The "BigSE" Project: Lessons Learned from Validating Industrial Text Mining

Rahul Krishna, Zhe Yu, A. Agrawal, Tim Menzies
Com Sci, NC State, USA
{ rkrish11, zyu9, aagrawa8, tjmenzie}@ncsu.edu

Manuel Dominguez, David Wolf
LexisNexis, Raleigh, USA
{manuel.dominguez, david.wolf}@lexisnexis.com

ABSTRACT

As businesses become increasingly reliant on big data analytics, it becomes increasingly important to *test* the choices made within the data miners. This paper reports lessons learned from the *BigSE Lab*, an industrial/university collaboration that augments industrial activity with low-cost testing of data miners (by graduate students).

BigSE is an experiment in academic/ industrial collaboration. Funded by a gift from LexisNexis, BigSE has no specific deliverables. Rather, it is fueled by a research question "what can industry and academia learn from each other?". Based on open source data and tools, the output of this work is (a) more exposure by commercial engineers to state-of-the-art methods and (b) more exposure by students to industrial text mining methods (plus research papers that comment on methods on how to improve those methods).

The results so far are encouraging. Students at BigSE Lab have found numerous "standard" choices for text mining that could be replaced by simpler and less resource intensive methods. Further, that work also found additional text mining choices that could significantly improve the performance of industrial data miners.

KEYWORDS: E-Discovery, Software Engineering, Testing.

1 Introduction

Much has been written about the application of data mining to software engineering. It is now routine to see at major SE conferences that a third (or more) of the papers used data miners to augment their analysis. Clearly, data mining has much to teach software engineering, but what about the other way around? What can software engineers teach data mining? What are the lessons learned from decades of SE that can improve data mining?

In 1975, Fred Brooks noted that half the effort of a software project is spent in *testing*. In an update to that book [11], written twenty years later, Brooks still asserted that testing remains a large task within any project. Accordingly, we should expect that when industrial data mining providers ship analytic tools, they should conduct extensive testing of those tools prior to release.

Some parts of commercial data mining tools are thoroughly tested prior to release (e.g. distributing tasks across a CPU farm or array of disk storage). However, other parts may not be so extensively explored. For example, LexisNexis is an international commercial company that offers Big Data solutions to clients. LexisNexis has invested time writing support and mining tools to assist with *E-discovery* (discussed in the next section). When LexisNexis ships E-Discovery products, those products contain numerous text mining *operators* to handle (for example) tokenization, featurization, normalization, classification, etc. Once a set of operators offers promising results, standard practice is to configure the text mining tool with those operators, then ship the resulting product.

LexisNexis contacted NcState with the question of "how can we validate if the operators we select for text mining are satisfactory?". This is a tricky problem in a commercial environment since an extensive exploration of data mining operators for tokenization, featurization, normalization, classification, etc. is a massive task. Having run such studies for many years [17], we can assert that this is mostly a trial-and-error process with a high percentage of errors. Commercial practitioners, responding to market pressures, may not be willing to explore this space of option since the large numbers of negative results are not consistent with an approach that quickly delivers incremental value to the market:

- The advantage of this approach is that, using it, commercial companies can maintain or extend their revenue stream.
- That said, it discourages extensive experimentation, and does not reward long sequences of negative results as data scientists try various options.

The trials and errors associated with exploring text mining operators are more suited to a research environment where persistent effort, with only occasional positive results, is more acceptable. Hence, LexisNexis and NcState jointly created the *BigSE Lab* where university graduate students explore the space of text mining operators proposed by industrial engineers. Funded by a gift from LexisNexis, the project has no specific deliverables. Rather, it is fueled by a research question "what can industry and academia learn from each other". Based on open source data and tools, the output of this work is (a) more exposure by commercial engineers to state-of-the-art methods and (b) more exposure by graduate students to more industrial text mining methods (plus research papers that comment on methods on how to improve those methods).

This paper documents the lessons learned from the BigSE project. These lessons divide into *process insights* that comment on methods for organizing this kind of collaboration and *technical lessons* that comment on different operators for text mining.

2 About the Domain: E-Discovery

The specific task explored in this work was *E-Discovery*. E-Discovery is part of civil litigation where one party (the producing party), offers up materials which are pertinent to a legal case. The producing party makes the materials available to a requesting party. Upon reception of a request, it is the duty of the producing party

BIGDSE'16, May 16 2016, Austin, TX, USA

DOI: http://dx.doi.org/10.1145/2896825.2896836

to make an inquiry to find all reasonably relevant materials in their possession and turn them over to the requesting party. For modern companies, this may mean searching through millions emails to find (say) 100 emails that are most relevant to the case at hand.

