UNIVERSITY OF CALIFORNIA

SANTA CRUZ

Applications of Text Classification to Enterprise Support Documents

A thesis submitted in partial satisfaction
of the requirements for the degree of

MASTER OF SCIENCE

in

TECHNOLOGY AND INFORMATION MANAGEMENT

by

DANIEL CORE

June 2012

The Thesis of DANIEL CORE
is approved:

Table of Contents

List of Figures

List of Tables

Abstract

Applications of Text Classification to Enterprise Support Documents

by

Daniel Core

In the business world today there is a vast amount of information, and in order to process this information it must be structured. I develop a set of classifiers and corresponding user interfaces to assign tags to data that correspond with a structured framework. The classifiers are applied through a case study in the support area of a major networking company. The three classifiers provide tags for technical support documents with an F measure of .585, customer service requests with an F measure of .706, and customer support forum messages with an F measure of .78.

1 Introduction

In recent years there has been a huge growth in the amount of data available at corporations [3]. As corporate data grows, it becomes ever more important to find useful knowledge hidden within this data. By utilizing knowledge management and data mining techniques to process this data, significant gains in productivity can be achieved [14]. Much of this data is contained in free form natural language text documents such as technical support documents, forums, and customer service requests. Organizing this data becomes a great challenge because of both its volume, and its unstructured nature. The ability to extract useful information from natural language is difficult, however it is key to processing this vast resource of data.

In this thesis I develop predictive tagging, which begins by structuring this data under a common hierarchical tag set. By utilizing an expert-developed tag set we can provide a structured framework to classify text documents regardless of where their repositories are. Using a common set of tags developed by subject matter experts to divide the data into a structured format, it is possible to categorize corporate data in a meaningful way. In order to tag these documents we utilize machine learning algorithms to predict the most likely tag for a given document with a high degree of accuracy. This tag prediction process is effective in allowing more complex relationships to be found that otherwise would be blocked by having data in different repositories. The service area of technology corporations is unique, and through the application of machine learning algorithms we can begin to parse the relationships that exist in these data sets. We applied machine learning techniques to the support area of a major networking corporation. An example use case follows.

To predict the tag for a new piece of data, in this case a technical support document to be published to customers, we treated the tags as classes in the machine learning sense. One benefit of this predictive tagging approach is a more consistent method of tagging documents. It gives the option of automatically tagging other text documents with the same set of tags, thereby making it easier to identify documents with similar themes, even if the docments have different formats and purposes. The company we studied has many support documents written by engineers that

Title
How to Determine the Type of Supervisor Module That is Installed in Catalyst 6500/6000 Series Switches
Introduction
This document provides some simple checks that an end user can carry out in order to determine the type of Supervisor Engine module that a Catalyst 6000 or 6500 switch uses. The document describes a procedure to use while the Supervisor Engine module is still up an running in the chassis and a procedure to use when the Supervisor Engine module is removed from the chassis These procedures apply to Catalyst 6000 and 6500 series switches that run Catalyst OS software as well as IOS system software. ...
Primary Tag: Catalyst 6000 Series Switches
Secondary Tag: Catalyst 6500 Series Switches

Table 1: Example Text Document

are tagged through a common metadata framework (MDF). The support documents are written for the company's hardware and software products, as well as on general interest networking technologies such as internet protocol routing. The general process followed by the engineers for creating the documents consists of using a web site to create the documents, and then selecting a set of tags for each document manually. This process can be made more efficient by automatically suggesting the tags for the engineers when they are creating the documents. A portion of a sample document is shown in table 1. The documents are semi-structured through XML, making it possible to extract sections such as the title and introduction. In the case of technical documents we are able to predict the primary tag of a document with an F measure of .6, indicating a high ratio of precision to recall as discussed in 3.4. After developing classifiers for this application we followed classification models on customer service request documents, and support forums. For the customer service request documents we are able to predict the primary tag with an F measure of .706.

Additionally, with the support forum messages we were able to predict with average F measure of .84.

1.1 Contributions

This section summarizes the key contributions from the two main areas of this thesis. The first is the development of the overall software system and its deployment. The second area is the theoretical experimentation and discussion of the classification models.

