# Latent Semantic Indexing with Selective Query Expansion

**Andy Garron**

**04/23/2012**

**Submitted to the faculty of Ursinus College in fulfillment of the requirements for Distinguished Honors in Computer Science**

**Ursinus College**

The Undersigned Faculty Committee Approves the

Distinguished Honors Thesis

"Latent Semantic Indexing with Selective Query Expansion"

submitted by Andy Garron

---

April Kontostathis, Advisor

Department of Mathematics and Computer Science

---

April Kontostathis, Committee Member

Department of Mathematics and Computer Science

---

Akshaye Dhawan, Committee Member

Department of Mathematics and Computer Science

---

Thomas Carroll, External Reviewer

Department of Physics

---

John P. Dougherty, External Reviewer

Department of Computer Science, Haverford College

---

Mohammed Yahdi, Chair

Department of Mathematics and Computer Science

---

Approval Date

**Abstract**

This thesis outlines research and experiments in Information Retrieval, specifically with the search method Latent Semantic Indexing (LSI). The design and implementation of a framework for performing LSI queries on large text databases with a focus on machine learning is documented. Also described and analyzed is our participation in TREC in 2010 and 2011. Finally, we describe our next LSI-based task, Cyberbullying detection, which involves applying our system to Internet-chat-based datasets to discover instances of online harassment.

The results of our research show that EDLSI is an effective technique for E-Discovery. Preliminary observations also show promise for EDLSI's effectiveness in Cyberbullying detection. We have shown that selective query expansion can be a useful mechanism for improving retrieval results when a specific initial query is provided. We believe that queries that are stated in general terms, however, may suffer from "expansion in the wrong direction" when certain iterative approaches to incorporating relevance feedback information are added into the search process.

# Chapter 1

# Introduction

Legal professionals are often presented with massive document sets that are potentially relevant to a case at hand. Any document can be used as important evidence at trial, but these datasets are generally too large to analyze manually. E-discovery refers to the use of electronic search engines to assist with, or automate, the legal discovery process. Finding an effective and efficient search algorithm for E-discovery is an interesting open problem.

The Text REtrieval Conference (TREC) is an annual conference started in 1992 to support research in the information retrieval community. It focuses on testing retrieval methods on large-scale datasets which are of interest to researchers from industry, academia, and the government. The conference is split into multiple tracks (Medical, microblog, etc.) in order to facilitate more focused research.

To provide a framework to study and improve on E-discovery techniques, the TREC Legal Track was created. This track attempts to emulate E-Discovery,

including the use of real legal datasets and attorneys. Each system is designed to retrieve documents that are relevant to a specific request for information (in 2010, there were eight topics; in 2011, there were three). This simulation includes an opportunity for machine learning based on relevance feedback. Teams implement machine learning systems to improve search results over multiple iterations by consulting with a Topic Authority (TA) about the relevancy of sets of documents compiled by the teams during the previous iteration. The TA is a legal expert who can answer questions about a particular topic.

This paper describes the research leading up to and the results produced by the Ursinus team during the 2010 and 2011 competitions. The system we implemented for both 2010 and 2011 is based on Latent Semantic Indexing (LSI), a search method that attempts to draw out the meaning of terms[5]. In particular, we implemented Essential Dimensions of LSI (EDLSI), which combines standard Vector Space retrieval with LSI in a "best of both worlds" approach[7][3]. In 2010, teams were granted a training set of known-relevant and known-irrelevant documents for each of eight queries with which to train their systems. In 2011, teams were allowed multiple submissions for each query ("runs"); after each run they received relevance judgments for a number of documents. This procedure lends itself intuitively to selective query expansion. In selective query expansion, we modify the query using information from documents that are known to be relevant in order to train the system to produce better retrieval results. We implemented selective query expansion as a machine learning feature in our system.

Results from the most recent TREC conference suggest that EDLSI with selective query expansion is competitive with other methods in large-scale E-

Discovery applications. Our system was consistently near the median when compared to all teams participating on a given topic with respect to estimated *F1*. Our learning method experienced diminishing returns, however, and our system was clearly better for one topic than for the other two. We believe this was due to query characteristics that move queries away from relevant documents in the query expansion cycle.

Also described is research in Cyberbullying detection using EDLSI and machine learning. Cyberbullying is defined as "willful and repeated harm inflicted through the medium of electronic text" and may entail flaming, trolling, cyberstalking, denigration, etc. [9] [11]. Using raw data from Formsping.me (a social website built around asking anonymous questions of other users) we experiment using our system to detect Cyberbullying throughout a large dataset. This process involves the use of Amazon's Mechanical Turk to build a training set we can use to apply machine learning, along with EDLSI in an attempt to discover instances of Cyberbullying. Results of initial experiments in the Cyberbullying task show promise, and close analysis highlights the need for adjustments to improve our results. In particular, documents containing high volumes of foul language often are returned as false positives.

# Chapter 2

# Background/Related Work

In this section we describe the basic algorithms used in our system.

## 2.1 Vector-Space Retrieval

In Standard Vector Space Retrieval [1], documents are represented as vectors of dimension $m$ x 1, where $m$ is the count of terms in the dataset and position $[i, 1]$ of each vector represents how many times term $i$ appears in the document. Queries are represented in the same way (vectors of termcounts), and it is assumed that documents with vectors closer to the query vector are more relevant to the query. One limitation of standard vector space retrieval is that if none of the query terms appear in a document, that document will not be returned as relevant. This can be counterintuitive when you are searching (for example) for cars, and there exist documents in the dataset that are about automobiles, but which never explicitly contain the term car. This problem is called synonymy. Another common problem is polysemy (words having multiple meanings, i.e.

