

Integrating Data and Text Mining Processes for Digital Library Applications

Robert Sanderson
University of Liverpool
Department of Computer Science
Ashton Street, Liverpool, L69 3GL
(+44) 151 795 4252
azaroth@liverpool.ac.uk

Paul Watry
University of Liverpool
Sydney Jones Library
Chatham Street, Liverpool, L69 3DA
(+44) 151 794 2696
pwatry@liverpool.ac.uk

ABSTRACT

This paper explores the integration of text mining and data mining techniques, digital library systems, and computational and data grid technologies with the objective of developing an online classification service exemplar. We discuss the current research issues relating to the use of data mining algorithms and toolkits for textual data; the necessary changes within the Cheshire3 Information Framework to accommodate analysis workflows; the outcomes of a demonstrator based on the National Library of Medicine's Medline dataset; and the provision of comparable metrics for evaluation purposes. The prototype has resulted in extremely accurate online classification services and offers a novel method of supporting text mining and data mining within a highly scaled computational environment, integrated seamlessly into the digital library architecture.

Categories and Subject Descriptors

H.3.7 [Information Storage and Retrieval]: Digital Libraries – Collection, Systems issues. I.2.7 [Artificial Intelligence]: Natural Language Processing.

General Terms

Algorithms, Performance, Design.

Keywords

Data Mining, Text Mining, Digital Libraries, Grid Processing.

1. INTRODUCTION

Recent years have shown increasing interest in the re-application of methodologies and ideas from related areas of information science and eScience within the digital library and digital preservation contexts. The continued advancement of data and text mining techniques has produced useful and accessible tools with which to implement or improve digital library services, including advanced document clustering and automated metadata extraction as a means to improve discovery. The natural language processing techniques used in text mining (following Hearst's definition[6]) may also be applied to text and data sources in large

scale digital libraries and repositories. On the eScience side, massively parallel computing and data grids are also being incorporated into mainstream digital library solution: examples include the integration[14] of the DSpace curation management system with the Storage Resource Broker; and the integration of the Cheshire3 Information Framework with the Storage Resource Broker as part of the National Archives and Records Administration digital preservation prototype[8].

The three areas of data and text mining, the grid, and digital libraries have inter-related benefits and challenges. The scale of digital libraries has continued to grow and is now at the point where data grid-based storage solutions are becoming of great relevance. Data grids are able to provide seamless access to petabytes of storage, geographically distributed to different locations, and stored in different types of repository. Files are identified by a logical identifier, rather than a file name; a system very familiar to digital libraries in terms of handles, DOIs and so forth.

Given this increased amount of available data and metadata, machine learning processes, supported by integration with digital library systems, are able to be more accurately trained, reducing manual labor and adding value to previous investment in content creation. Data mining processes typically include classification (predict if a given document is a member of a particular class or domain), clustering (grouping together similar documents) and association rule mining (discovering rules that interrelate documents or terms within those documents). However, both data and text mining processes are computationally expensive and can benefit in turn from distribution across multiple machines for parallel computation.

Exploratory prototypes have been developed to examine the convergence of information access systems with tools for data analysis. These advances point to further work required to determine the extent to which, if at all, natural language processing can aid in the accuracy and efficiency of data mining being performed on a large collection of texts. In particular, can classification engines be generated that are more accurate, faster to classify or faster to train, handle more data at once, or all of the above?

This paper begins to explore some of the synergy between these overlapping areas, focusing on the text and data mining aspects. We present our research into this integration and the developments required within the Cheshire3 Information Framework to support it, using a prototype service developed for the National Centre for Text Mining (NaCTeM) in the UK to

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

JCDL '07, June 18–23, 2007, Vancouver, British Columbia, Canada.

Copyright 2007 ACM 978-1-59593-644-8/07/0006...\$5.00.

evaluate the success of the work. This application focuses on the text and data mining integration work within a digital library rather than simply using the data stored in it. The scale of the full Medline[9] data set, currently some 16 million abstracts, combined with the computational expense of both performing natural language processing and then training machine learning tools is an ideal scenario for also investigating the use of distributed processing.