The following are some of the main contributions of the software system. As part of this thesis I

- Provided an extendable platform for using predictive tagging of documents in the enterprise support area
- Developed integration of Weka [10], machine learning package, and Unstructured Information Management Architecture (UIMA) [6], natural language processing framework (NLP)
- Developed separate classifiers with a high level of accuracy within their domains

Some of the key contributions to the theoretical experimentation and discussion of the classification models follow. My work in this thesis

- Demonstrated the impact different machine learning algorithms have on the accuracy of the individual classifiers
- Investigated the most accurate method for combining individual classifiers either directly combining the data or utilizing a hierarchical method
- Investigated how the size and quality of the training data effects the final accuracy of the classifier

2 Overview of Technology

In the development of this work various open source tools were utilized, which played a key role in allowing for rapid development cycles without the high cost associated with this commercial software. The unstructured information management architecture (UIMA) software provided the overall framework and data parsing, while the Weka associated packages handled the classification. We are not aware of a previous open source integration between UIMA and Weka. I developed the integration for this project.

2.1 Unstructured Information Management Architecture(UIMA)

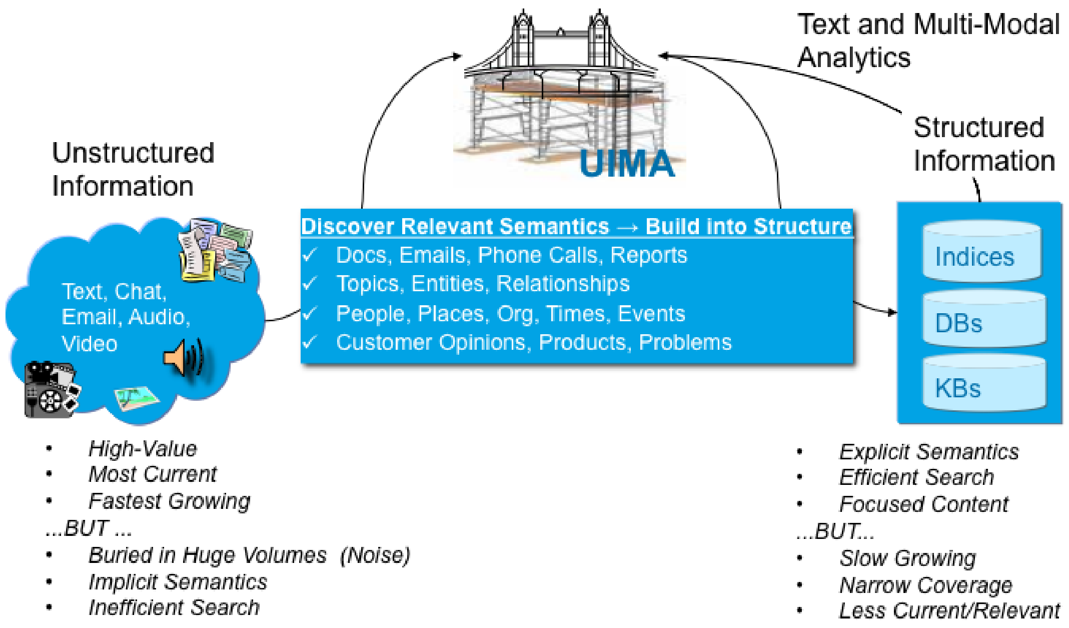

Text and Multi-Modal Analytics
UIMA
Unstructured Information
Structured Information
Text, Chat, Email, Audio, Video
Discover Relevant Semantics → Build into Structure
✓ Docs, Emails, Phone Calls, Reports
✓ Topics, Entities, Relationships
✓ People, Places, Org, Times, Events
✓ Customer Opinions, Products, Problems
Indices
DBs
KBs
• High-Value
• Most Current
• Fastest Growing
...BUT ...
• Buried in Huge Volumes (Noise)
• Implicit Semantics
• Inefficient Search
• Explicit Semantics
• Efficient Search
• Focused Content
...BUT...
• Slow Growing
• Narrow Coverage
• Less Current/Relevant

Figure 1: Unstructured Information Management Architecture (UIMA)

UIMA is a framework for handling NLP based problems [6]. UIMA was created as a project at IBM, and was showcased through the Watson Jeopardy challenger [5] for the deep question answering problem that was part of the Jeopardy quiz television program. The project was then spun off into the open source community, and is currently managed as an Apache project. The framework has become an industry standard for natural language components. Figure 1 shows the overall process that UIMA utilizes. The goal is to provide a framework that provides a bridge between the structured and unstructured worlds. Unstructured data is not just contained within the enterprise services sector but also with many other areas that are not limited to text retrieval including audio and video processing. Utilizing UIMA as a framework for deploying our classifiers provides many advantages.

