Supervised Learning in the Wild: Text Classification for Critical Technologies

Arun S. Maiya, Francisco Loaiza-Lemos, and Robert M. Rolfe
Institute for Defense Analyses
4850 Mark Center Drive, Alexandria, VA 22311
Email: {amaiya, floaiza, rolfe}@ida.org

***Abstract*—We explore the problem of locating documents pertaining to critical technologies (*e.g.,* restricted, proprietary, or sensitive technical information) from among a massive and highly heterogeneous collection of largely unimportant files. We present a system that employs the use of supervised machine learning (*i.e.,* pattern recognition) to detect such critical documents. To address difficult or ambiguous instances, we supplement the text classifier with an *automated* keyword search. That is, we extract, in an automated fashion, discriminative terms (*i.e.,* keywords) from the training set and match them against documents during the classification process. We demonstrate the effectiveness of this hybrid approach through a series of validation tests and case studies.**

I. Introduction

We explore the problem of locating documents pertaining to critical technologies from among a massive and highly heterogeneous collection of largely unimportant files. We refer to such sensitive documents simply as *critical documents*, which may contain restricted or proprietary information. We explore the extent to which we can detect critical documents on user workstations, laptops, and file servers (an environment we refer to as "in the wild" given the high level of heterogeneity among files on workstation hard drives). Such a capability has applications to several areas including consequence management of cyber security breaches. How does one go about finding such "needles in haystacks" or even determining if any "needles" exist on a particular system? The standard approach to making such a determination is to employ the use of *keyword searches* (or queries). The documents returned by the search are then manually reviewed by a human to determine and confirm relevancy to a given critical topic. (Throughout this paper, documents relevant to a critical technology of interest are referred to as *positive*, while documents unrelated to the critical technology are referred to as *negative*.) Such keyword searches to locate critical (or positive) documents pose several challenges.

Keyword Search Challenges:

1) **Ad hoc Nature of Queries:** The keywords (or terms) used in a search query are often chosen in an *ad hoc* manner. For certain topics, it may be unclear as to which keywords are best to efficiently hone in on the most relevant documents. We find that this especially holds true for documents pertaining to critical technologies.
2) **Over-Constrained Queries:** The keywords in a query may not be fully representative of the topic. Thus, many relevant documents will be missing from the search results (*i.e.,* false negatives).
3) **Under-Constrained Queries:** The keywords may appear in numerous documents in a text corpus including many *non-relevant* documents (*i.e.,* false positives). Moreover, certain words have multiple meanings (*e.g.,* the acronym CAT will be present in documents referring to cats). This last point is especially pertinent to the DoD, which engages in frequent use of acronyms.
4) **Relevancy:** Not all relevant documents returned by a keyword search are *equally* relevant. Moreover, the frequency of a keyword within a document may or may not have a connection to document relevance. For instance, a PowerPoint briefing highly pertinent to a certain technology may, in fact, only include the *name* of the technology in the title slide while discussing esoteric aspects of the technology in the remaining slides. We find that standard document relevancy (or ranking) measures such as TF-IDF and BM25 can prove inadequate under such scenarios (see [9] for more information on these measures).

Indeed, these challenges have resulted in a significant burden on the staff of the organization sponsoring the present work, which motivated the authors to investigate alternative solutions. Here, we investigate the use of *supervised machine learning* (*i.e.,* pattern recognition) to detect and locate critical documents. In supervised learning, given a set of both positive examples (*i.e.,* sample documents representing the topic of interest) and negative examples (*i.e.,* sample documents *not* representing the topic of interest), a mathematical model describing how words in a document affect its topic is constructed. This model is referred to as a *classifier*[1] and is used to correctly categorize (or classify) documents as either pertaining to the topic of interest or not. The example documents used to construct the classifier are referred to as the *training set*, and the process of constructing a classifier from these examples is referred to as *training* or *learning*. The *precision* of a classifier is the fraction of retrieved documents that are relevant to the search [9]: precision =

[1]Note that, throughout this paper, we use the term "classify" in the general sense of "categorize" as opposed to relating to restricted information.