First, background and related work is considered, showing that individually the components have been well researched by others, but the full integration of all three areas has yet to be developed to maximize the potential value. We then turn to the methodology of how this integration was performed within the Cheshire3 architecture. The experiments and results of the NaCTeM prototype are then discussed to demonstrate the correctness, scalability and utility of the integration work and finally conclusions are drawn.

2. BACKGROUND AND RELATED WORK

The computational grid, although not yet the ubiquitous resource envisioned by the NSF's Office of Cyberinfrastructure, has been described in detail in the past and is becoming increasingly important for digital libraries and information retrieval as for all aspects of the information technology sector. As a data intensive application, digital libraries require some dedicated infrastructure to make efficient use of the available processing resources. The requirements for such an infrastructure are currently being investigated by several projects including DILIGENT[3] and Cheshire, but it is also moving into mainstream industry with Oracle discussing their own digital library solutions built on grid technology at EGEE 06[10], for example.

Data grid solutions for massive storage have also made their way into the core of digital library and digital preservation thinking, with the Storage Resource Broker (SRB)[11] from the San Diego Supercomputer Center in particular having been discussed in international digital library proceedings in the past.

However, data mining and text mining applications within the digital library context have been perhaps more specialized to date. Machine learning technologies such as Hidden Markov Models and Support Vector Machines have been used for name disambiguation[5], metadata enhancement and to analyze service generated metadata such as query logs and click streams, however full advantage of the general techniques within digital library services does not appear to be as widespread as one might expect, given their utility. The major causes for this are, in the eyes of the authors:

1. As digital libraries typically contain primarily textual documents, only algorithms suited for text processing are of interest. Many data mining algorithms balk at the high dimensionality presented by a full document/term matrix.
2. Data mining toolkits often do not account for sparse input vectors; instead they require an entry for every attribute, even if the available algorithms would accept sparse input. When each term is an attribute (and the frequency within the document the value for each record), this leads to

an input data set which is too large to fit into main memory for any reasonably sized collection.

3. Data mining processes are computationally very expensive and technically challenging to implement, and hence only solutions to specific problems tend to be investigated. The co-integration of massively parallel processing can help alleviate this factor.

Text mining processes within digital libraries are even less well established, and for the most part simply use the digital library infrastructure as a convenient location to discover texts to process, rather than being fully integrated within services and workflows. The "marrying" of the Greenstone digital library with the GATE text mining environment[15] provides some initial experimentation in this area, and considering the co-development of the WEKA data mining toolkit, one might expect to see a fully integrated solution available in time.

The computational expense to execute text mining based analysis, point three above for data mining also, is believed to be the major cause for the lack of widespread use of text mining processes. There are only a very few natural language processing systems available for general use that are fast enough to cope with even medium scale digital libraries without parallel processing, and the most advantageous method of integration into document processing workflows is often not obvious.

Therefore, it is proposed that while the computational and data grid integration is very important for dramatically increasing the scalability of such systems within current models for digital libraries, the real utility comes in the added integration of computationally expensive processing such as data and text mining applications embedded within the architecture for seamlessly extending the information available for discovery and analysis. When all of these subsystems are available in one architecture, the disadvantages normally encountered (large computational requirements, large storage requirements) are taken care of, which has not been addressed by previous work in this area.

In the next section we will describe the text and data mining algorithms and implementations selected for this initial integration research, and how they were integrated into the Cheshire3 Information Framework.

3. METHODOLOGY

3.1 Algorithms

As outlined in points 1 and 2 above, not all algorithms and toolkits are equally suited for dealing with textual data, often represented as vectors for these purposes. Clustering and Association Rule Mining (ARM) tend to deal more easily with this sort of data and document clustering is well understood, so we have focused initially on the classification process. Four algorithms of varying degrees of sophistication and computational requirements were selected for this first round of integration.

Fast Domain Finder is a TF/IDF based classification technique developed internally at Liverpool. As the name suggests, it does not require much computation and it can also be used successfully with only a small number of training instances, including when determining documents belong to multiple

domains. However it is only suitable for predicting the domain(s) of a document from the set of classes it was trained on and is not a general purpose classification algorithm.