This process is extensively monitored by both sides involved in the litigation, as well as the judges overseeing the process. Companies can be heavily sanctioned if they fail to conduct a reasonable search in good faith, or if they do not respond in a timely manner[1]. If the producing party is perceived to be working in bad faith during this process, then this can result in court sanctions. Hence, there is much commercial interest in E-Discovery tools that are thorough and discourage over-use, misuse, or abuse of procedural tools that increase cost and result in delay.

Initially, document discovery was a mostly manual process conducted by teams of attorneys. Team members would laboriously read through all the documents, marking them as important or otherwise with tags based on the document's relevance to the request. This method of search is now known as *linear review*. Baron et al. [1] report that the review rates for different topics at the TREC 2006 Legal Track ranged from 12.3 to 67.5 documents per hour, and averaging 24.7. In 2007, the average review rate was around 20 documents per hour, and around 21.5 per hour in 2008 [20]. Roitblat et al. [21] reported that for a large-scale review, 225 attorneys were required to each work nearly 2,000 hours to review 1.6 million documents, at a rate of 14.8 documents per hour. Borden [2] cites a review of "fairly technical" documents running at the rate of 45 documents per hour, and states 50 to 60 documents per hour as the "e-discovery industry average."

The time required for linear review depended on several factors relating to document collection, requests, and reviewer time. The typical review rate was a few minutes per document. As is to be expected, with the growth in size of the collection, such a review process becomes more and more impractical. To remedy this issue a higher degree of automation is required. This automation process is summarized by a widely-cited EDRM Reference Model[2]. Starting from "Information management", where the intent is to incorporate all the information processing that is required by the organization before the e-discovery process starts, to "Presentation" which is the part of the process that prepares depositions, etc.

The LexisNexis experience is that "Document search" represents 60 to 80% of the total cost. Hence, this work focused on improving that search capability.

3 Process Lessons

This section reviews how BigSE was created and operated. For those already familiar with BigData projects, some of the lessons listed in this section will unsurprising (perhaps, even trivial). We present them nevertheless since, if the reader wants to propose something like BigSE to their organization, then their management might find the process lessons of this section most relevant.

3.1 Hardware

Modern Big Data infrastructures make it possible to offer a faster series of conclusions to industrial partners. For example, using BigData tools, NcState was able to maintain the interest of LexisNexis engineers with a continual stream of weekly innovations and other results. BigSE students run their experiments on the NcState High Performance Computing facility (HPC is a network of 10,000 machines with 2 to 8 cores on each). A standard run for us is a 25 times repeat cross-validation experiment where 5 variants of a learner are run over 25 data sets. Depending on the size of the data and the speed of the learner, such a run can take hours to days to terminate on a single machine. However, on HPC, that same run terminate in times as short as 30 minutes. Note that it took a few weeks of script tweaking before we take full advantage of HPC but that was time well spent since it meant we could:

[1]https://www.law.cornell.edu/rules/frcp/rule_26

[2]http://www.edrm.net/resources/edrm-stages-explained

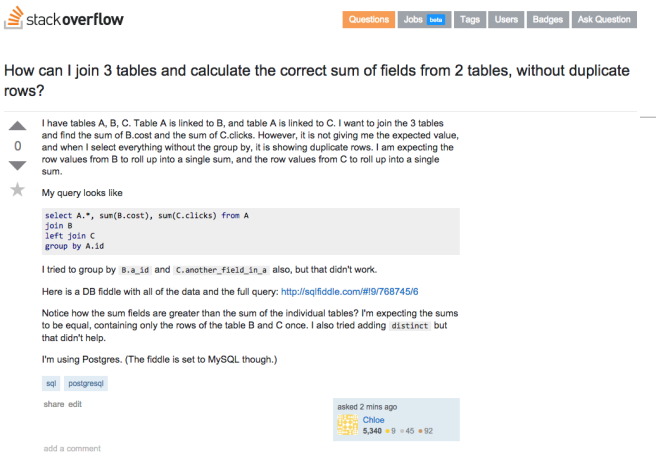


Figure 1: Example Stackoverflow.com data used in this work.

Lesson 1
Use Big Data to generate more conclusions, sooner.

3.2 Domain Sampling Materials

At the start of the project, LexisNexis spent some effort building and presenting training materials to introduce NcState to their business domains.

Those materials were of two forms. Firstly, at the first four weekly meetings, LexisNexis presented slides describing the business domain within which the text mining was to occur. These briefings were important in learning what mattered, and what did not matter, to the industrial client.

Lesson 2
In an industrial/academic relationship, it will take some time for the industrial partner to explain the details of their business to an academic partner.