$\frac{|\text{relevant documents} \cap \text{retrieved documents}|}{|\text{retrieved documents}|}$. The *recall* of a classifier is the fraction of *all* relevant documents successfully retrieved [9]: recall $= \frac{|\text{relevant documents} \cap \text{retrieved documents}|}{|\text{relevant documents}|}$. Precision and recall can be combined into a composite score known as the *F-score*.

An ideal classifier can *potentially* address all four of the aforementioned problems associated with standard keyword searches. For instance, a classifier trained to achieve both high precision and high recall minimizes both false positives (under-constrainment) and false negatives (over-constrainment), respectively. Moreover, under certain machine learning approaches, *confidence* scores can be assigned to documents to infer relevancy to a given topic. A machine learning approach also has the added advantage of bypassing the need to manually determine an optimal keyword set for use in a query. It is the learning algorithm that implicitly determines, in an automated fashion, the importance of words to various topics. For all these reasons, we explore the use of supervised machine learning to address the problem of finding critical documents. Machine learning, however, faces its own unique challenges that must be addressed.

Machine Learning Challenges:

1) **Class Imbalance:** For many realistic scenarios, the class distribution is skewed heavily towards one category (or class) and away from the other. More often than not, it is of primary interest to find or detect the rare instances compromising the *minority* class. For the problem domain covered in this work, documents pertaining to critical technologies often compromise a very small minority of all possible files on all possible hard drives.[2] It is well-known that such imbalances cause problems for machine learning algorithms and tend to deteriorate the accuracy of trained classifiers (*e.g.,* see [3]).
2) **High Variance of Negative Class:** In this particular problem domain, the negative class is not only substantially larger than the positive class but also exhibits an extremely high degree of *variance* (*e.g.,* heterogeneity). For instance, a given file system of a workstation may not only contain critical documents but also non-critical work-related documents, operating system and library files, browser cookies, and personal files such as resumes, recipes, or perhaps downloaded articles. Thus, the level of heterogeneity is significantly higher than in typical machine learning settings (*e.g.,* classifying news articles on a news website). If the negative examples used in training do not adequately reflect this heterogeneity, then this can significantly decrease precision. That is, negative documents not adequately represented by the negative training examples may be misclassified as positive.
3) **Scarcity of Examples:** We find that there often exists a scarcity of examples for use in training the classifier. For a variety of reasons, subject matter experts typically can supply us only with a small set of *positive* example documents pertaining to a critical technology. From this, problems can manifest in two ways. First, if supplied positive examples are not fully representative of the positive class, the recall suffers, as positive documents may be unrecognized by the classifier. Second, it is unclear on how one should go about finding and selecting negative examples for inclusion into the training set (especially given the high variance and large size of the negative class). Poor choices here can increase the false positives and hurt precision.
4) **Embedded Information:** In some cases, given a critical technology, a largely negative document may contain a small bit of information pertaining to the positive class (*i.e.,* the critical technology). These documents tend to be missed by the classifier, as the document, on the whole, is more characteristic of the negative class. However, despite the fact that these cases would not typically be considered false negatives, we would still like to detect such instances in large text corpora.

[2]We should note, however, that on particular file systems used by individuals heavily involved in a critical technology area, there might be a larger number of documents containing keywords related to the technology. Such scenarios contribute to the inadequacy of simple keyword searches.

Overview of Approach. In this work, we present a *hybrid* system that seeks to address all eight of the aforementioned challenges. To mitigate aforementioned issues associated with conventional keyword searches in this problem domain, we focus primarily on supervised machine learning. To address the machine learning challenges of class imbalance and high negative variance, we employ a technique in which the negative training set is iteratively grown by sampling only the most *informative* documents from the negative class. Finally, we develop an intuitive term weighting measure to extract the most discriminative keywords from the training set in an automated fashion. For documents labeled as *negative* by the classifier, in a second phase of analysis, we match these discriminative keywords against documents labeled negative and sort results by a function of hit count (descending order). This supplemental *automated keyword search* serves to help hone in on both *false negatives* arising from a scarcity of training examples and *true negative* documents embedded with a small bit of *positive* (and possibly critical) information. Thus, all eight of the problems are addressed by this hybrid system. We demonstrate its effectiveness through a series of validation tests and case studies.

II. Proposed Method

We now present our approach for the detection of critical documents. We begin by briefly describing the core of the approach: machine learning.