For integration within the Cheshire3 architecture, FDF was reimplemented from Java using native Cheshire3 processing objects. This allows for the execution to be seamlessly distributed between multiple machines.

Naïve Bayes is a probabilistic algorithm which is capable of learning many classes simultaneously and can be trained incrementally. It works best when there is a significant amount of training data in order to fine tune the probabilities. When the evidence suggests more than one possible class it can return the calculated probabilities for each, which is important in scenarios, for example, where an instance may have more than one valid domain, similar to FDF above.

The Reverend library[4] was selected for this integration as a pure python implementation that was easy to use and extend, while still being reasonably fast. It does not support Bayesian Networks, however applications that would use such a network would likely catered for by a Support Vector Machine based classifier. Reverend supports merging of trained models, which is important for distributed processing as each machine in the compute cluster can learn a subset of the model and once all subsets have been finished, they need only be merged by one machine into the complete model.

Support Vector Machines are a very accurate classification technique and, once trained, are very fast to make predictions. They are very good at supporting sparse data, and some implementations can return probability information for the predictions. On the flip side, SVM is a binary classifier – it can only distinguish between two classes at once. It is possible to train n-1 SVMs to predict between n classes, where each predicts one class or not that class, however this is very slow for many classes.

LibSVM[1] from the University of Taiwan proved to be a very capable and fast implementation for integration. It includes command line tools, instructions for use as a library within C programs, and a Python wrapper library allowing it to be used directly from within Cheshire3 rather than via a command line shell pipe.

Classification Association Rule Mining (CARM) is a constrained use of ARM, where the consequent of the generated rules is a single class label attribute. The procedure is otherwise the same as general ARM. CARM is capable of learning n classes natively and supports sparse input, but perhaps the most important distinction to make in comparison to the other three algorithms above is that it the classification model generated is human-understandable. The rules used to make the prediction can be inspected to see why the given class was selected, unlike the more mathematically oriented SVM, FDF and Bayesian classifiers. Most CARM implementations do not allow for incremental training, however.

The integrated CARM implementation comes from the Department of Computer Science in Liverpool. TFPC[2] is an a-priori based algorithm which makes use of novel and fast tree structures to efficiently record the data from which the rules are derived. Investigation into incremental uses of these structures is currently being undertaken. Unlike the previous implementations,

TFPC is only available in Java at the current time so integration is via external calls rather than natively within Python.

Natural language processing tools are the cornerstone of text mining. In order to treat text as language rather than data, it is essential for the machine to know as much as possible about the words being processed. There are several key techniques which have been integrated into Cheshire3: part of speech tagging, phrase chunking, and deep parsing of the grammatical structures. The tools integrated for these processes are all from the Tsujii Laboratory at the University of Tokyo[13], through their association with NaCTeM. As they are written in C++, the integration is via input/output pipes to the command line utilities.

The integration of grid processing and storage in Cheshire3 has been described in previous papers; for further details and general descriptions of the Cheshire3 framework, interested parties are referred in particular to [7], and [12].

3.2 Digital Library Architecture

A few changes to the Cheshire3 architecture were required in order to properly integrate the tools described above. It would have been possible to simply bolt them on as special cases, however the aim was seamless integration without any preconceptions or restrictions as to how they would be used. They should fit as cleanly as possible within existing object interactions and not require additional layers of code between those interactions.

To summarize the architecture described in the papers referenced above, Cheshire3 has 4 main types of object – data (a representation of a document or dataset; gray rectangles), storage (a layer over a database or persistent storage mechanism; gray cylinders), processing (an object that implements some transformation of a data object; gray ovals), and abstract (a collection of other objects and associated metadata or attributes; white rectangles). Figure 1 presents the most common interactions between the different types of objects, but a short example of a typical data ingest process will hopefully be enlightening. The DocumentFactory is given the path to a directory containing PDF files, which it sequentially loads and makes available to the system. The PDF documents are then turned into XML through a series of PreParsers, and the originals are stored in a DocumentStore. A Parser then parses the XML and the resulting Record is stored in a RecordStore for later retrieval, before being given to a series of Index objects. Each Index then extracts the data using XPath, and uses a series of Normalisers to transform each term. The terms and record pointers are stored in an IndexStore to enable discovery.