"plant" referring to either a green thing with leaves or some sort of factory). If the majority of the documents in a dataset mentioning plant also mention roots, leaves, stems, etc., searching for "plant" will return the documents about the green life forms as more relevant than those primarily about production facilities. In the next section, we describe Latent Semantic Indexing. LSI has been shown to use second-order and higher-order word co-occurrence to overcome the synonymy and polysemy problems in some corpora [8].

## 2.2 Latent Semantic Indexing

LSI is an extension of the Vector Space search model. It is designed to extract the "meaning" of words by using their co-occurrences with other words that appear in the documents of a corpus [5]. Latent Semantic Indexing, like Vector Space retrieval, uses the term-document matrix of a dataset, i.e. the matrix that contains a dataset's terms in its rows and documents in its columns, with position $[i, j]$ representing how many times term $i$ appears in document $j$.

LSI is named for its ability to draw out the "latent semantics" in a dataset. That is to say, information about what a term might mean with respect to other terms that appear often in the same documents. This is accomplished by computing the Singular Value Decomposition (SVD) of the term-document matrix. The SVD is a matrix factorization method that splits a matrix, $A$, into three separate matrices $U$, $S$, and $V$ where:

$$A = U \times S \times V^T$$

The eigenvalues of $A^T A$ are computed, and the square roots are taken to

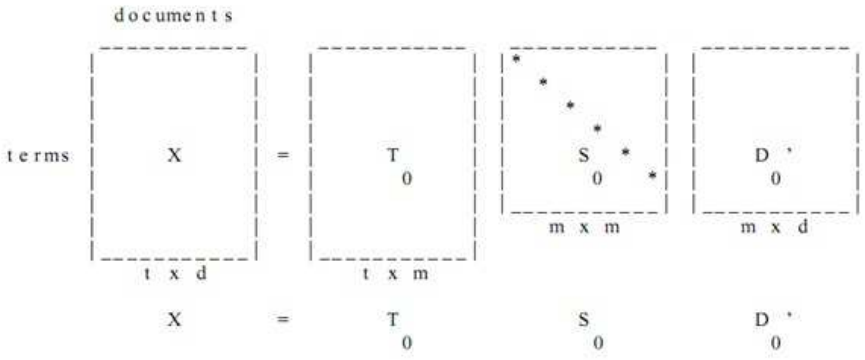


Figure 2.1: Full SVD of termdoc matrix $X$ ($T = U$, $D = V$) [5]

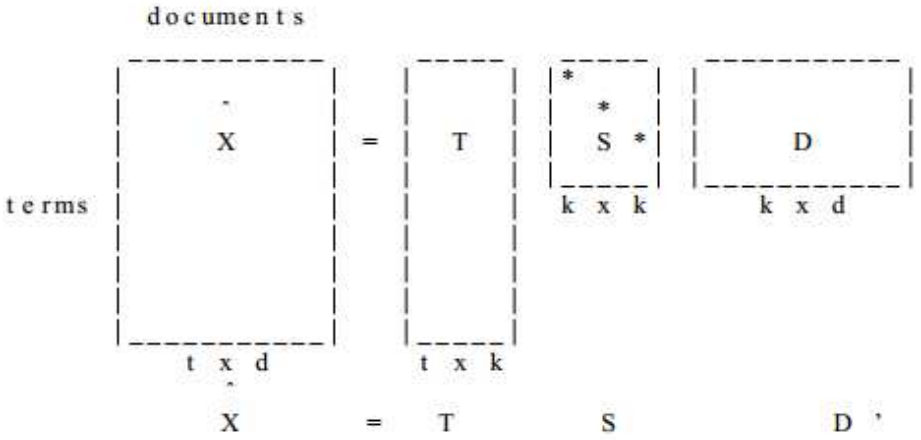


Figure 2.2: SVD reduced to $k$ dimensions [5]

obtain the singular values of $A$. These values are then arranged in descending order on the diagonal of $S$. $U$ and $V$ are then computed accordingly. In LSI, $U$ is known as the term matrix, and $V$ is known as the document matrix.

Factoring the matrix into this format is not sufficient, because no information has been gained. Multiplying the matrices back together produces the original term-document matrix. The power of LSI comes from truncating the $U$, $S$, and $V$ matrices to $k$ dimensions (See Figs. 2.1 and 2.2).

Multiplying $U_k S_k V_k^T$ produces the best rank-$k$ approximation of the original term-document matrix [7]. This recalculated matrix $A_k$ is very dense whereas $A$ is very sparse, which poses problems in actual computation, but conceptually is where the power of LSI comes through: $A_k$ contains all the information regarding the higher-order relationships between terms in the dataset [8]. In practice, fast algorithms are used to compute the partial SVD to $k$ dimensions (rather than decomposing the entire matrix and truncating) [6].

## 2.3 Running Queries

In both LSI and vector space retrieval, documents in the dataset are represented as vectors, with each vector position representing the weight for each term in the corpus. Queries are represented in the same way. To run a query, each document vector is compared to the query vector. The most common metric for comparison is cosine similarity, the cosine of the angle between a document vector and the query vector. The result of this calculation is referred to as the document weight or the relevance to the query. The documents with cosine similarities near one are very close to the query vector (since $\cos(0) = 1$, meaning the cosine similarity approaches one as the document and the query get closer to one another) and are assumed to be more relevant to the query. In LSI, we have to first transform the query vector into the same space as the document vectors before we can compute the cosine. Each document vector is taken from a column of $V^T$, and the equation for transforming the query vector is:

$$q_k = q^T U_k S_k^{-1}$$

Each document vector then has its cosine similarity to the query vector calculated, and that result is recorded as the final relevance score for the document/query pair [3].