A. The Learning Algorithm

We employ the use of Linear Support Vector Machines (LinearSVM) as our main learning algorithm [4]. LinearSVMs have been shown to be both scalable and well-suited to classifying high-dimensional but sparse data such as text documents (*e.g.,* see [2], [4], [7]). Formally, we begin with a set of labeled documents $D = \{(d_1, y_1), \ldots, (d_n, y_n)\}$ from which we will build our document classifier, where y_i is

the topic label of document d_i for all i.[3] Each document in the training set D is represented as a bag of words (*i.e.,* a multiset structure). To employ the use of machine learning, we must transform each document $d_i \in \{d_i\}_{i=1}^{n}$ into a *feature vector* $\vec{x_i}$ for use in machine learning. Each component of the vector represents a term (*e.g.,* word) from the vocabulary of D, and the component value is assigned using a well-studied weighting scheme referred to as TF-IDF. More information on such representations can be found in [9], [10].

A LinearSVM algorithm accepts $\hat{D} = \{(\vec{x_1}, y_1), \ldots, (\vec{x_n}, y_n)\}$ as input. A linear separator (or hyperplane[4]) that best divides the positive documents from negative documents is sought. This can be stated mathematically as an optimization problem [2], [5]. For brevity and focus, formal presentations and derivations of the problem are omitted.[5] However, the basic idea can be graphically illustrated through simple example. In Figure 1, of the four possible hyperplanes that divide the red circle points and green square points, it is the hyperplane shown in plot (D) that leaves the largest margin and best divides the points. New documents supplied to the algorithm are plotted in this space and classified as positive or negative based on which side of the hyperplane they fall. We conclude this section by noting that the distance of a new document from the hyperplane can be employed by the classifier as a score of confidence in the assignment of documents to particular categories (positive or negative in the binary case). Next, we discuss the extraction of discriminative terms from training sets.

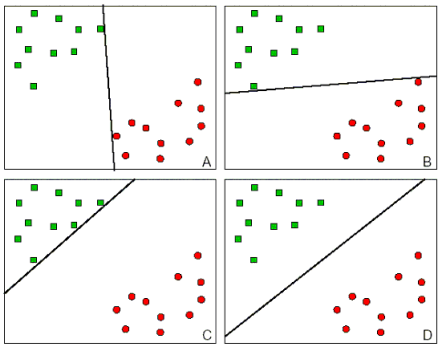

Fig. 1. SVM classifiers are essentially linear separators that best divide existing data. In this simple 2-dimensional illustration, it is the linear separator in (D) that best divides the points (and is, thus, chosen).

B. Extracting Discriminative Terms from Training Sets

Which terms are the most discriminative in accurately distinguishing critical documents? Such terms have two uses in our work (to be discussed in greater depth later): 1) retrieving potentially *informative* negative examples for use in training and 2) performing an *automated keyword search* to "catch" critical documents missed by the classifier. We leverage the concepts of *entropy* and *information gain* from the field of information theory [9]. The *entropy* H of a set of labeled documents D measures impurity as follows: $\mathrm{H}(D) = -p^+ \log_2(p^+) - p^- \log_2(p^-)$, where p^+ and p^- are the proportions of positive and negative documents in D, respectively.[6] For instance, if all documents are positive (or negative), $\mathrm{H}(D) = 0$, while a perfectly even split of positive and negative documents has entropy of 1. The information gain IG of a word w in training set D, then, is the expected entropy reduction due to segmenting on w:

$$\mathrm{IG}(w, D) = \mathrm{H}(D) - \frac{|D^w|}{|D|}\mathrm{H}(D^w) - \frac{|\overline{D^w}|}{|D|}\mathrm{H}(\overline{D^w}),$$

where D^w is the set of documents in D containing word w. Thus, words with the highest information gain in a training set are expected to be the most discriminative. We also devise a second intuitive term weighting measure to extract discriminative terms we refer to using the somewhat oxymoronic phrase, "probably necessary but insufficient" or PNI. Intuitively, the most discriminative terms will be present in the positive class with *high* probability (*i.e., probable necessity*) but may also occur in the negative class with lower probability (*i.e., insufficiency*). Our objective, then, is to identify terms with *maximal* necessity and *minimal* insufficiency as follows:

$$\mathrm{PNI}(w, D) = \frac{\mathrm{pdf}(w) + 1}{|D^+|} \cdot \log_2 \frac{|D^-|}{\mathrm{ndf}(w) + 1},$$

where $\mathrm{pdf}(w)$ is the positive document frequency of w (*i.e.,* the number of positive documents containing w), $\mathrm{ndf}(w)$ is the negative document frequency of w, and D^+ and D^- are the sets of positive and negative documents (respectively). Due to the sensitive nature of critical technologies, we cannot demonstrate keyword extraction on actual data from our problem domain. Thus, we demonstrate performance on the *20 Newsgroups* dataset [1], a standard benchmark dataset in text classification. The *20 Newsgroups* dataset is a collection of documents from twenty newsgroups each covering a different topic. We consider three randomly selected newsgroups and use each as the positive class, while considering the remaining 19 groups as the negative class. As shown in Table I, both measures are equally effective in extracting words most intuitively representative of each newsgroup topic. For this work, we choose to use PNI for keyword extraction.

C. Developing Training Sets

We now turn our attention to construction of training sets, the input to both the LinearSVM and PNI algorithms. Recall from earlier that adequate training examples for this problem domain are scarce. We are typically supplied with just a small set of only *positive* training examples. Thus, we initially complement this training set with a small selection of random DoD

[3]For instance, $y_0 = 1$ if document d_0 is a critical document, and $y_0 = 0$ otherwise.

[4]A linear separator, formally referred to as a hyperplane, divides an n-dimensional space in half. For instance, in 2-dimensional space, a linear separator is simply a line. In 3-dimensional space, it is a plane.

[5]For more information, the reader may refer to the original paper by Cortes and Vapnik [2].

[6]Note that $\log_2(0)$ is taken to be 0.

Newsgroup	Information Gain	PNI
sci.electronics	circuits, voltage, amp, electronics, detector	circuits, voltage, electronics, amp, power
sci.space	space, orbit, nasa, moon launch	space, orbit, nasa, moon, launch
talk.politics.guns	guns, firearms, weapons, batf, atf	guns, firearms, weapons, batf, atf

TABLE I
TOP 5 DISCRIMINATIVE TERMS (*i.e.,* KEYWORDS) EXTRACTED FOR THREE NEWSGROUPS IN THE *20-Newsgroups* [1] TRAINING SET. BOTH MEASURES ARE EQUALLY EFFECTIVE IN EXTRACTING KEYWORDS THAT BEST CAPTURE THE TOPIC OF EACH NEWSGROUP.

technical reports unrelated to the topic of interest (*i.e.,* negative examples). This is our *initial* but incomplete version of training set $\mathcal{D}$. We find that, for even mediocre performance, the training set must be significantly augmented and transformed. The positive examples are sensitive, supplied by subject matter experts, and are, for all practical purposes, fixed. Thus, to improve classifier performance, we must focus on augmenting the *negative* examples. As described previously, the negative class is characterized by significantly larger size and variance than that of the set of critical documents in which we are interested. Our first task is to *grow* the pool of negative documents to better capture the size and variance of the negative class. However, since our negative class consists of almost any kind of file that could ever be present on a computer hard drive, it is a virtually infinite set. Thus, we focus on acquiring only the most *informative* negative examples.

What are *informative* examples? Refer to Figure 1 and note that the placement of the hyperplane is wholly dependent on those points (*i.e.,* documents) that are closest to the "border" between the positive and negative class. Such "border-line" examples are referred to as *support vectors* and are used by the LinearSVM learning algorithm to approximate the optimal placement of the hyperplane. This fact yields a key insight: examples *not* located near the border can be discarded, as they have no effect on the learned classifier or its ultimate performance. The retained examples near the hyperplane are referred to as *informative*. This leads us to an intriguing approach to improving the set of negative examples: using the initial (or most recently trained classifier) to seek out negative examples that are either near the hyperplane of a classifier (*i.e.,* ambiguous) or completely misclassified by a classifier. In the process, we are able to iteratively improve the classifier by, in effect, *exploiting* its own inadequacies. This process, known as *active learning* [3], is realized in three distinct ways.