The second form of materials that was important to this project was a "mock" data generator. This "mock", built by LexisNexis, generated data representative of the kinds of data seen in industrial practice. Given a random number seed, this "mock" pulled data from Stackoverflow.com (e.g. see Figure 1) and built a data set containing the attribute and class distributions seen in current LexisNexis E-Discovery work.

The importance of the "mock" was three-fold. Firstly, it gave NcState ready access to large numbers of non-confidential data sets. Secondly, when we reported results, it gave LexisNexis confidence our methods would work on their kind of data. Thirdly, the LexisNexis "mock" taught NcState about the unique features of the problems faced during E-Discovery. Much of the literature on text mining Stackoverflow.com assumes "tag-level" while the data more relevant to E-Discovery is "site-level", which we characterize as follows:

- **Tag-level**: Each data set contains thousands of posts. The title and body of the posts are concatenated to form the independent variables for classification. The first tag for the

posts is treated as the class of the post. In each data set, one specific tag is selected as the "relevant" class and all other tags are considered as "irrelevant" classes.

- **Site-level**: generated from the LexisNexis "mock", which produces a two-class data set comprising 20000 threads, where "relevant" examples from one specific sub-site were 2 to 6% of the data. The task of site-level prediction is to predict which sub-site a test example belongs to, instead of tag.

The site-level/tag-level distinction is important since much of the text mining literature on Stackoverflow.com is aimed at tag-level tasks [13, 19, 22]. For example, Clayton and Byrne, 2013 [22] have worked on StackOverflow tag prediction and developed an ACT-R inspired Bayesian probabilistic model. This approach achieves a 65% of accuracy by choosing the tag that has the highest log odds of being correct, given the tag's prior log odds of occurrence and adjusting for the log likelihood ratio of the words in the post being associated with the tag. For another example, see also Kuo, 2011 [13]'s work on predicting the next word in a corpus– which is an interesting task but not relevant to E-Discovery.

In the following, we glean on the nature of the data to show that focusing on site-level there are a unique set better choices for text mining this particular kind of data. Hence:

Lesson 3
Text mining methods should be tuned to specifics of the data set used in a particular context.

Lesson 4
After sorting out the CPU farm, the next important task is to build a "mock" generator for data that is representative of the specific target domain.

3.3 Other Management Details

3.3.1 Nondisclosure Agreement

All students and faculty involved in BigSE sign non-disclosure agreements with LexisNexis. The particular NDA used here was not particularly restrictive. Researchers at NcState agreed not to share confidential information gained from LexisNexis as well as being somewhat circumspect in their publications (e.g. showing LexisNexis engineers drafts of any paper and asking for comments and/or corrections). In return, NcState agreed to document all their analysis of publicly available data sets and tools, shared with LexisNexis engineers in private Github repositories. The ease with which these NDAs were developed taught us that:

Lesson 5
Corporate confidentiality is an important concern, but it does not need to stifle industry/research interactions regarding Big Data.

Lesson 6
Use of open source tools and data aids collaborative interactions between industry and academia.

3.3.2 Use of Github

The Github repositories proved useful in several ways. Firstly, LexisNexis engineers have immediate and ready access to any interesting methods and/or findings created by NcState. Secondly, rather than wasting time writing PowerPoint slides, NcState students could report their results using the Markdown tools within the Github issue report systems. Thirdly, using Github, LexisNexis developers could monitor and provide quick feedback on the NcState code and issues, as they arose. Lastly, and perhaps more importantly for building a trusting relationship, LexisNexis could see consistent effort on the part of NcState. SeLAB members are urged to do all their planning and coding and task management in Github. This gives LexisNexis an accurate appreciate for all the work conducted by the graduate students. Of course, the use of Github is only a recommendation, there are several communally used revision system that offer similar benefits and those could work just as well.

Lesson 7
Github tremendously simplifies industry/ academia communication for projects that make extensive use of prototyping.

3.3.3 Staffing and Training

To start-up the lab BigSE, LexisNexis funded several graduate students for one year. Also, LexisNexis engineers gave talks at NcState graduate recruitment days (and to the graduate seminar series). From those talks, several other non-paid graduate students joined the project as part of their research hours subjects. In all, BigSE is now staffed by six graduate Ph.D. students who sit in adjacent cubicles and who share their coding tools and tricks.

Newcomers to BigSE are given analysis tasks that the team has previously completed and asked to reproduce the results or challenge them. For that initial study, they have access to all prior code written by the team and their challenge is to understand the parts and assemble them into a working whole.

LexisNexis and NcState meet each week to present and review the results. That meeting includes a round robin session where each graduate student is asked to show some results, or say "pass". This session imposes some competetive pressure on the students to achieve results in order to "show off" in front of their peers and in front of the LexisNexis engineers. Not every student is expected to have something to show each week, but it becomes abundantly clear within the overall team if some student is "passing" all the time.