## 2.4 Essential Dimensions of LSI (EDLSI)

One of the disadvantages of LSI is that $k$ (how far we calculate the dimensions of the SVD matrices) must be large enough to contain enough information to accurately represent the dataset we are using. It has been shown that if only a few dimensions of the SVD are computed, they contain the most important data that the LSI extracts from the dataset, but not enough to accurately run searches [7]. However, these "Essential Dimensions of LSI" can still be used, in conjunction with the original term-document matrix. We can use the LSI scores on documents to try and draw out the latent semantics in a dataset, while also using the raw power of vector-space retrieval, by simply adding together the results from LSI and the results from vector-space. Weights are applied to balance the contributions from each source. The final relevance score, $w_{EDLSI}$, for a document, $d$, is computed as:

$$w_{EDLSI} = (1 - x)w_{LSI} + (x)w_{vector}$$

where $w_{LSI}$ is the relevance score from traditional LSI, $w_{vector}$ is the relevance score from traditional vector space retrieval, and $x$ is a weighting factor ($0 \leq x \leq 1$).

# Chapter 3

# Summer Fellows 2010 Work

In this section, the implementation of an EDLSI system is described. Work in Summer 2010 focused on implementing the system and participation in TREC 2010's Legal Track.

## 3.1 TREC 2010 Legal Track Details

The TREC Legal Track in 2010 had a learning task, where teams implemented systems that were able to learn from given information and optimize themselves appropriately for the dataset. For each of eight queries, teams were given a training set comprised of documents that were known to be relevant and a set of documents that were known to be irrelevant. It was among our challenges to come up with an efficient and effective way of retrieving relevant documents using machine learning techniques.

## 3.2 TREC 2010 Methodology

This subsection describes the steps we took to process and search the dataset used during TREC 2010 and to analyze the results of those queries in order to arrive at a final submission.

### 3.2.1 Preprocessing the Enron Email Dataset

In 2009, 2010, and 2011, the TREC Legal Track used the Enron email dataset. The Enron email dataset contains 685,592 documents in many formats (including .txt, .pst, .ppt, etc.) that take up 3.71 gigabytes of disk space. We removed all documents that fit any of the following criteria:

- Were not in .txt format
- Contained no text other than the information universal to all documents (disclaimers, etc.)
- Contained only junk terms, including:
  - terms longer than 17 characters
  - terms with characters not from the English alphabet
  - terms with 3 or more of the same character in a row
  - terms that appeared in greater than 100,000 documents

After culling documents and terms, we were left with 456,968 documents and 302,119 unique terms. Each document was then added to a Lemur index (an efficient way of storing large amounts of data for fast retrieval) in order to create the term-document matrix for the dataset. The Lemur Toolkit is designed to take a set of documents and add them quickly to an index [10].

### 3.2.2 Calculating the Term Document Matrix

The main challenge in creating the term-document matrix is working with limited storage space. When the Enron email dataset is trimmed of useless terms and documents, there are 456,968 documents and 302,119 terms. In order to store a dense matrix of that size in memory, we would need just over one terabyte.

This exceeds the memory available on our research systems. However, because most terms appear in few documents, the matrix will be very sparse. In sparse matrix format, the term-document size for the TREC dataset is 30,252,865 nonzeroes, requiring only 0.2254 gigabytes of storage space, a much more manageable size. Using the Lemur API for C++ to interface with the index of the dataset, we create the sparse term-document matrix and apply the term frequency - inverse document frequency (tf-idf) weighting scheme.

### 3.2.3 Tf-idf Weighting

We used the tf-idf weighting scheme. It attempts to weight terms more heavily if they appear often in a given document, but is offset by the total count of the term in the corpus, so that words that are common in general do not receive extra weighting. The score of a term in a document (what goes into position $[i, j]$ of the term-document matrix, where $i$ is the term number and $j$ is the document number) is computed as:

$$score = tf \text{ x } idf$$

$$tf = \frac{D_{[i,j]}}{TC_j}$$

$$idf = log(\frac{DC}{T_i})$$

where $D_{[i,j]}$ is the number of times term $i$ appears in document $j$, $TC_j$is the total count of terms in document $j$, $DC$ is the total count of documents, and $T_i$ is number of documents containing term $i$.

Term frequency ($tf$) measures the local importance of a term within a particular document. Inverse document frequency ($idf$) measures the discriminatory power of the term within the entire corpus.

Tf-idf weighting was used for our term-document matrices in both 2010 ad 2011.

### 3.2.4 Computing the SVD

In 2010, we used the SVDLIB toolset [12] to compute the partial SVD of the termdoc matrix to $k$ dimensions. The ideal value for $k$ varies among datasets, and there is no way to determine the optimal value except by experimentation. It is safe to assume, however, that 75 singular values for a matrix the size of the Enron dataset termdoc is too small; in 2010, we were unable to calculate more than 75 dimensions due to memory restrictions on our research machines.