1) Growing the Pool with Non-Technical Informative Examples: First, we run our initial classifier on a random workstation hard drive on which, with near perfect certainty, no critical documents are present (*e.g.,* the personal laptop of the first author). In the trained classifier, the distance of support vectors from the hyperplane is normalized to 1 (for positive support vectors) and -1 (for negative support vectors). Thus, we should take documents present on the hard drive with hyperplane distance values of greater than -1 and add them to the pool of negative examples.[7] Note that this will include documents misclassified as positive in addition to documents correctly classified as negative but near the border. This step is effective in acquiring informative system or application files that are likely to be encountered on many hard drives. By incorporating them into the pool of negative examples, we are able to accurately disregard them during the text classification process.

2) Growing the Pool with Technical Informative Examples: Second, we take the discriminative terms extracted from the training set using our PNI measure and use them to conduct keyword searches on both Google Search and Google Scholar. Our aim here is to locate documents that contain terms highly relevant to the critical technology in question but, at the same time, are wholly unrelated[8] to the critical technology. We find that this heuristic is reasonably effective in locating highly informative (or "border-line") negative examples. Such training documents contribute significantly to the robustness of the final classifier.

3) Balancing the Training Set: At the conclusion of the previous two steps, we typically arrive at a pool of negative examples that is significantly larger than the set of positive examples. Our final step is to filter down the constructed pool of negative examples. We correct this balance by again leveraging hyperplane distance and the inadequacies of the most recently trained classifier. We begin with a single positive example and a single negative example (randomly drawn from our pool) and train a classifier based only on these two examples. We, then, run the resultant classifier on the remaining examples. The example that falls closest to the hyperplane is added to the training set, and a new classifier is trained only on these three examples. The process repeats until no further examples are nearer to the hyperplane than the support vectors, which corresponds to a distance threshold of ± 1. This approach to "under-sampling" the training pool serves to both select out the most informative (or "best-of-the-best") examples and also to remove redundancy from the training set. Interestingly, in the process of these iterations, the training set becomes *balanced*, as sets of examples near the border tend to be more balanced than the dataset as a whole [3]. Thus, by sampling only the most informative examples and removing redundant ones, we are able to also correct the class imbalance.

We conclude this section by noting that these three phases of augmenting the set of negative training examples can be repeated in an iterative fashion to arrive at the final training set. We now turn our attention to the automated keyword search.

[7] In practice, we generally only add documents with hyperplane distance values of greater than -0.1, as we find this is sufficient.

[8] Since positive examples consist of largely sensitive information, most or all documents located through public search engines will be decidedly negative examples. Our objective here is to find those examples that are likely confusing to our existing classifier.

D. Automated Keyword Search

As described previously, by improving the set of negative examples, we are better able to improve the *precision* of our final classifier. However, we have not augmented the set of *positive* examples in any way. Depending on the dataset, this can potentially deteriorate the *recall* of our classifier. That is, if the positive training examples do not adequately reflect or represent all critical documents in the positive class, then some encountered critical documents may be misclassified as negative. As mentioned previously, sets of positive examples for critical technology areas are typically limited, supplied by subject matter experts, and cannot be further augmented. Thus, we must find alternative means to assist analysts in honing in on false negatives. We extract keywords from the training set using the PNI measure described in Section II-B. During the text classification process, we match the top 15 keywords against each document analyzed. The final results, then, not only show the assigned class of the document (*e.g.,* positive or negative), but show keyword matches as an aid to analysts.

E. Putting it All Together

Using the algorithms and methodologies described in the previous sections, we combine the classifier and automated keyword search into a single, cohesive application. On the back end of the application, we implement a text extraction and indexing engine that combs through documents on a file system, extracts plain text from various document formats (*e.g.,* PDF, Microsoft Office), parses the text, and stores word counts into a relational database. For the front end, we design the graphical user interface for the system as a local Web application, which allows users to control aspects of the text extraction, indexing, machine learning classification, and automated keyword search. The results of the classifier and automated keyword search are output to an Excel spreadsheet. Each row shows the filename of the document, whether or not the document is categorized as pertaining to a critical technology, the confidence score of the categorization (*i.e.,* distance to hyperplane), and the automated keyword matches. Finally, we sort documents assigned to positive classes by confidence score and sort documents assigned to the negative class by cosine similarity [9] to the keyword matches (both sorts are in descending order). In the next section, we will evaluate the extent to which this Excel report allows analysts to quickly hone in on the most critical documents on large file systems.

