ABSTRACT

Boolean search is still the method of choice for many kinds of professional search, such as constructing systematic reviews in legal and medical fields. It is effective for fast, high-recall document classification. Its drawback is the difficulty in crafting a Boolean query that captures semantically relevant documents. Ambiguous search terms lead to the inclusion of non-relevant documents. We propose a new exploratory document classification system that guides the user in creating a Boolean search query with both high recall and high precision. To increase recall, the system suggests semantically similar search terms. However, unlike other systems that use interactive query expansion, ours also includes a mechanism for removing non-relevant documents that inevitably get matched by an expanded query. The system explicitly identifies semantically ambiguous query terms that reduce the precision of the search. Removing non-relevant documents increases precision without reducing recall.

1. INTRODUCTION

Boolean search uses exact-matching logic on keywords to find relevant documents from a large dataset. It has been widely used for document retrieval since at least the 1960’s despite the frequent criticism of its drawbacks. For some information retrieval tasks, its benefits outweigh its shortcomings. Examples include the construction of systematic reviews of medical research, legal precedent, and patent prior-art. Kim et al. [4] and Pohl et al. [6] describe the many reasons practitioners continue to use Boolean search, including an emphasis on recall and reproducibility.

Frants et al. [3] consolidated the criticisms in academic literature against Boolean search. The primary drawback is that queries are difficult to formulate for most users. Another drawback is that the size of the output is difficult to control. We present an interactive system for document classification that simplifies the process of semantically refining a Boolean search query. Our approach is based on the annotation of individual words. The method is therefore fast to use in comparison to approaches requiring the analysis of complete documents. No advanced knowledge of Boolean search is expected of the user. Instead, our system helps the user explore the document database by refining the set of words presented for labeling based on prior word labels. The user can choose to label any subset of these words and the system updates its word suggestions based on this feedback.

We use interactive query expansion to increase recall by soliciting relevance labels of new words from the user. However, query expansion also degrades precision when the user chooses query terms that are ambiguous and can be used in non-relevant contexts. Our system includes a novel way of identifying such unwanted term usage. It suggests a separate list of words that a user can select to remove non-relevant documents, increasing precision without reducing recall.

2. RELATED WORK

We briefly describe some recent work on query expansion for Boolean search. None of these systems allow the user to detect and correct query drift.

Kim et al. [4] present a way to help professional searchers by suggesting Boolean queries they can add to the overall search. They use pseudo-relevant documents to automatically generate such queries. They then evaluate features of the queries in order to rank them for effectiveness.

Bashir and Rauber [1] compare methods used for patent prior-art searches. They propose using Language Modeling with pseudo-relevance feedback to automatically perform query expansion.

Xue and Croft [8] and Larkey [5] have systems that can convert a full user-selected patent into a Boolean query. Larkey’s system also incorporates user-guided query expansion. It can suggest phrases related to the initial query, or phrases that contain the initial query.

Sciano and Inkpen [7] use Wikipedia to automatically identify the correct sense of a list of advertising keywords, in order to prevent wasted ad impressions. Their system automatically generates a set of negative keywords.

3. SYSTEM DESIGN

The system interacts with the user in a series of iterations, where at every iteration the user labels a subset of words suggested by the system followed by the system updating the set of word suggestions. The user interface is shown in Figure 1. Note that the users of our system do not directly work with Boolean operators. Instead, they tag suggested words with labels in order to expand and refine the underlying implicit query. After the user completes labeling one
or more new query words (indicated by pressing “refresh”),
the system incorporates the new information and updates
all suggestions. The specific word labels are described in
the following subsections. At the end of the iterative pro-
cess, the system formulates a boolean query which is used
to retrieve documents.

3.1 Boolean Query Model

The simplicity our system provides is made possible by
restricting the Boolean query, \( q \), to the following form:

\[
q = (a_1 \lor a_2 \lor \ldots \lor a_m) \land \lnot (x_1 \lor x_2 \lor \ldots \lor x_n)
\]

In this query, the terms \( a_i \) are denoted anchor-words. Ide-
ally, these words should be unambiguous, since documents
that contain even one of them are considered relevant. The
terms \( x_i \) are called ax-words; these are negative keywords
which disqualify documents and remove them from the search
results. Ax-words are used to compensate for imperfect or
ambiguous anchor-words. For example, the anchor-word
“condition” might be used when searching for documents
restricting the Boolean query,

\[
x_i \in A_i, \quad x_j \in X_j
\]

This redundancy also confers certainty. Documents that
contain multiple anchor-words and support-words are more
likely to be relevant than those that contain only one. We
suggest as ax-words those that can be accessed by the system. Let \( D_{q_i} \) represent the
subset of documents in \( D \) that match query \( q_i \).