### 3.2.5 TREC 2010 Learning and Submissions

Three runs were allowed for each team for each topic. For each topic, we were given a training set of documents known to be relevant or irrelevant. Due to time constraints, we implemented a very basic learning system. The only variables in our system at that time were $k$, the dimensionality of the truncated term-document SVD and $x$, the weighting factor for Vector Space vs LSI. A good value for $x$ across a variety of datasets has been shown to be $x = .2$, so we used that value [7]. We ran EDLSI queries with varying values of $k$, from 5 to the

75 we were able to calculate. The queries themselves were created from terms relevant to each of the given topics. To decide which EDLSI runs to submit, a scoring algorithm was implemented that judges how well we did with respect to the training set.

Our scoring algorithm ranks the documents in the training set according to how they are ranked in the result set. Let $C$ = the number of documents known to be relevant in the training set. Let $r$ be the number of known-relevant documents that are listed as $C$th most-relevant or worse, and $p$ be the number of known-irrelevant documents that are listed $C$th most-relevant or better. Then $r + p = s$ is the score of a query result, and the higher the score, the worse the query performed. To put it plainly: ideally, the ranked training list would be the $C$ known-relevant documents in any order, followed by known-irrelevant documents in any order. This scoring algorithm measures how far the actual ranked list is from that ideal.

After scoring each run, we found that the two most successful runs (on average across all topics) were $k = 35$ and $k = 70$. We submitted those two runs as well as a pure LSI run as a control to measure the benefit of EDLSI over LSI.

## 3.3 TREC 2010 Legal Track Results Discussion

Tables 3.1 and 3.2 show hypothetical *F1*s and accuracies, respectively, for the best run, median run, and worst run of all TREC across all topics, as well as the performance of each of our three submissions, where $F1 = 2 \times (\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}})$. *F1* is a standard metric that attempts to balance the importance of precision and recall using the harmonic mean of the two. These *F1*s are denoted "hy-

| Topic | TREC Best | TREC Median | TREC Worst | $k = 35$ | $k = 70$ | LSI |
|---|---|---|---|---|---|---|
| 200 | 20.0% | 2.1% | 0.4% | 2.1% | 2.8% | 2.8% |
| 201 | 30.9% | 5.0% | 0.9% | 1.5% | 5.1% | 3.7% |
| 202 | 80.1% | 16.9% | 1.6% | 3.1% | 1.6% | 1.8% |
| 203 | 32.5% | 11.3% | 0.5% | 5.5% | 11.3% | 11.3% |
| 204 | 26.0% | 3.9% | 0.6% | 3.2% | 8.5% | 2.3% |
| 205 | 51.2% | 31.0% | 1.0% | 3.4% | 7.0% | 7.0% |
| 206 | 6.1% | 2.8% | 0.2% | 2.8% | 4.7% | 4.7% |
| 207 | 89.9% | 11.4% | 2.1% | 8.1% | 11.4% | 3.8% |

Table 3.1: 2010 Hypothetical *F1* Measurements for all topics: TREC Best/Median/Worst and Ursinus runs with $k = 35$, $k = 70$, and LSI

| Topic | TREC Best | TREC Median | TREC Worst | $k = 35$ | $k = 70$ | LSI |
|---|---|---|---|---|---|---|
| 200 | 75.3% | 15.4% | 0.9% | 13.4% | 26.3% | 26.3% |
| 201 | 85.1% | 35.4% | 1.0% | 17.5% | 64.5% | 38.2% |
| 202 | 97.1% | 35.7% | 1.5% | 35.7% | 77.0% | 78.3% |
| 203 | 89.1% | 34.1% | 0.6% | 35.0% | 63.1% | 63.1% |
| 204 | 99.4% | 49.7% | 1.3% | 81.7% | 79.6% | 8.0% |
| 205 | 68.9% | 30.8% | 1.0% | 10.2% | 8.6% | 8.6% |
| 206 | 83.0% | 5.1% | 0.3% | 7.5% | 11.1% | 11.1% |
| 207 | 89.4% | 32.0% | 7.2% | 76.2% | 44.7% | 45.7% |

Table 3.2: 2010 Relative Accuracy Measurements for all topics: TREC Best/Median/Worst and Ursinus runs with $k = 35$, $k = 70$, and LSI

pothetical" because they were calculated using an estimated count of relevant documents in the dataset as opposed to an actual count, as obtaining relevance judgments for each document in such a large dataset is not feasible[4].

For 6 of 8 topics, the $k$-selection of 70 in EDLSI was our best submission, which implies that if we had been able to select a higher $k$, the results would have been more accurate.

Relative to other TREC submissions, EDLSI was not an outlier, neither good nor bad. In all but two topics, the $k$=70 submission scored higher than the median across all TREC submissions. Its average relative accuracy was nearly double the TREC median relative accuracy, and 20 points below the

best submission for a topic. Interestingly, there were two topics where the $k$=35 submission outperformed the $k$=70 submission.

These results showed promise for EDLSI in this context, especially considering the primitivity of the learning we implemented. We went into 2011 with a fully functional EDLSI system and ideas for implementing machine learning, as well as a goal to analyze the characteristics of the topics on which the higher-$k$ EDLSI did not perform best, in order to use alternate algorithms on topics which share those characteristics.

# Chapter 4

# TREC Legal 2011

The primary objective of the TREC Legal Track in 2011 was to use of machine learning to improve retrieval. Teams attempted to implement information retrieval systems that are able to learn when relevance feedback is available. For each query, we were allowed to submit subsets of the dataset for relevance judgment to a TA. When those judgments were returned, teams updated their systems using that information in an attempt to improve future queries. In 2011 we improved our base system by implementing the R package, irlba, to increase the number of dimensions we could obtain with LSI, and by implementing a new method of machine learning, selective query expansion.