III. Experimental Evaluation

For the purpose of this evaluation and to demonstrate our approach, we focus on the binary classification problem of finding documents pertaining to a very particular critical technology. Due to the sensitive nature of our problem domain, we will simply refer to this critical technology as Technology X. We constructed a training set for Technology X using the methodology discussed in Section II-C. Our final training set consisted of 51 positive examples of Technology X and 53 negative examples for a total of 104 training examples. We begin by discussing validations tests.

A. Validation Tests

1) 10-Fold Cross-Validation: Using our constructed training set, we first performed *10-fold cross-validation*, a standard technique used to estimate the extent to which the performance of a trained classifier will generalize to new, independent datasets (*i.e.,* real-world settings) [9]. As shown in Table II, the LinearSVM classifier performed exceedingly well overall, especially given the relatively small number of training examples available to us.[9] (For comparison purposes, we also show results for the Naive Bayes learning algorithm [9], one of the relatively worse performing learning algorithms.) We also observed the precision of the classifier to be relatively higher than the recall. We attribute this to the fact that the set of positive examples for our problem domain is limited and fixed. Finally, we found that, on average, the automated keyword search ranked the false negatives within the top 3 of those assigned to the negative class.

Algorithm	Precision	Recall	F-Score
LinearSVM	0.98 ± 0.02	0.93 ± 0.03	0.95 ± 0.02
Naive Bayes	0.77 ± 0.03	0.92 ± 0.03	0.83 ± 0.02

TABLE II
10-Fold Cross-Validation Results. Mean precision, recall, and F-score (with standard errors) for Technology X. Such results validate our choice of LinearSVM.

2) A "Needle in Haystack" Test: For a more realistic setting, we design the following validation test. We took five additional positive examples on which we did *not* train and a server for which we were confident that no documents pertaining to Technology X existed. The server consisted of nearly half a million files in total. We inserted these five files into the file system of the server and executed our classifier on the machine. When sorting all files by distance to hyperplane (*i.e.,* confidence), we would expect the five documents pertaining to Technology X to appear at the very top, and this is precisely what we observed. In this test, none of the remaining negative documents on the server were misclassified as positive. Moreover, the automated keyword search confirmed that no additional documents pertaining to Technology X existed on this system.

B. Case Study: A Confirmed Positive Case

We acquired an external hard drive on which the home directories of six users were copied. Exhaustive and time-consuming reviews by analysts indicated that this drive contained at least a few critical documents pertaining to Technology X (embedded in a plethora of work-related documents unrelated to Technology X). In total, the drive consisted of 25,007 documents, mostly technical in nature. Upon execution of our system, the classifier identified 15 documents as pertaining to Technology X. A manual review by a subject matter expert confirmed that all 15 documents did, in fact, pertain to Technology X. That is, they were *true positives*. Two of the 15

[9]We also note that the average area under the ROC curve for LinearSVM was 0.99.

documents were near the hyperplane and are, therefore, good candidates for future inclusion into the training set.

Next, to develop an approximation for ground truth, we had two analysts conduct a manual analysis of the drive using their existing practices: manual keyword searches using *ad hoc* queries. We note that the manual search took a total of 12 person hours, while the classifier (and automated keyword search) typically complete their processing in seconds. A total of 21 documents were manually identified by the two analysts as being related to Technology X. This set included all 15 of the documents identified by the classifier, which left six remaining documents requiring further examination. A review by a subject matter expert revealed that three of the six were, in fact, *unrelated* to Technology X (*true negatives* from the perspective of the classifier). One of the files was determined to be a legitimate critical document (a *false negative* for the classifier). The final two were documents with *embedded information* (referred to hereafter as *embedded documents*). Recall from Section I that such files are *true negatives* (in the sense of machine learning), since, as a whole, the documents do not pertain solely to Technology X. For instance, one of the documents was a PowerPoint brief describing various success stories of different technologies. Only one of the slides within this document discussed Technology X, while the remaining slides discussed other technologies. Our *automated keyword search* was specifically developed for detection of such documents in addition to detecting *false negatives*.