Query expansion suggestions are those that could be la-
beled as anchor- or support-words. The relevance of a word
\( w \) is calculated using a scoring function, \( s_A(w) \), defined as

\[
s_A(w) = \frac{[\text{contains}(D_{q_i}, w)]}{|D_{q_i}|(|\text{contains}(D, w) + \lambda)}
\]

where \( \lambda \) is included to penalize rare words. The scoring function is based on Point-
wise Mutual Information (PMI) [2]. The PMI between two
terms is a measure of their co-occurrence that uses their
independent occurrence as a baseline. Intuitively, a word \( w \) that has high PMI with documents from the current query
\( q_i \) occurs often in these documents and is likely to be a good
candidate for an anchor- or support-word.

3.2 Query Expansion

Coming up with a minimal set of anchor-words and ax-
words is a difficult task because the user may not know which
topics are in the dataset and which words describe each
topic. Our system helps explore the data and choose appro-
riate query words by offering suggestions in an interactive
manner. At every iteration, the user can label some subset
of the suggested words as anchor-, ax-, or support-words. We
define support-words as terms that are too broadly used to
be effective anchor-words, but provide information for dis-
ambiguation. For instance, consider an ambiguous anchor-
word, “pitch.” It is most likely associated with baseball, but
also be used in settings related to music, computer mon-
itors, etc. A support-word such as “game” or “team” would
clarify the usage of the word “pitch” in a document. Note
that support-words do not form part of the search query, and
are only used by the system to generate more discriminative
ax-word suggestions (described below).

**Query Expansion Suggestions:** Let \( A_i, X_i, \) and \( S_i \)

\[
A_i, X_i, \quad S_i
\]

de note the set of labeled anchor, ax, and support words
respectively at iteration \( i \). The query at iteration \( i \), \( q_i \), is
then 

\[
q_i = (a_{i1} \lor a_{i2} \lor \ldots \lor a_{im}) \land \lnot (x_{i1} \lor x_{i2} \lor \ldots \lor x_{in}),
\]

\( a_{ij} \in A_i, \quad x_{ij} \in X_j \). Let \( D \) denote the set of all documents
that can be accessed by the system. Let \( D_{q_i} \) represent the
subset of documents in \( D \) that match query \( q_i \).

Query expansion suggestions are those that could be la-
beled as anchor- or support-words. The relevance of a word
\( w \) is calculated using a scoring function, \( s_A(w) \), defined as

\[
s_A(w) = \frac{|\text{contains}(D_{q_i}, w)|}{|D_{q_i}|(|\text{contains}(D, w) + \lambda)}
\]

where contains(Z, w) is the subset of documents in Z that
contain word w. The smoothing factor, \( \lambda \), is included to
penalize rare words. The scoring function is based on Point-
wise Mutual Information (PMI) [2]. The PMI between two
entities is a measure of their co-occurrence that uses their
independent occurrence as a baseline. Intuitively, a word \( w \) that has high PMI with documents from the current query
\( q_i \) occurs often in these documents and is likely to be a good
candidate for an anchor- or support-word.

**Ax-word Suggestions:** Ax-word suggestions are intended to
bring attention to possible alternate non-relevant ways an
anchor-word is used in the dataset. To create these sugges-
tions, we take advantage of the fact that anchor-words can
be correlated, adding redundant information to the query.
This redundancy also confers certainty. Documents that
contain multiple anchor-words and support-words are more
likely to be relevant than those that contain only one. We
separate relevant documents into those that are solidly-relevant,
and those that are barely-relevant, and suggest as ax-words
the words that are most associated with the barely-relevant
documents.

A word \( w \) is ranked on its likelihood of being an ax-word

![Figure 1: Interactive Boolean Search System](image-url)
with respect to each anchor-word \( a \in A_i \). The scoring function, \( s_X(w, a) \), is defined as
\[
s_X(w, a) = \frac{|\text{contains}(B_i(a), w)|}{|B_i(a)|(|\text{contains}(D_{n+1}, w)| + \lambda)}
\]
Here \( B_i(a) \) is the set of barely-relevant documents in \( D_{n+1} \) that contains the anchor-word \( a \) and no other anchor-words or support-words. While not necessary, we used the same value for \( \lambda \) in both \( s_X \) and \( s_s \).

Function \( s_X \) is defined similarly to \( s_s \). However, instead of finding words that help separate relevant documents from non-relevant documents, \( s_X \) finds words that separate barely-relevant documents from solidly-relevant documents. Such suggestions are useful in discovering alternate uses for anchor-word \( a \), and make the most intuitive sense when displayed alongside \( a \). Therefore, our system displays the top five ax-word suggestions (called “Ambiguous 1”, etc.) to the right of each associated anchor-word (Figure 1).