## 4.1 Preprocessing

The Enron Email dataset was used in both TREC 2010 and TREC 2011. Our preprocessing methods remained the same. They are outlined in sections 3.2.1, 3.2.2, and 3.2.3.

## 4.2 Computing the SVD

A package called "irlba" for the R language was released after work on TREC 2010 finished, but before work on TREC 2011 began[2]. It is designed to allow for further calculation of the partial SVD on massive matrices. We used this library to calculate the SVD of the ENRON corpus to 200 dimensions, as opposed to the 75 in 2010. Though we never tested exclusively for improvements due to dimensionality expansion, we believe this boosted the performance of the LSI component of our system.

## 4.3 Run Submission

In 2011, no relevance information is available until after the first run; therefore, at least one blind run had to be used. For this run we developed queries based on the request for information. These initial queries are shown in Tables 4.1, 4.2, and 4.3. We used EDLSI with weighting parameter of $x = .2$ (as outlined in 2.4) to run these queries.

The documents with the top 100 relevance scores from the blind run were sent to the TA for relevance determination. When the judgments for these 100 documents were returned by the TA, we modified our query based on this new information. We refer to this process as selective query expansion, because we are expanding the query with terms from documents we have selected specifically because they are known to be relevant. It is reasonable to assume that we can use these known-relevant documents as query vectors to return other documents that are similar, and hopefully relevant to the query. We implemented selective query expansion by taking the known-relevant documents, adding their vectors

| clickpaper | dealbench | energydesk | enromarkt |
|---|---|---|---|
| enroncredit | enrondirect | enrononline | enronweather |
| epoweronline | hottap | newpowercompany | water2water |

Table 4.1: Initial Query Terms for Topic 401

| capital | commission | commodities | congress | crude | derivatives | djia |
|---|---|---|---|---|---|---|
| dow | dowjones | exchange | federal | financial | foreign | ftc |
| funds | futures | gold | growth | illegal | international | invest |
| investment | investments | jones | law | laws | legal | london |
| markets | metals | mutual | nasdaq | nyse | Oil | options |
| parliament | poors | precious | products | regulation | regulations | reserve |
| rules | shanghai | silver | standard | stock | stocks | tokyo |

Table 4.2: Initial Query Terms for Topic 402

| environment | environment | environmental | oil |
|---|---|---|---|
| spill | emissions | emission | footprint |
| warming | legal | illegal | |

Table 4.3: Initial Query Terms for Topic 403

to the previous query vector, and using this new vector as the query vector in a new EDLSI run.

We had time to repeat this process multiple times before a final submission was required. In each iteration, we submitted an additional 100 non-judged documents to the TA and used the judgment information to expand our queries.

## 4.4 TREC 2011 Submission Details

Our runs were all produced by our EDLSI system. All submissions used a $k$ of 200 and a weighting factor $x$ of 0.2. The first submission for each topic was an EDLSI search with no machine-learning implemented. The query vectors we used were lists of terms we believed would be commonly found in relevant

documents, and were drawn from terms we saw in the request for information or in related literature. The subsequent submission for each topic used the same EDLSI parameters along with selective query expansion. The documents we used for query expansion were all documents judged to be relevant by the TA. In each iteration, we sent the top 100 unjudged documents returned in the previous search to the TA. Documents that were known to be relevant were given a .999 relevance probability, documents known to be irrelevant were given a .001 probability, documents with unknown relevance received a probability equal to the cosine similarity score. We repeated the process twice. Thus our final submission for each topic used relevance judgments from 200 documents. The mop-up run used the same EDLSI parameters and selective query expansion using all the relevance judgments collected from all teams.

## 4.5 TREC 2011 Results

The results released at the 2011 TREC conference show that the EDLSI system is a promising algorithm for use in E-discovery tasks and the results also allow us to identify interesting future research paths.

The results from all topics are shown in Tables 4.4, 4.5, and 4.6. These tables show both the estimated and actual *F1* scores over iterations of the machine learning process for all topics, as well as the estimated amount of relevant documents across the dataset for that run. The difference between actual and *F1* scores is a nuance in the TREC scoring system. Hypothetical scores represent the scores achieved when a cutoff rank of documents is chosen using information gleaned from a sample set of approximately 5600 documents per topic. The scores at this cutoff are "hypothetical" because they *could* be achieved, but only

if that sample set of documents had already been assessed to determine the optimal cutoff rank. These scores show some unexpected fluctuation throughout iterations of the learning process because the number of documents estimated to be relevant throughout the dataset changes over time. Actual scores infer an optimal cutoff rank using the sum of probabilities submitted up to all possible cutoff ranks (which is also an estimate of how many documents actually are relevant up to that cutoff), calculates *F1* there, and chooses the rank with the best *F1*. These scores are "actual" because they can be achieved only using information in the submission. These results are analyzed in section 4.5.1.

