulled
OR jean)
```

This query did not select any documents from the irrelevant set, and consisted of just 9 terms. When submitted to Google, this query produced 91 relevant documents, and just 59 irrelevant documents. Again there is a 48% reduction in irrelevant retrieved links using the synthesised query. LSA weighting further benefits the results by 8%.

## 6. DISCUSSION

In both of the example tasks the query synthesised from the documents returned by the original single-word query was successful in improving significantly on the precision and recall of that initial query. The number of relevant documents produced by the original query did not appear to be an impediment to the success of the query-building algorithm, as the second trial was successful in synthesising useful search terms from just 18 relevant documents identified by the initial query.

The inclusion of the LSA weighting factor in the ranking of terms appeared to be beneficial — in both trials the query generated using the LSA weight produced a higher percentage of relevant documents

than the query produced without this weight. This effect was relatively small but is on an already strong base as the non-LSA query had already significantly improved on the original query. These results would need to be demonstrated over a larger range of tests before it could be regarded as significant and more universally applicable.

More significantly, the LSA weight benefits the synthesis process by constructing queries which are more intuitive or sensible if examined by the humans, although this is inherently a subjective issue. The non-LSA queries included a number of terms which would appear to be unrelated to the main topic of the search, or so general as to be of little discriminatory function (e.g. "lived" and "tel" in the elephant trial; "cylindrical", "issued" and "putting" in the mushroom trial). The LSA queries were not totally successful in eliminating such terms (e.g. "quantities" in the elephant trial; "jean" in the mushroom trial), but overall appear more closely related to the problem domain than the non-LSA queries. The choice of these non-intuitive terms in the synthesised query is an indication of the difficulties that human users face when attempting a search in an area unfamiliar to them. Without a computer-aided process they are left to a difficult struggle.

## 7. CONCLUSION

The paper has described a method for constructing search queries from a collection of relevant and irrelevant text documents. An initial single-word query produced by the user is used to gather sample documents which are manually classified as relevant or irrelevant to the task at hand. These documents are pre-processed to produce a list of terms for use in constructing improved queries. It should be noted that the pre-processing stage is not strictly necessary. For example, elimination of stop words is not essential to the method. A term that appears frequently among the irrelevant documents – including the stop terms – would be relegated to lower levels of selection. Likewise, we have not found any real evidence to support stemming of the terms. However, a reduction in the size and number of terms to handle during processing makes it faster and more efficient. As a consequence we have chosen to remove stop words and stem the words.

Following pre-processing, a query is constructed using the algorithm given in Section 3. This process is essentially greedy — in our experiments, we found the greedy approach to be adequate and there is little need to seek optimised and perfect Boolean expressions. The results in Section 5 indicate that the algorithm successfully produces a much higher proportion of relevant documents than was obtained by the original query, and does so without any need for domain knowledge other than the classification of the documents returned by that original query.

We have experimented with the incorporation of weights based on LSA in the ranking of terms for

possible inclusion in the expanded query. The results reported here indicate that this may prove beneficial, both in terms of improving the precision of the query, and increasing the extent to which the query matches the user's intuition about the search task, which may increase user acceptance of this process.

In this paper we have used a relatively simplistic approach to eliminate terms from any queries which are too complex to be directly submitted to a search engine. An area for future research is identifying whether the system's performance can be further improved by using a more sophisticated algorithm to accomplish this task — for example by re-working the query into conjunctive normal form and analysing the discriminatory power of the resulting minterms as suggested in [8].

We are also exploring the possibility of integrating the query construction procedure with a browser.

## REFERENCES

- [1] Oyama, S., Kokubo, T. and Ishida, T. (2004): Domain-specific web search with keyword spices, *IEEE Transaction on Knowledge and Data Engineering*, 16(1): 17-27.
- [2] Kherfi, M.L., Ziou, D., and Bernardi, A. (2004). Learning from negative example in relevance feedback for content-based image retrieval, *Proc. of the IEEE Internl. Conf. on Pattern Recognition*, (Quebec, Canada).
- [3] Kherfi, M.L., Ziou, D., and Bernardi, A. (2004). Image retrieval from the World Wide Web: issues, techniques and systems, *ACM Computing Surveys*, 36(1): 35-67
- [4] Aho, A.V. and Ullman, J.D. (1992): *Foundations of computer science*, New York: Computer Science Press.
- [5] Witten, I.H. and Frank, E. (2000): *Data mining: practical machine learning tools and techniques with Java implementations*. San Francisco: Morgan Kaufmann Publishers.
- [6] Sebastiani, F. (2002): Machine learning in automated text categorization, *ACM Computing Surveys*, 34(1): 1-47.
- [7] Sanchez, S. N., Triantaphyllou, E., Chen, J. and Liao, T.W. (2002), "An Incremental Learning Algorithm for Constructing Boolean Functions from Positive and Negative Examples", *Computers and Operations Research* 29(12) 2002. 1677 – 1700.
- [8] Patro, S. and Malhotra, V.M. (2004). Generating Web Search Query from Examples, (under review).
- [9] Landauer, T.K., Foltz, P.W. and Laham, D. (1998): Introduction to latent semantic analysis, *Discourse Processes*, 25: 259-284.
- [10] Berry, M. W., S. T. Dumais, et al. (1995). Using linear algebra for intelligent information retrieval, *Siam Review* 37(4): 573-595.