Many of the mappings for the text and data mining applications within the architecture are very straight forward. The NLP tools found natural homes as PreParsers and also Normalisers when it was useful to work with only a section of text extracted during indexing rather than the entire document. The annotation of the text can be done either in the common inline fashion, where the part of speech is appended to each term and separated by a '/' character, or by adding additional attributes to the object representing the term within the framework.

Classification was modeled as a multi-stage PreParser – it first takes in a document representing the collection of training instances, processes it and returns the model created as a second

document. This model can be stored and then loaded again at run time via the configuration of the PreParser. The trained PreParser then takes in individual documents which it annotates with the predicted class as additional metadata. This split usage did not require any substantial changes to the architecture, just an awareness within the workflows that the learning phase was needed if the model had not been pre-built.

It was the very first object used in most Cheshire3 ingestion workflows that required the most extensive changes. Previously, a collection of documents (for example a set of MARC or XML files) was modeled as a data object called a DocumentGroup. DocumentGroups had to be constructed outside of the architecture as they were the main input into the system; they could not be built by a processing object unlike other data objects. This deficiency was recognized early on, and DocumentGroups were replaced by a processing object called a DocumentFactory. DocumentFactories fill exactly the same role, however being processing objects they can be included in workflow chains. Another change which this brought about is that DocumentFactories could be configured before runtime in a configuration file. This means that pre-configured applications could set the default values that control the discovery of the individual documents (such as the top level XML tag, or the character encoding) once in the configuration, rather than requiring them to be given every invocation. The most important effect of the change, however, is that DocumentFactories could be invoked remotely in parallel processing mode. This solved many troublesome inconsistencies, such as when it is desirable to create a subset of documents without passing the dataset around between nodes, as this can now be done by a call to a DocumentFactory instead.

While an important change since previous presentations of the architecture, it was not quite sufficient to fully integrate the data

mining processes. A subclass of the DocumentFactory was also required: an AccumulatingDocumentFactory. Normal DocumentFactories identify their collection of documents from a single call, and the documents are then instantly available to iterate through. This does not work for the architecture for classification described above, as each document, rendered appropriately for the classifier, must be loaded individually, whereas the classifier requires a single document with all of the information available together.

The AccumulatingDocumentFactory makes this possible. Instead of making the documents available instantly, it continues to accumulate information until a document is requested. It is only at the point of request that the output document(s) are created, thus allowing all of the term vectors to be merged together into the single document/term matrix required for training.

4. EXPERIMENTS

The National Centre for Text Mining in the UK is researching ways in which to make text mining oriented services available, both for use centrally and for integration into external workflows. The particular work that concerns this paper is a prototype for a classification service creation service; that is to say a service which can be configured to create other online classification services. A user of the service will select two or more discriminating topics for the system to build a classification model from, based on matching documents, and then both enable evaluation against further unseen documents as well as process documents uploaded by the user.

The ingestion process is the critical aspect, with respect to the integrated architecture. Each document is natively in XML including metadata, controlled access fields such as the MeSH headings, the title and abstract. The text mining tools, however,