The goal of the classifier is to provide a basis for future expansion for a unified method of tagging more enterprise support documents. To allow for the classifier to be reused and expanded the UIMA framework was used for the deployment [6]. UIMA provides a modular framework that allows for individual analysis engines to be recombined in different configurations to suit the current need. Analysis engines are individual processors that add annotations to the data. The type of annotation can vary from classification predictions to part-of-speech annotations. The analysis engines can be as simple as annotating sentences or applying the classifier and tagging the document with the proper tag. UIMA also contains a REST interface that can deploy analysis engines. REST services implement a common set of web interfaces, allowing for others to make HTTP requests to the system and have the processing done on a remote machine with the results returned. RESTful services provide a standard interface for others to interact with the classifier. One of the contributions of this project is the creation of an analysis engine that provides a bridge between UIMA and the classification package Weka [10].

2.2 Weka

Weka is a software tool that implements a standard set of machine learning algorithms as well as training, test, and many other capabilities [10]. Weka contains the major machine learning algorithms including decision trees, bayesian models, and K-nearest neighbor models. Weka provides both a GUI as well as an API that can be used from your own code. Weka integrates well within the UIMA framework because it is also java based, and therefore the APIs can be called from within UIMA. Weka provides the ability to quickly prototype the system by utilizing the built in classification training, and test facilities. This prepackaged method works well in our context because the novelty of the work comes from the application to the dataset, and the overall system rather than the development of more advanced machine learning algorithms. In order to process multi-label data we utilized a multi-label extension to Weka, MULAN [16]. MULAN is a library-based on Weka, and provides support for multi-label versions of the standard Weka machine learning algorithms including binary relevance and powerset methods. The features are more limited than Weka, only providing programmatic access through an API. However it still provides training, and test facilities.

3 Survey of Methods

The following section provides an overview of the technical methods used in developing the various classifiers. The process of classifier development begins with the preprocessing of the data into a machine usable format. Once in a usable format the generation of the models can begin. When the models are trained they are then evaluated using different methods.

3.1 Representation of Documents

In order to classify documents, each document needs to be represented in some manner appropriate to the method being used. Documents can be represented nominally, with the actual characters being used, or numerically, generally as a variation of a numeric word vector. We focused on representing the documents as word vectors. Different aspects of the document can be stored in the vector. The bag-of-words approach treats each word as a token, and represents its presence in the document as a binary system. This results in a document set that treats the presence or absence of each word with equal importance, and records nothing about its frequency with respect to the document or corpus. The bag-of-words model does not record any ordering of the words in the document. The lack of order prevents associations being made as to where in a document words appear. The frequency problem can be mitigated using term frequency(TF) and inverse document frequency(IDF) transformations. The term frequency looks at the frequency of the term in the document. The more frequently the term appears in the document, the more likely it is to be important. The IDF transformation records the inverse frequency of the word appearing in the whole corpus. The more frequently a word appears in the corpus, the less unique it is, and therefore its presence contributes less to the class of a particular document. We utilized a bag-of-words model, as the accuracy is not necessarily improved by the transformations, and the process adds complexities to the system.

3.2 Model Generation

The model generation phase can be made of different types of machine learning models. These models fall into three general categories: probabilistic models, decision tree based models, and lazy learners. In this thesis I utilized the naive Bayesian model, c45 decision tree, and K-nearest neighbors models. They represent a wide range of methods that can be used in the prediction of classes.

3.2.1 Naive Bayes Classification

Naive Bayesian classification is a relatively simple technique for creating classification that, despite its simplicity, is effective in many applications. While there have been many new and novel techniques developed recently, the naive Bayesian classifier remains relevant because it performs similarly to new techniques in many cases [13]. The naive Bayesian classifier utilizes the basic probability rule known as Bayes' theorem.

$$P(c_k|x) = P(c_k) \times \frac{P(x|c_k)}{P(x)} \tag{1}$$

Equation 1 shows Bayes' theorem in its original form. This equation can be used to calculate the probability of a class c_k given the probability of a document being of type x and the likelihood of c_k.

$$\widehat{P}(c_k|x) = \frac{\widehat{P}(c_k) \times \prod_{j=1}^{d} \widehat{P}(x_j|c_k)}{\widehat{P}(x)} \tag{2}$$

The theorem can be simplified to equation 2 where $\widehat{}$ indicates an estimation, k indexes the possible document classes, and j indexes the possible document features.

It is possible to estimate the probability that a document falls in to a class c_k by utilizing the overall estimate of the probability of c_k. This probability is then multiplied by the product of the joint probability of the documents features x_j given the class c_k. In effect the probability of $P(x)$ can be ignored because it will be the same for all documents. This allows for efficient calculation of the posterior probability of the class. To allow for this simplification an independence assumption is utilized [13]. The evidence created from each text feature is used independently to predict the overall likelihood of a given class.