While our students have a steep learning curve in their first two months, by month three they are typically generating results that surprise and interest LexisNexis. Hence, with the proviso that newcomers can be immersed in an environment with more experienced people:

Lesson 8
With the right support structures, novice Big Data graduate students can become productive in two to three months.

3.3.4 Schedule and Tasks

As to specific tasks, graduate students are assigned a mix of projects. Some are very short-term (e.g a one week study on the merits of TF*IDF vs term frequency) and some are more medium to long-term that can take weeks to months (e.g. literature reviews on entity recognition or active learning experiments).

Initially, to build trust between LexisNexis and NcState, the focus of the work was on very short term projects (weekly incremental reports). Now, the focus has changed and LexisNexis is prepared to discuss longer-term projects that deliver incremental value, that may yet take months to complete. Hence we recommend:

Lesson 9
Initially, build trust with short-term goals, then expand later.

The above lesson is very important, it holds true for all kinds of collaborations and not just for industrial-academic collaborations.

3.3.5 Preparing for "Critical Audits"

This "portfolio" approach that combines short low-risk tasks with longer high-risk tasks is useful management trick for speculative projects. In commercial endeavors, every so often, projects receive a critical "audit" where engineers have to occasionally prove their worth to a (perhaps) critical audience that is eager to cut expenses. So far, there have been no critical audits of the BigSE project, but it is always best to be prepared. Ensuring a continual supply of conclusions (from the short-term projects), is one way to increase the odds of surviving the critical audit. Hence we advise that:

Lesson 10
It is useful to run a mix of short and long term projects.

4 Technical Lessons

The experiments performed at BigSE are many and varied and constantly changing. So far, we have focused on methods to improve classification, active learning, and entity recognition.

In order to report within the seven page limit of this paper, we focus here on the classification work. Note that the following results are not a claim that the best way to do all text mining is via the methods championed below. Rather, we say that for the specific site-level problem explored by LexisNexis for the E-Discovery work, the following choices have been found to be useful. For other kinds of data, we would recommend further study to find the results that work best for those other kinds of data.

4.1 Performance Metrics

There are two contradictory goals that should be achieved according to our task:

a) Precision (p) since all the documents being predicted as 'related' should be examined by human efforts, a high precision can greatly reduce the cost of this examination.

b) Recall (r) a high recall can reduce the number of 'related' documents that we failed to retrieve.

For the purpose of studying both precision and recall, we use: $F_1 = 2pr/(p+r)$ as our performance metric.

4.2 Standard Experiment

Starting with LexisNexis' default settings to their text miners (stemming and stop words removal for tokenizer, term frequency for featurization, hashing trick for dimensionality reduction, L2 normalization on rows for normalization, linear SVM for classifier), we modified one option at a time (e.g. swapped the hashing trick for Tf*idf). Different options in a single process (e.g. hashing trick and TF-IDF selection in dimensionality reduction) were compared in each experiment with 5 by 5 cross-validation (20% as training sample and 80% as testing sample since in real tasks, the training sample is always less). This rig was applied to 10 tag-level problems and 15 site-level problems (using an SVM with a linear kernel as the classifier).

4.3 Comment on External Validity

There are several validity threats to the design of this study:

a) The above experiments show results from a limited number of text mining methods. The space of known text mining methods is truly vast and the following studies represent just a small portion of the total space.

b) All the above used data that we have characterized as tag-level or site-level samples from posts in Stackoverflow.com. For other kinds of data, we recommend repeating this analysis to find the right text mining methods for that kind of data.

Due to these threats to external validity, we make no presumption that the following conclusions work best for all text mining applications. We only assert that the following work better than some alternatives in our domain.

That said, we assert that the following is externally valid:

Lesson 11
Commercial organizations can outsource at least some of their their validation work to university partners.

Further, as part of that validation process:

Lesson 12
The university partners can sometimes find better choices that those currently employed by the commercial organization (e.g. see the SMOTE results, below).

The rest of this section offers support for these two lessons.

4.4 Results

The results are presented in Figure 2. Three interesting results in those plots are listed below.

1. Performance deltas: In Figure 2, the tag-level problems are shown on the left and the site-level problems are shown on the right. The results show a low valley on the left and higher plateau on the right; i.e. methods that work well for site-level do not work well for tag-level. This result underlines Lesson 3&4 on the importance of testing via data that is particular to specific domains.