When sorting documents assigned to the negative class in descending order by cosine similarity, we found that the sole *false negative* and one of the two *embedded documents* were both within the top 20 (out of nearly 25,000 documents), while the second *embedded document* was within the top 70. Analysts confirmed that such a ranking allows for efficient detection of critical information. Finally, we found that the use of *only* automated keyword searches was wholly inadequate, as some of the richest documents pertaining to Technology X did not appear anywhere near the top and the signal-to-noise ratio was quite low. This further motivates our use of text classification as the primary retrieval technique in this hybrid approach. Results are summarized in Figure 2 as a Venn diagram.

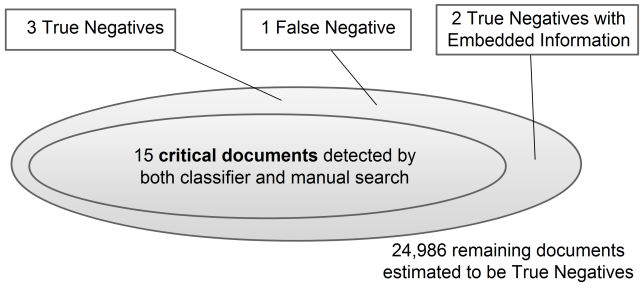

Fig. 2. Venn diagram showing results for our case study. The inner ellipse is the document set returned by the classifier, while the outer ellipse is the document set returned by a tedious manual search. (True/False and Positive/Negative assignments are from the perspective of the classifier.)

IV. Related Work

To the best of our knowledge, there is very little work on the use of text classification for the detection of sensitive subject matter such as critical technologies. Literature that most closely resembles our work involves the use of machine learning to segment scientific articles by topic (*e.g.,* see [6]). However, our problem domain is somewhat more difficult than those addressed in most papers, as the variance and size of the negative class is significantly higher. Also related to our work, as referenced earlier, is the large body of literature on scalable machine learning techniques (*e.g.,* [4]), the class imbalance problem in machine learning (*e.g.,* [3]), and document retrieval and relevancy (*e.g.,* [10]). Finally, since we are initially supplied with a set of only positive examples, research on learning from only positive and unlabeled examples (or *PU learning* [8]) is very loosely related to our work. However, PU learning methods are largely inapplicable to our domain of study, since there is, in fact, an abundance of easily-identifiable negative examples, and the key is really to sample from them.

V. Conclusion

In this paper, we have presented a hybrid system that exploits the use of both supervised machine learning and automated keyword searches to detect critical documents from among a large collection of unimportant files. Through a series of validation tests and case studies, we find that such a hybrid approach is effective in assisting analysts in locating critical documents on large file systems. In addition to the cases described in Section III, we have subsequently tested the system on several additional cases with equally successful results. There are several additional areas we plan to investigate in the future. These include the application of *unsupervised* learning techniques to this problem (*e.g.,* latent semantic analysis and latent Dirichlet allocation).

References

[1] Ana Cardoso-Cachopo. *Improving Methods for Single-label Text Categorization*. PhD thesis, Instituto Superior Técnico, October 2007.

[2] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995.

[3] Seyda Ertekin, Jian Huang, Leon Bottou, and Lee Giles. Learning on the border: active learning in imbalanced data classification. In *CIKM '07*.

[4] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research*, 9:1871–1874, August 2008.

[5] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, corrected edition, July 2003.

[6] Qi He, Bi Chen, Jian Pei, Baojun Qiu, Prasenjit Mitra, and Lee Giles. Detecting topic evolution in scientific literature: how can citations help? In *CIKM '09*.

[7] Thorsten Joachims. Text categorization with Support Vector Machines: Learning with many relevant features Machine Learning: ECML-98.

[8] Bing Liu, Yang Dai, Xiaoli Li, Wee S. Lee, and Philip S. Yu. Building Text Classifiers Using Positive and Unlabeled Examples. In *ICDM '03*.

[9] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge University Press, 1 edition, July 2008.

[10] Ho C. Wu, Robert Wing Pong Luk, Kam F. Wong, and Kui L. Kwok. Interpreting TF-IDF term weights as making relevance decisions. *ACM Trans. Inf. Syst.*, 26(3):1–37, June 2008.