### 3.3 Effect Indicators

We also display effect indicators as integers next to words in most of the text areas. These indicate how much the relevant document count will change if the word is labeled. The meaning of the number depends on the context. The suggestion column shows how many new documents will be added if the word is tagged as an anchor-word. The ax-word column shows how many new documents will be added if the word is removed from the set of ax-words. The anchor-word column shows how many documents will be dropped if that anchor-word is removed. The ax-word suggestion matrix shows how many documents will be dropped if that word is tagged as an ax-word.

These indicators serve three purposes:

1. They help the user control how many documents are returned in the search.
2. A large number in the suggestion column can indicate that word would make an ambiguous anchor-word, with unconsidered, non-relevant uses.
3. Small numbers indicate diminishing returns. It may not be worth the user’s time to continue to refine the search query with such words.

### 4. SYSTEM EVALUATION

Our experiments evaluate the usefulness of system suggestions using the class labels in the 20 Newsgroups dataset.\(^1\) We remove the message header lines (except the subject line) as well as all common stop-words. We convert the text to lowercase before tokenizing.

We first consider whether ax-words can be found that would meaningfully improve retrieval precision. For each of the 20 classes, we calculate the precision of each single-anchor-word classifier. We then find, through an exhaustive search, the best possible ax-word to pair with each anchor-word and calculate the precision improvement. Anchor-ax pairs that matched fewer than 100 documents are discarded as uninteresting. Figure 2 shows the results. We see that the majority of words have low precision, indicating that they are not useful for identifying the class. Many of these low-precision words cannot be meaningfully improved by adding an ax-word. The highest-precision words also cannot be meaningfully improved, because they are already unambiguously associated with the class. However, many words fall in the 5-15% improvement band which indicates an upper bound on the F1-score improvement expected by the addition of ax-words to the query.

We assume the user can recognize good ax-words, but not find them without seeing them suggested. We next evaluate the quality of the system’s ax-word suggestions. To test this objectively, we first define a set of hypothetical queries. For each class, we begin with a query made up of the top 20 anchor words, chosen according to their individual precision, and all matching at least 100 documents. With these sets of anchor-words, we generate the usual five ax-word suggestions per anchor-word. We use a smoothing parameter of 100 for all experiments. For each class, we find the fraction of ax-word suggestions that would, if chosen, improve the classifier F1-score. We also find the fraction of ax-word suggestions that would increase the F1-score by a meaningful amount, set to 0.5%. The results in Table 1 show an average of 66% of the suggestions would help if chosen as ax-words, and that an average of 22% of them would help meaningfully. This illustrates how, in this experiment, a high fraction of the ax-word suggestions are useful.

As part of the system evaluation, we also include the details of one illustrative interactive search session in Figure 3. The x-axis shows each word in the order it was chosen, and the y-axis shows the F1-score, precision, and recall of the classifier at each step. Normally, when interacting with the system, the user can alternate between choosing anchor-words, support-words, and ax-words. However, for easier visualization of this particular example, we performed the search in only two phases. In the first phase, only anchor-words were chosen. In the second phase, only ax-words were chosen. In this illustrative example, the first phase steadily increases the recall, sometimes at the expense of the precision. The F1-score peaks in phase one and starts to decline. Phase two, adding ax-words, is seen to steadily bring the precision back up with only a slight reduction in recall. In this example, we achieved our goal of having a high-recall classifier that also maintains high precision.

---

\(^1\)http://qwone.com/~jason/20Newsgroups/
5. CONCLUSION

We present a new interactive Boolean search system, suited for high-recall applications. One significant contribution is the mechanism for identifying and removing non-relevant documents retrieved due to ambiguous search terms. Using the 20 Newsgroups dataset, we showed that filtering documents using ax-words has potential to increase precision. The system’s word suggestions are effective at finding ax-words that can meaningfully increase the F1-score of the search. We also gave details of a simplified example search session, showing how adding anchor-words increases recall at the cost of precision, but how adding ax-words can subsequently increase the precision. Future work would include a user study to evaluate the system under real-world use.

6. ACKNOWLEDGMENT

This work is supported by Chevron USA, Inc. under the joint project Center for Interactive Smart Oilfield Technologies (CiSoft), at the University of Southern California.

7. REFERENCES


Figure 3: Sample session, searching for rec.sport.baseball in the 20 Newsgroups dataset.

Table 1: The middle column indicates the fraction of ax-word suggestions presented by the system that, if chosen, would improve the classifier F1-score, if even just a little. The last column indicates the fraction that would boost the classifier F1-score by at least 0.5%, a more meaningful amount.