2. Low IQR: In Figure 2, the solid lines at the top show the median (50th percent) value and the dotted lines at bottom show the IQR (the inter-quartile range calculated from the (75th-25th) percentile). Note that, usually, the dotted lines are close to zero; i.e. the variance in our conclusions across multiple samples is mostly very low. Hence:

Lesson 13
The performance variance is small enough to make definitive statements about what methods work best for site-level problems.

3. Blurred results: It is difficult to distinguish between some the results since, looking at Figure 2 (a) and (b) and (c), the overall benefit of certain treatments is marginal. That is:

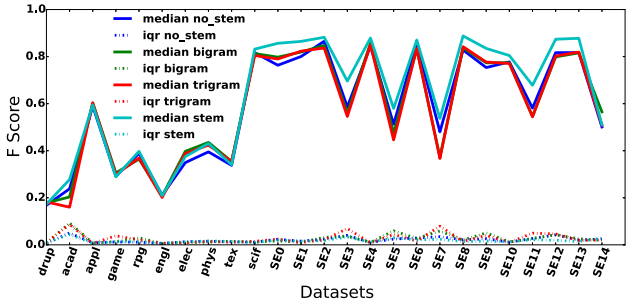
Lesson 14
Not all standard text mining methods are powerful or useful.

The rest of this section discusses more specific results.

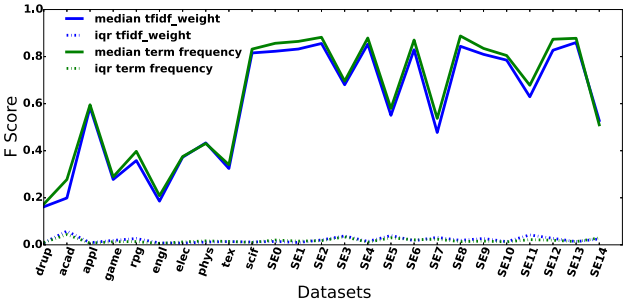
4.4.1 Tokenization

Tokenization is the very first step in natural language processing. It removes the spirit of symbols by identifying the most basic units which need not to be decomposed in the subsequent processing [25]. **Shingling** introduces phrases as unit token. A word-based w-grams shingling captures contiguous sequences of words as unit tokens and each unit token has w words [5]. Bigram and trigram shingles uses w=2 and w=3-grams, respectively. **Stemming and stop words removal** combines words that have same meaning and removes very frequently used words. An example of stemming and stop words removal would be to convert "i want to eat apple since i like eating apples" to "want eat apple since like eat apple". While stemming and stop words are considered an effective way to improve the text classification performance [27], some contradictory evidences deprecates these techniques [19, 22].

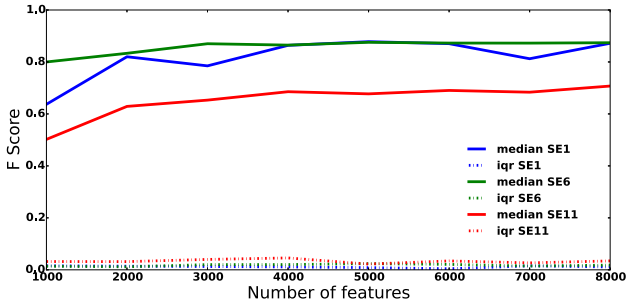
The cost of these tokenizers is very low (each can be completed in a simple pass through the data) but their observed benefit is