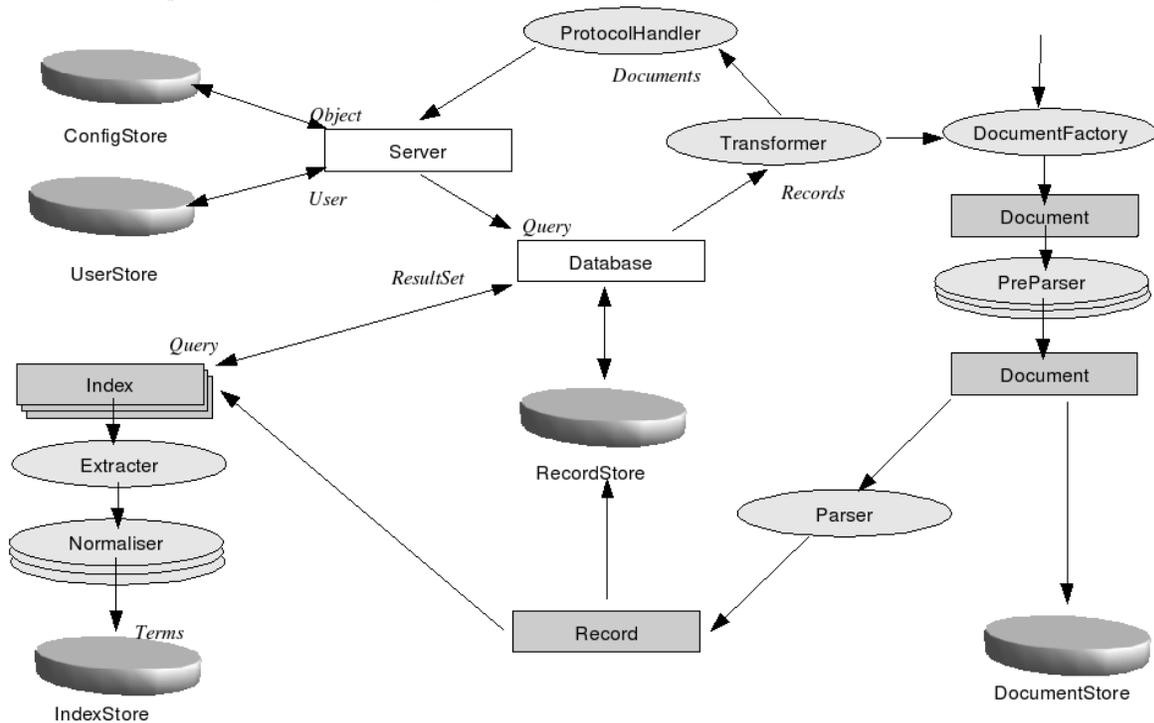


Figure 1. Cheshire3 Object Model

expect raw text, separated into individual sentences. Secondly, in order to properly evaluate the different combinations of tools and preprocessing, it is important to deal with the text as a term, rather than annotating the record directly, to allow different tools to create different indexes. As such, the appropriate modeling was

to use the NLP tools as Normalisers within the Cheshire3 framework rather than PreParsers.

Another important aspect of the ingestion workflow is the ability to store extracted data as a Document. This means that the results of computationally expensive processing phases can be stored such that if, in the future, different post-processing is required or desirable, the workflow can be restarted from the point after the linguistic analysis. A DocumentFactory in the workflow takes care of this aspect, and can store the results in the data grid rather than locally, which will be important for when the service is built for all 16 million records as this annotated data is much larger than the original abstract.

The workflow uses two different part of speech taggers – the basic 'tagger' and Genia, which also returns the linguistic stem of the word along with the part of speech. This allows the comparison at the same time of the standard porter stemming algorithm with the term set produced by full linguistic analysis of the root words. Filters can then be applied based on the results of these tools. Indexes containing only verbs, nouns and adjectives were created rather than using the more typical stoplist based approach to omit unhelpful words.

After the terms have been combined in an inverted index and assigned sequential term identifiers, one vector file per index is created. These vectors will then be used in the model creation stage to build one classifier each, allowing the utility of each term preprocessing workflow to be analysed based on the accuracy of the resulting classifier. Summary metadata concerning global word frequencies is also generated at this time.

The prototype service built on top of this information combines aspects of information retrieval, data mining and text mining together to create an application that is only feasible in an architecture with all three aspects fully integrated. IR techniques are used to identify the most relevant documents to a particular subject, given as a MeSH heading. The system first identifies all of the headings below the given one in the hierarchy, and finds all Records which have any of those headings. The term vectors for the abstract are then used by an AccumulatingDocumentFactory to create a Document representing the matrix for the selected records. This matrix document is then ingested by an SVM PreParser which creates a model based on the documents and is saved for future use.

During the document selection process, every tenth document is extracted to a test set, implemented as a second AccumulatingDocumentFactory. Once the model has been generated, it is possible to then check its accuracy by running against these known Documents. It can thus be viewed as a single fold of ten-fold cross validation. After the application has been migrated to a service, it will also accept uploaded documents of unknown class to have its terms extracted and processed by the same natural language processing workflow, and then classify it against the selected model.