### 4.5.1 Results Discussion

In Table 4.4 we see that actual *F1* sees increases in both rounds of learning, while hypothetical *F1* sees a drop from 34.3% to 25.7% before increasing to 44.6% on the mopup run. Given that the estimated precision for topic 401 showed substantial increases in both rounds of learning (See Fig. 4.1), estimated recall must have experienced a drop. Actual *F1* does not see this decrease as the amount of documents *known* to be relevant does not change over iterations of the learning process. Topic 402 also shows a strictly increasing *F1* (see Table 4.5), while topic 403 shows fluctuation in actual *F1* over all runs (see Table 4.6). The initial drop in hypothetical *F1* is present in all topics. This may be because the estimated count of total relevant documents increases as runs are submitted. An increase in the count of relevant documents could lower the hypothetical *F1* score by lowering recall. For this reason, we closely examine estimated precision over iterations of the learning process.

| Run | Actual *F1* | Hypothetical *F1* | Est. Docs Relevant |
|---|---|---|---|
| Initial Run | 16.8% | 34.3% | 2393 |
| Run 2 | 17.3% | 25.7% | 5495 |
| Mopup Run | 42.7% | 44.6% | 18251 |

Table 4.4: Actual *F1*, Hypothetical *F1*, and Est. Docs Relevant for Topic 401 for all Runs

| Run | Actual *F1* | Hypothetical *F1* | Est. Docs Relevant |
|---|---|---|---|
| Initial Run | 3.6% | 8.6% | 5866 |
| Run 2 | 3.8% | 8.5% | 7592 |
| Mopup Run | 6.8% | 15.8% | 11523 |

Table 4.5: Actual *F1*, Hypothetical *F1*, and Est. Docs Relevant for Topic 402 for all Runs

| Run | Actual *F1* | Hypothetical *F1* | Est. Docs Relevant |
|---|---|---|---|
| Initial Run | 3.3% | 8.3% | 3420 |
| Run 2 | 1.2% | 8.2% | 9000 |
| Run 3 | 4.8% | 10.3% | 10099 |
| Mopup Run | 4.3% | 23.9% | 13369 |

Table 4.6: Actual *F1*, Hypothetical *F1*, and Est. Docs Relevant for Topic 403 for all Runs

Fig. 4.1 represents the estimated precision metric at various document cut-offs. Here we see that we obtained significant improvements on the first round of machine learning. Further runs proved to have diminishing returns, especially on topic 403. This may be due to our query expansion process tending towards documents that, while similar to the documents known to be relevant, are not actually relevant to the topic. This effect is more pronounced on topics 402 and 403, and this may be related to the nature of topic 401 versus that of 402 and 403 – topics relevant to 401 contain, intuitively, more terms that are specific to the topic at hand than 402 or 403.

The initial query used for topic 401 (See Table 4.1) consisted of terms specif-

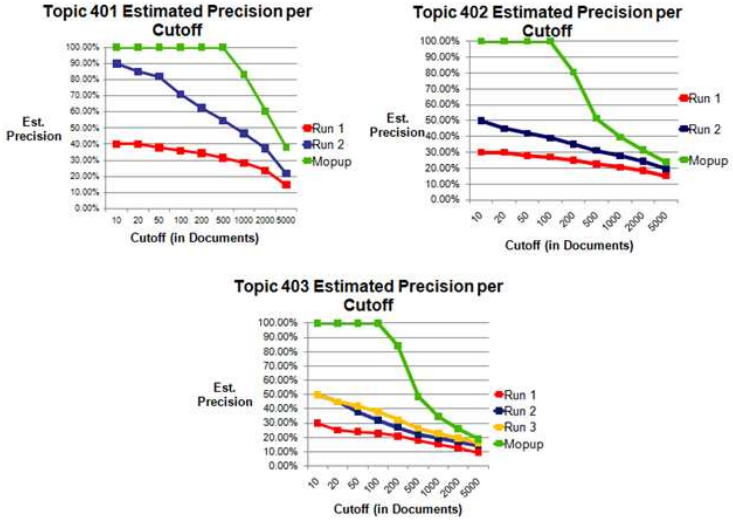


Figure 4.1: Ursinus System Estimated Precision per Cutoff across all runs, all topics

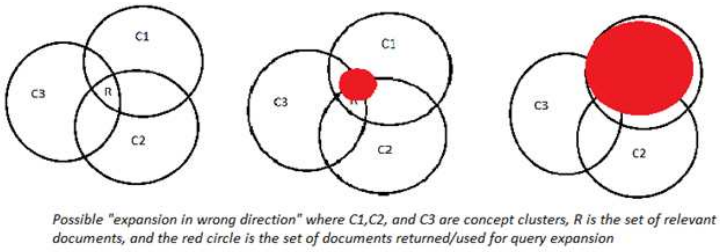


Figure 4.2: Graphical Representation of Expansion in the Wrong Direction

ically related to Enron's online systems. These are terms you would expect to find only in documents that discuss use and maintenance of Enron's online systems, and these terms co-occur with terms such as: link, search, servers, etc. that are also likely to be relevant to topic 401. Thus the machine learning can be expected to be effective.

Unlike topic 401, topics 402 and 403 used terms that are common throughout the dataset (see Tables 4.2 and 4.3), and this may have caused the machine learning to not perform as well as on topic 401. It is likely that terms common

to large subsets of the dataset may have returned documents that trained the machine learning system away from other relevant documents.

Intuitively, as suggested by the graphic in Fig. 4.2, given an initial query with terms that prove to be good discriminators, we would experience an increase in recall (especially in early iterations of the learning process) while precision stays the same or decreases slightly as the learning process is repeated. However, a more general initial query (one that has many terms that are not good discriminators), may return relevant documents that contain a lot of non-relevant terms in them. When these documents are used for query expansion, the non-relevant terms may be given too much weight. We refer to this process as "expansion in the wrong direction." One of the ways it would be reflected is in dimishing returns with respect to precision over iterations of the learning process, as more of the documents we expect to be relevant actually are not. This is precisely what we see over multiple runs for query 403, (see Fig. 4.1), and we believe this "expansion in the wrong direction" actually occurred.