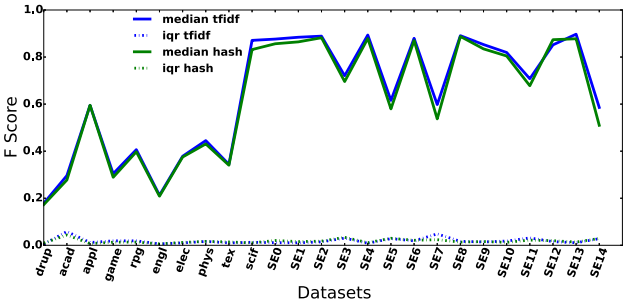
(a) Tokenization



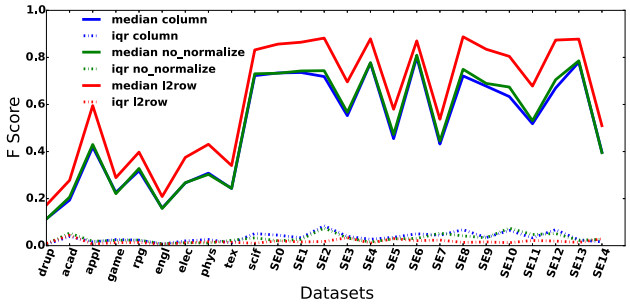
(b) Featurization



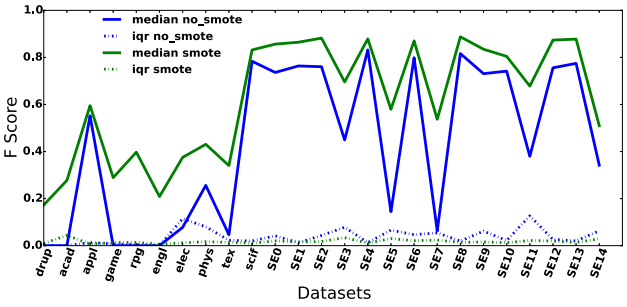
(c) Number of Features



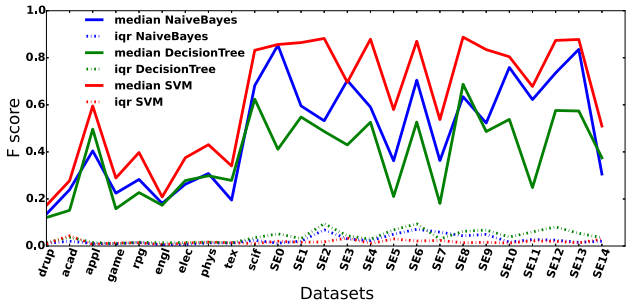
(d) Dimensionality Reduction



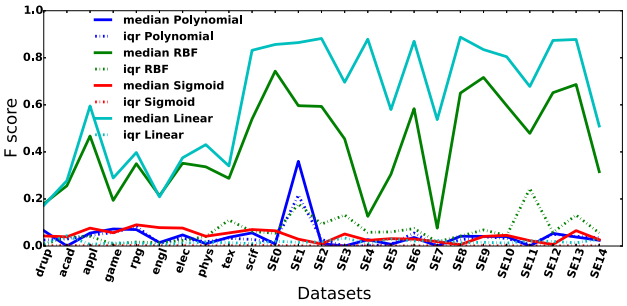
(e) Normalization



(f) Data Balancing



(g) Classifiers



(h) SVM Kernels

Figure 2: Results seen in multiple 5*5 cross-validation experiments. Unless otherwise specified, the learner here is an SVM with a linear kernel. In all these plots, *higher* lines mean *better* performance. Within each plot, the solid lines (at top) are median scores and the dashed lines (at bottom) are inter-quartile ranges (75th-25th percentile). Also, except for Figure 2(c) ten left-hand-side data sets are tag-level while the remaining are site-level data sets generated from our "mock" data generator.

small. As shown in Figure 2(a), the average improvement of stemming and stop words removal in median value is 1% on tag-level data and 6% on site-level data sets. Note that deltas are very close to the IQRs of 3%,4% for tag-level,site-level (respectively). Hence:

Lesson 15
Despite their prominence in the literature, the benefit of stemming and stop words and shingling for our data sets is minimal.

4.4.2 Featurization

Featurization is the process which transforms the tokens of each document into a vector of weights of each unique token, which can be used to train the classifier [15]. **Term frequency**, also known as word count [15], would convert "want eat apple since like eat apple" to *{eat: 2, apple: 2, want: 1, since: 1, like: 1}*. **Tf-idf weight** calculates not only the word count, but also the number of documents each token appears in , calculated as follows:

$$tfidf(t,d) = |t \in d| \cdot log\left(\frac{|D|}{|d \in D : t \in d|}\right) \quad (1)$$

where D, T are all documents and tokens and $t \in T, d \in D$. The intuition behind Tf-idf is that the more frequent a token appears and the less documents it appears in, the more important it is. Tf-idf is a popular feature for text categorization [4, 19].

Term frequency performs no worse than tf-idf, Figure 2(b). Hence:

Lesson 16
*Despite its prominence in the literature, the added value of Tf*idf for our data sets is minimal.*

4.4.3 Dimensionality Reduction

Dealing with the tables of data containing all the words in English, would mean processing a very large number of columns. A useful method for removing noise and reducing the memory costs of learning is dimensionality reduction.

Two methods for finding those weights are the *hashing trick* and *Td-idf*. **Feature selection by tf-idf score** sorts tokens by their tf-idf score (see Equation 1) then and picks the N highest-scored tokens. The **Hashing trick** is a single-pass method that turns features into entries of a matrix by applying a hash function to the features. The number of columns of the output matrix can be pre-defined and thus dimensionality reduction can be done by applying hashing trick [26]. For example, the hashing trick might convert {eat: 2, apple: 2, want: 1, since: 1, like: 1} into *{3, 3, 1}*.

Figure 2(c) shows a sample of our results of using Td*idf to select for an increasing number of tokens (due to space reasons, not all the results are shown). In all our data sets, no additional benefit was seen after using 4000 terms.