5. RESULTS

The full integration of the various tools is able to demonstrate its utility quite easily. To consider the IR aspects first, the linguistic analysis of the stem has advantages over Porter, if computational expense is not a factor such as when massively distributed processing is available. Not only are the edge cases where Porter generates a strange or inaccurate stem negated, the resulting term list can be displayed more easily to a user for keyword browsing. The more accurate stems will, intuitively, slightly improve retrieval, but means that queries also need to be stemmed using the same NLP based algorithms, which takes additional processing at runtime.

The ability to distinguish parts of speech is also intuitively important for determining relevance ranking, as there are many words with the same orthography but very different meanings. By calculating the relevance score on only the appropriate part of speech for the query context, higher precision will naturally be obtained. For example, 'ram' (male sheep) and 'ram' (to run into forcefully) will likely be used in very different documents. This has not been quantified yet, but will be in future work against the TREC datasets.

By filtering on parts of speech (noun, verb and adjective only, for example), the constructed indexes are significantly smaller than the equivalent indexes created with a static stoplist, without losing any accuracy. Additionally, and more importantly, it solves the problem of words used as proper nouns which are normally stoplisted. For example, 'The Who' would be tagged as a proper noun phrase rather than discarded as an article and a 'wh' word. This allows the server to correctly answer such a query without the need to maintain the proximity based locations of every occurrence of very common words.

Finally, it is possible to create multiword term indexes from extracted noun or verb phrases only, rather than a simple N-gram approach. Again this results in a more useful, smaller index that allows for common adjacency queries to be answered with a single look up without storing all combinations of words.

The vectors produced from the processed terms are then fed as input to the classification components. This was found to improve both the performance and speed of the classification. Using a 10 class text classification problem from the dataset and SVM learners, the following table summarizes the results from using different vector sources.

Table 1. Vector Source and Accuracy Results

<i>Vector Source</i>	<i>Avg Terms</i>	<i>TCV Accuracy</i>
Every word in document	99	85.7%
Porter Stem of words in document	95	86.2%
Part of Speech filtered words	69	85.2%
Porter Stem of filtered words	65	86.3%
Genia filtered words	68	85.5%
Genia Stem filtered words	64	87.2%

Using the linguistic stems, filtered for only nouns, verbs and adjectives the average number of attributes per instance was reduced from 99 down to 64, while at the same time the ten-fold cross validation accuracy improved from 85.7% up to 87.2%. Also notable is that while the filtering reduced the average number of entries in each vector, it also required stemming in order to also increase the accuracy of the system at the same time. The magnitude of the improvement is low as the SVM algorithm is very good at ignoring irrelevant attributes or instances. This is not to say that the improvement should be brushed over. Quite the opposite, as a 2% improvement in accuracy while significantly reducing the number of terms before the SVM learning phase as it shows that the text mining based filtering does a better job at selecting terms for classification than the data mining process does. In conjunction with other techniques available for improving the accuracy of classifiers, further accuracy is also doubtless attainable.

For smaller numbers of classes, the accuracy improves to greater than 97% for most two class problems, "Antibodies" vs "Behavioural Psychology" obtains 98% accuracy, for example. If the second class is instead a random selection of documents not in the first class, the accuracy drops to the low 90s as the maximum margin hyperplane to be discovered between the two classes is much less obvious when one of the two is essentially random.

Even difficult multi-class problems like "Bacteria" vs "Parasite" vs "Virus" vs "Neoplasm" training with only 3000 instances of each was able to obtain 81% accuracy. More widely separated classes, such as 'Head' vs 'Acids' vs 'Sleep Disorders' vs 'Ethics' trained with just 500 documents each obtained 87% accuracy.

These experiments were repeated using the Naïve Bayes implementation as well, however the accuracy was not as high, tending on average to be about 10% lower. The reason for this is strongly believed to be that the library discards the frequency information concerning each term and only records its presence, whereas the SVM implementation makes appropriate use of it. The CARM implementation is only slightly below the SVM accuracy, and while FDF has been used very successfully in other projects it was not applicable in this instance.

This demonstrates the possible improvements that can be made to textual classification analyses through the use of tools originally designed for text mining rather than data mining. Not only does the required number of attributes drop significantly, the accuracy increases at the same time.