# Chapter 5

# Cyberbullying Task

We have been working on integrating research on Cyberbullying detection into the LSI system. Cyberbullying is the "willful and repeated" use of information technology (social media, instant messaging, email, etc.) to harm someone, whether through trolling, flaming, cyberthreats, or other inappropriate approaches [9] [11]. A tool that could effectively detect cyberbullying would be useful both to guardians that want to provide protection for their children online without watching over their shoulders and to websites looking to moderate their content more efficiently.

## 5.1 Cyberbullying Dataset

The dataset we use for the cyberbullying task is drawn from Formspring.me, a social website used primarily by high-school and college students that allows users to ask anonymous questions of other users. The anonymity of the website makes cyberbullying common, so it is a perfect place from which to draw our

data.

The data is arranged in question/answer pairs. Each of these pairs is sent to Amazon's Mechanical Turk, a crowdsourcing application that lets us gather data from human judges about each pair: namely, whether or not the pair contains cyberbullying. If two of three judges say that a pair contains cyberbullying, we list it as containing cyberbullying. At the end of this process, we are left with documents that consist of a question, an answer, and a yes/no for bullying, which is analogous to the relevance judgments from section 3.2.5. Treating each question and answer as the text of the document, what we have is a training set of 13,652 documents with relevance judgments. The entire dataset contains 24135 documents; this dataset is analogous to that of the 2010 TREC Legal task.

This task is difficult in part due to the target demographic for software of this nature. Terms are often misspelled by children, both intentionally and unintentionally, increasing the termcount and decreasing the potential co-occurrence that might otherwise have been found in a dataset. One way we combat this danger is with term compression – that is, collapsing repeated characters in each term to one character. This reduces what might be many terms with the same meaning into a single term so that high-order term co-occurences can be inferred more easily by the LSI sytem.

## 5.2 Cyberbullying Experiments

Our experiments thus far with Cyberbullying have been analogous to the TREC Legal 2011 task. Given the training set of 13,652 documents, 916 of which were known to contain Cyberbullying according to Amazon's Mechanical Turk, we

are making attempts to find documents that contain Cyberbullying in another set of 10,483 unjudged documents from Formspring.me. We began by running an EDLSI search with $k = 500$ (a large $k$, but not so large that LSI's results begin to closely approximate those of Vector Space retrieval) and a weighting factor of $x = .2$ for the LSI portion, leaving $1 - x = .8$ for Vector Space (the same weighting parameters we used in TREC).

Sampling the data we have collected and labeled suggests that about 6.7% of a given Formspring dataset is likely to contain bullying. Based on that figure, we chose to submit the top 1000 documents ranked after an initial search to Amazon's Mechanical Turk in order to see how we performed at various cutoffs in the first 10% of our result set and in order to further build our training set. We will continue iterating in this way to see how much each new set of relevance judgments improves the performance of the EDLSI searches.

## 5.3 Cyberbullying Results Discussion

At the time of writing, we have only iterated one time through the Cyberbullying learning process, so data regarding improvements over iterations of the process is still unavailable.

Our initial query results (Table 5.1) show that our first attempt was a bit over-aggressive, with a 17% false positive rate in the first 100 documents. This may have been affected by the system's tendency to return documents filled with foul language that might not specifically be bullying (examples include sexual solicitation that was well-received by the answerer or questions about sexual preferences) and documents that might have been clear bullying of a third party not involved in the question or the answer (for example: trash-

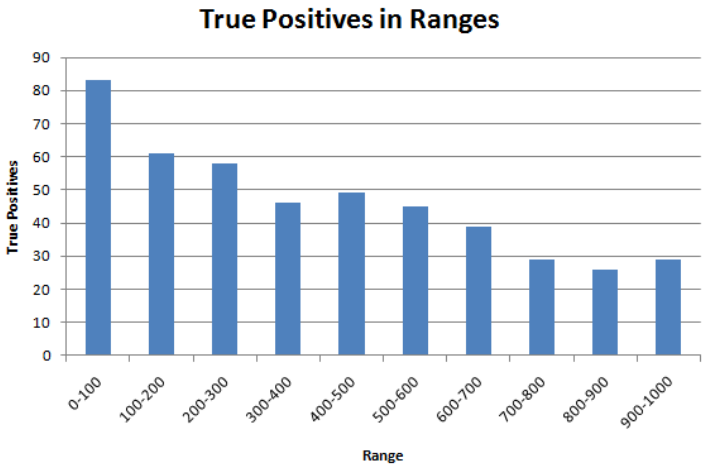


Figure 5.1: Amount of Bullying Documents Gained in Each Range

talking and name-calling of a girl by both the asker and the answerer) but were not marked as bullying, which may indicate more of a problem with the task we presented the Mechanical Turk workers than with our system. These documents may be of interest to parents, even if they don't explicitly contain bullying.

Recall grows approximately linearly with a 45-degree slope when documents are ranked at random. This forms a linear recall baseline. The initial query recall curve lies consistently above this linear recall baseline. Recall improvement grows fastest at earlier cutoffs (See Fig. 5.1), which shows that the results are frontloaded well: the bullying subset the system suggests is pushed towards the top of the ranked list and not distributed randomly or evenly.

Also of note are the 465 bullying documents found in the first 9.5% of the dataset being searched. If the 6.7% approximation for bullying in an arbitrary Formspring.me dataset holds true, then we have already found 69.4% of them in the first 9.5% of the dataset.