As shown in Figure 2(d), compares the results of the hashing trick with Tf*idf selecting for 4000 words. The differences of these two methods in median value is no bigger than the IQRs. Hence:

Lesson 17
When reducing memory costs of storing our kind of text mining data in RAM, a simple hashing trick will suffice,

4.4.4 Normalization

Normalization is a process which adjust the values measured on different scales to a notionally common scale.

Normalization on columns transforms a value x in a column to $(x - min)/(max - min)$ (where *min*, *max* come from that column). In general data mining tasks, the weight of different features are measured on different scales. Normalization on columns of the feature matrix eliminates these differences among features.

L2 Normalization on rows takes feature vector of each document and divides it by its L2 norm. It is the standard normalization method for text categorization [8]. The main purpose of implementing L2 normalization on rows is to rescale the vector of feature weights to unit length. For example, if results of applying the hashing trick are *{3, 3, 1}*, then L2-normalization would divide each number by $\sqrt{(3^2 + 3^2 + 1^2)}$ to produce $\{0.69, 0.69, 0.23\}$.

As shown in Figure 2(e), column normalization offers little to no improvement over no normalization. However, L2 Normalization on rows can add up to 20% of the F_1 score.

Lesson 18
Normalizing rows can be more useful than normalizing columns.

4.4.5 Data Balancing

In imbalanced data sets, the target class are a small minority within the data. The LexisNexis data is highly imbalanced with the target class may be as low as 2% of the data. Yet prior to BigSE, LexisNexis was not exploring data balancing methods.

Figure 2(f) shows experiments with applying the **SMOTE** data balancer to our data. SMOTE over-samples the minority class by introducing synthetic examples along the line segments joining any pair of nearby real samples in neighborhood. Proposed in 2002 [6], SMOTE is a very effective over-sampling method and has been widely applied during the past decades [3, 9, 14].

As shown in Figure 2(f), the average improvement of SMOTE in median value is 22% on tag-level data and 17% of site-level data. Note that, in these SMOTE experiments, we SMOTEd the training set but the distributions in test set were left "as is", Also, the average IQR for that plot is 4%,9% on tag-level,site-level data; i.e. is much smaller than the median improvement. Hence we say:

Lesson 19
Class balancing (e.g. with SMOTE) is very useful for this kind of text mining.

4.4.6 Classification

Classifier guesses new labels using models built from prior data. **SVM** is a widely-used learning model for many data mining tasks [10, 19]. SVM's use some *kernel* that maps the raw data into another space where the data might be more separable. Linear SVM is of the simplest kernel function but others include polynomial, radial bias, sigmoid, etc.

There are many other kinds of classifiers. **Naive Bayes** represents a family of simple probabilistic classifiers which has been studied extensively since 1950s. Multinomial Naive Bayes is especially designed for text categorization [16] and always considered as a baseline method for text categorization.

Decision Tree learners are a popular model for classification. CART is one implementation of decision tree learners that has been proved useful in prior text mining applications [18].

Figure 2(g) compares linear SVM with other kinds of classifiers. The results from that figure are very clear: linear SVM outperforms Naive Byes and decision tree learning (with CART).

Figure 2(h) studies the effect of different kernel functions in SVM. Note that linear SVM is the clear winner.

Lesson 20
Linear SVM is the recommended classifier for site-level data.

5 Conclusions

This paper has listed the lessons learned from one the BigSE LexisNexis/NcState partnership. This lab has found that. most of the operator choices made by LexisNexis are demonstrably better than many other choices. However, in some cases, NcState found certain operators much faster than others (e.g. required only a single pass of the data). Also, in one case, NcState showed that one operator (SMOTE) not currently used by LexisNexis offered useful improvements in their text miners. SMOTE is now scheduled for inclusion in the next release of the LexisNexis tools.

More specifically, for the site-level problem that is relevant to LexisNexis' E-Discovery problem, BigSE recommends:

1. **Tokenization:** Stemming and removal of stop words were generally useful. Shingling offered no noticeable benefits.
2. **Featurization:** Contrary to popular recommendation, term frequency (TF) proved to work just as well as using TF-IDF.
3. **Normalization:** Use L2 normalization on the rows.
4. **Dimensionality Reduction:** Given how there is so little difference between hashing trick and TF-IDF, using the hashing trick in place of TF-IDF offered an added benefit of needing fewer dimensions (and less memory) and taking less time (TF-IDF needs two passes of the data); the hashing trick can be applied on each row, in a single pass of the data).
5. **Data Balancing:** The low prevalence of "interesting" class in the test collection was handled well by SMOTE.
6. **Classification:** It was surprising to note that a simple Linear SVM outperformed other kernels.