6. CONCLUSIONS

Data mining, text mining and grid based tools within a digital library architecture have been shown individually in the past to enhance existing content and services. This research and development work goes one step further and demonstrates the new types of service and scales at which the techniques can be applied which are only made possible by simultaneously having seamless access to all of the components. The synergies between the components have been demonstrated, providing evidence both that the architecture is sound and scope for future applications is broad enough to warrant additional research.

In particular, an increase in accuracy in the resulting classification models has been shown, whilst at the same time reducing the number of attributes required per instance. This improves the

speed of the algorithm, decreasing the amount of time required to both learn the model and make predictions using it. The online services generated by the experimental service were typically of extremely high accuracy, greater than 95% in most 2 class problems and greater than 85% for most others.

The integration was also shown to be important for information retrieval purposes, reducing the size of the inverted indexes required without adversely affecting retrieval precision. At the same time, it solves many other problems related to stop lists, such as when non-nouns are used as proper nouns.

7. ACKNOWLEDGMENTS

Design of the Cheshire3 system was supported by the NSF and JISC (U.K.) under the International Digital Libraries Program away #IIS-9975164. The National Centre for Text Mining is supported by the JISC. The authors would also like to thank Ray R. Larson, Brian Rea and Clare Llewellyn for their input to the paper and research.

8. REFERENCES

- [1] Chang, C., Lin, C. "LIBSVM: a library for support vector machines", 2001 <http://www.csi.ntu.edu.tw/~cjlin/libsvm>
- [2] Coenen, F.P., Leng, P. and Goulbourne, "G. Tree Structures for Mining Association Rules". *Journal of Data Mining and Knowledge Discovery*, 8(1) 2004 pp 25-51
- [3] Diligent Project, "A Digital Library Infrastructure on Grid Enabled Technology", <http://www.diligentproject.org/>
- [4] Divmod, "Reverend", <http://divmod.org/trac/wiki/DivmodReverend>
- [5] Han, H., Giles, C. Lee, Manavoglu, E. et al. "Automatic Document Metadata Extraction using Support Vector Machines" In *Proc. 2003 Joint Conference on Digital Libraries (JCDL 2003)*, pages 37-48, Houston, USA, 2003
- [6] Hearst, M. A. "Untangling Text Mining", *Procs Annual Meeting of the Association for Computational Linguistics*. University of Maryland, June 1999.
- [7] Larson, R. R., and Sanderson, R. "Grid-Based Digital Libraries: Cheshire3 and Distributed Retrieval" In *Proc. 5th ACM/IEEE Joint Conference on Digital Libraries (JCDL 2005)*, pages 112-113, Denver, USA, 2005
- [8] Larson, R. R., and Sanderson, R. "Cheshire3: Retrieving from Tera-Scale Grid-Based Digital Libraries", *Procs. of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, (2006) p. 730, doi: 10.1145/1148170.1148343
- [9] National Library of Medicine, "Pubmed Medline", <http://www.ncbi.nlm.nih.gov/entrez/>
- [10] Oracle Corporation, "Oracle debates future Grid directions at EGEE 06", <http://www.oracle.com/global/eu/innovation/fs/egee06.html>
- [11] Rajesekar, M., Wan M., Moore, R., et al, "Storage Resource Broker - Managing Distributed Data in a Grid." *Computer Society of India Journal*, 33(4):42-54, 2003

- [12] Sanderson, R. and Larson, R. R. "Indexing and Searching Tera-Scale Grid-Based Digital Libraries" In Proc. First International Conference on Scalable Information Systems, Hong Kong, 2006
- [13] Tsuruoka, Y. and Tsujii, J., "Bidirectional Inference with the Easiest-First Strategy for Tagging Sequence Data" In Proc. HLT/EMNLP 2005, pp. 467-474
- [14] UCSD, "DSpace/SRB Integration Project", <https://libnet.ucsd.edu/nara/>
- [15] Witten, I., Tablan, V. et al. "Text Mining in a Digital Library", <http://www.dcs.shef.ac.uk/~valyt/download/greenstone-gate.pdf>