We plan to proceed by further expanding our query using the new bullying

| Cutoff | Recall | Precision | *F1* |
|---|---|---|---|
| 100 | 17.85% | 83.00% | 29.38% |
| 200 | 30.97% | 72.00% | 43.31% |
| 300 | 43.44% | 67.33% | 52.81% |
| 400 | 53.33% | 62.00% | 57.34% |
| 500 | 63.87% | 59.40% | 61.55% |
| 600 | 73.55% | 57.00% | 64.23% |
| 700 | 81.94% | 54.43% | 65.41% |
| 800 | 88.17% | 51.25% | 64.82% |
| 900 | 93.76% | 48.44% | 63.88% |
| 1000 | 100.00% | 46.50% | 63.48% |

Table 5.1: Recall, Precision, and *F1* at Various Cutoffs after Initial Query

documents we have discovered and by implementing additional learning features. For example, we might attempt to combat "friendly" bullying. Some documents were judged to not contain bullying presumably based on the presence of one or more smiley face emoticons (" :) ") when they would otherwise be clear examples of bullying.

# Chapter 6

# Conclusions and Future Research

For TREC 2011, we improved our EDLSI system by increasing the number of dimensions we use in LSI, and by implementing selective query expansion. EDLSI was found to be more effective for the topic with the most specific query terms (topic 401). Selective query expansion proved most effective on topic 401; the query expansion results for topics 402 and 403 show little improvement after the first round of relevance feedback is incorporated.

EDLSI shows promise on the Cyberbullying task as well, staying ahead of the linear recall baseline at all ranks, and leading to higher precisions at low ranks. This is useful for applications that monitor Cyberbullying situations, as we are getting closer to finding ideal cutoffs for notifying parents or moderators.

Certain types of documents were able to deceive the EDLSI system, such as consensual sexual solicitation and "friendly" bullying. We look forward to

documenting potential improvements over iterations of the learning process as well as adding extra learning features in order to improve the accuracy of our Cyberbullying search results by weeding out this type of document.

Other opportunities for future work include:

- Continuing $k$-optimization: The selection of $k$ in LSI and EDLSI is critical. If $k$ is too low, the LSI portion of the results will be inaccurate. If $k$ is too high, computation time increases substantially, and the latent semantic effect is watered-down. Finding an optimal $k$ for a dataset is an area of ongoing research.

- Negative weight analysis: Because we have selective query expansion working, we can see the positive effect using a known-relevant document for query expansion can have on a query. Going in the other direction, using a known-irrelevant document could help weed out irrelevant documents that might otherwise be returned as relevant. Clustering the large datasets we work with may assist in this process.

- Topic/Query analysis: Continued analysis of the characteristics of the topics with less successful results could lead to a better machine learning process; perhaps different algorithms should be used for different queries. Automatically determining the best algorithm based on query characteristics would be particularly useful, both for E-discovery and for the Cyberbullying task.

# Bibliography

[1] R. Baeza-Yates and B. Ribeiro-Neto (2011). *Modern Information Retrieval: The Concepts and Technology behind Search*. New York: Addison Wesley. Print.

[2] J. Baglama and L. Reichel (2011). irlba. http://illposed.net/irlba.html

[3] M. Berry, S. Dumais, and G. O'Brien (1995). Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):575-595.

[4] G. Cormack, M. Grossman, B. Hedin, D. Oard (2011). Overview of the TREC 2010 Legal Track. http://trec.nist.gov/pubs/trec19/papers/LEGAL10.OVERVIEW.pdf

[5] S. Deerwester, S. Dumais, T. Landauer, G. Furnas, and R. Harshman (1990). Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science*, 41(6):391-407.

[6] J. Demmel, J. Dongarra, B. Parlett, W. Kahan, D. Bindel, Y. Hida, X. Li, O. Marques, E. Riedy, C. Vmel, J. Langou, P. Luszczek, J. Kurzak, A. Buttari, J. Langou, and S. Tomov (2007). Prospectus for the next LAPACK and ScaLAPACK libraries. In *Proceedings of the 8th international Conference*

*on Applied Parallel Computing: State of the Art in Scientific Computing* (Ume, Sweden). B. Kgstrm, E. Elmroth, J. Dongarra, and J. Waniewski, Eds. Lecture Notes In Computer Science. Springer-Verlag, Berlin, Heidelberg, 11-23.

[7] A. Kontostathis (2007). Essential dimensions of latent semantic indexing (LSI). *Proceedings of the 40th Hawaii International Conference on System Sciences*. January 3-6, 2007

[8] A. Kontostathis and W.M. Pottenger. (2006). A framework for understanding LSI performance. *Information Processing and Management*. Volume 42, number 1, pages 56-73.

[9] I. McGhee, J. Bayzick, A. Kontostathis, L. Edwards, A. McBride, and E. Jakubowski. (2011). Learning to Identify Internet Sexual Predation. *International Journal on Electronic Commerce.* Volume 15, Number 3. Spring 2011

[10] P. Ogilvie and J. Callan (2002). Experiments Using the Lemur Toolkit,In *Proceedings of the Tenth Text Retrieval Conference (TREC-10)*, pages 103-108

[11] J. Patchin and S. Hinduja. (2006). Bullies move beyond the schoolyard; a preliminary look at cyberbullying. *Youth violence and juvenile justice.* 4:2 148-16

[12] D. Rohde (2007). SVDLIBC. http://tedlab.mit.edu/ dr/SVDLIBC/

[13] G. Salton and C. Buckley (1988). Term-weighting approaches in automatic text retrieval. *Information Process Management*,24(5):513-523.