As mentioned above, we make no presumption that the above list of choices works best for all text mining applications (but it is a set of choices that work better than others, in our domain). As for other domains, we would recommend that organizations assemble their own validation team to find the best choices in those domains.

The LexisNexis/NcState collaboration has been useful for both parties. LexisNexis got to explore more text mining operators while NcState got to expose their research students to the realities of real-world industrial data mining. Also, BigSE has proved to be a pathway from research to employment (e.g. BigSE students will work as summer interns at LexisNexis). Based on these results, LexisNexis management has extended the collaboration till the end of 2016 (and further work in 2017 and beyond is begin discussed).

As to future work, as mentioned above, these studies have explored only a small portion of the very large space of operators available in text mining. Our next phase is to explore automatic tuning methods that can explore that large space. In order to explore a larger decision space as well as avoiding local optimum, heuristic search algorithms such as differential evolution, GALE, and NSGA-II will be applied to tune the decisions for a better approach [7, 12, 23]. We are working on this to achieve both less number of comparisons and better solutions.

Another useful future direction is active learning. One constant challenge in the LexisNexis work is gaining access to labelled data. In theory, active learning could reduce the cost of humans reading and classifying documents during E-Discovery. For example instead of 20% of the data, only a thousand samples will be selected and then used for training. The performance of prediction will rely heavily on how representative the selected samples are, and active learning is the key method to achieve it [24].

6 References

[1] J. R. Baron, D. D. Lewis, and D. W. Oard. Trec 2006 legal track overview. In *TREC*, 2006.

[2] B. B. Borden. The demise of linear review. In *Williams Mullen E-Discovery Alert.*, 2010.

[3] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap. Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In *Adv. in Knowledge Discovery and Data Mining*. Springer, 2009.

[4] M. F. Caropreso, S. Matwin, and F. Sebastiani. A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization. *Text databases and document management: Theory and practice*, pages 78–102, 2001.

[5] M. Chang and C. K. Poon. Using phrases as features in email classification. *Journal of Systems and Software*, 82:1036–1045, 2009.

[6] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, pages 321–357, 2002.

[7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.

[8] E. Frank and R. R. Bouckaert. Naive bayes for text classification with unbalanced classes. In *Knowledge Discovery in Databases: PKDD 2006*, pages 503–510. Springer, 2006.

[9] H. Han, W.-Y. Wang, and B.-H. Mao. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *Advances in intelligent computing*, pages 878–887. Springer, 2005.

[10] T. Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM, 2006.

[11] F. P. B. Jr. *The mythical man-month - essays on software engineering (2. ed.)*. Addison-Wesley, 1995.

[12] J. Krall, T. Menzies, and M. Davies. Gale: Geometric active learning for search-based software engineering. *IEEE Transactions on Software Engineering*, pages to–appear, 2015.

[13] D. Kuo. On word prediction methods. Technical report, Technical report, EECS Department, University of California, Berkeley, 2011.

[14] J. Luengo, A. Fernández, S. García, and F. Herrera. Addressing data complexity for imbalanced data sets: analysis of smote-based oversampling and evolutionary undersampling. *Soft Computing*, 15(10):1909–1936, 2011.

[15] C. D. Manning and H. Schütze. *Foundations of statistical natural language processing*. MIT press, 1999.

[16] A. McCallum, K. Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998.

[17] T. Menzies, E. Kocaguneli, B. Turhan, L. Minku, and F. Peters. *Sharing Data and Models in Software Engineering: Sharing Data and Models*. Morgan Kaufmann, 2014.

[18] O. Miotto, T. W. Tan, and V. Brusic. Supporting the curation of biological databases with reusable text mining. *Genome informatics*, 16(2):32–44, 2005.

[19] M. K. Moharana. Tag recommender for stackoverflow questions.

[20] D. W. Oard, B. Hedin, S. Tomlinson, and J. R. Baron. Overview of the trec 2008 legal track. Technical report, DTIC Document, 2008.

[21] H. L. Roitblat, A. Kershaw, and P. Oot. Document categorization in legal electronic discovery: computer classification vs. manual review. *Journal of the American Society for Information Science and Technology*, 61(1):70–80, 2010.

[22] C. Stanley and M. D. Byrne. Predicting tags for stackoverflow posts. In *Proceedings of ICCM*, volume 2013, 2013.

[23] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[24] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66, 2002.

[25] J. J. Webster and C. Kit. Tokenization as the initial phase in nlp. In *Proceedings of the 14th conference on Computational linguistics-Volume 4*, pages 1106–1110. Association for Computational Linguistics, 1992.

[26] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1113–1120. ACM, 2009.

[27] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *ICML*, volume 97, pages 412–420, 1997.