There are several advantages to using a naive Bayesian classifier. The naive Bayesian classifier is relatively simple to calculate as compared to other more complex techniques; therefore it utilizes less computational resources. Because there is little extra analysis in creating the model other than manipulating the counts of the features in each document the Bayesian classifier is time-efficient to create. The speed of creation can yield a competitive model in many cases [2]. Because the Bayesian model is based on counts of features, it is simple to update. The update process requires updating the overall counts of the word features and the counts of the word features within the new documents class.

There are many disadvantages to using Bayesian classifiers. Because the model relies on the strong independence assumption that all features are unrelated there is a potential for lost information, especially when the theorem is applied to textual data. In natural language there are clear constructs for writing, and in theory we should be able to use these for increased prediction accuracy [13]. Although it would seem logical to have gains in accuracy through complex models, many times the gain in accuracy is negligible given the increase in complexity in the model. Another disadvantage is document length. The difference in length of two documents can effect the predicted class significantly. The increased length results in more words being used, and in the bag-of-words model these are given the same significance as a single word in a short document. The words in the short document are more likely to have a higher significance because the document has less length. It is possible to mitigate the document length problem through normalization.

3.2.2 Decision Trees

Decision trees are models that utilize the document vector to create a tree structure to predict the end class of the document. In this case a Java-based implementation of the C4.5 tree algorithm was used [10]. The C4.5 creates a tree by splitting up the training set features by using information gain [15]. The tree is split on the attributes that create the highest information gain. The tree is split until one of the following conditions is met: the remaining training data all consists of the same class; there is no information gained by splitting the tree on the current attribute; or a new class is encountered causing the tree to create a new node. Once the tree is built there are many advantages to using a tree based model. The tree based model provides an efficient means to classify documents. Splitting on attributes is generally a time efficient method to create the tree, and once the tree is created it is not computationally expensive to apply.

The disadvantages of the tree based model are that it can over-fit the data causing inaccurate predictions. This inaccuracy is accounted for in the model training through pruning the tree branches. Over-fitting is a result of the model using every feature to create the tree. Pruning the tree reduces this over fitting by eliminating branches that do not contribute to the accuracy of the classifier [15]. The branches containing splits that cause errors are pruned back up the tree, reducing over fitting. In addition, decision trees suffer when the classes are not distinct. Decision trees can be thought to break up the attribute space into smaller subsections, and if there is significant overlap between sections the tree cannot accurately predict the outcome. However these dataset related problems are faced with most machine learning models.

3.2.3 K-Nearest Neighbors

Nearest neighbor based models are simple classifiers that utilize the word vectors of a document to predict its class. In this model the documents are projected into N dimensional space,where N is the number of possible attributes, and their proximity is calculated with the nearest K documents being selected to predict the final class.

K is the number of nearest neighbors that are chosen to make the classification. The final class is very sensitive to the value of K chosen [11]. When training the model, only the document vector and the training tag are stored. Because there is little information necessary to make the prediction the training portion of the model is very short. Therefore the calculation of the neighbors is pushed to run time; however, the calculation has been highly optimized [1].

3.3 Ensemble Classification

Utilizing many classifiers can increase the overall accuracy of a prediction significantly. There are two major methods of ensemble classifiers; bagging and boosting. Bagging classifiers break the training set up randomly and use the data to train a set of classifiers. This can increase the classification accuracy because each classifier can classify a specific section [4]. In boosting, an ensemble of classifiers is trained using the errors from the previous classifiers in the set. For this project the AdaBoost method was used [7]. The AdaBoost algorithm is a boosting algorithm that does adaptive weighting to learn from all iterations of a given classifier [7]. This adaptive boosting provides higher accuracy.

3.4 Evaluation Measures

If the true positives tp, and false negatives fn are known then the following metrics can be used to evaluate a classifier.

$$Recall = \frac{tp}{tp + fn} \tag{3}$$

$$Precision = \frac{tp}{tp + fp} \tag{4}$$

$$F = 2 \times \frac{precision \times recall}{precision + recall} \tag{5}$$

Evaluation of the classifier was done by using cross validation using the training set. Cross validation takes the training set and breaks it into a training and test set randomly. This process is repeated over multiple iterations to create a set of average statistics for the classifier. The statistics calculated include the accuracy which is both the true and false positives over the total amount of predictions. The precision indicates the overall quality of the classifier, while recall indicates the quantity of the documents that are classified correctly. The F measure is the harmonic mean of precision and recall with a number close to 1 indicating both high precision and recall. These measures give a good picture of the overall usefulness of the classifier. The level of effective F measure varies for different data sets. In some data sets it is more important to have high precision where recall may be lower resulting in a lower overall F measure. In addition to the cross validation of the classifier we utilized evaluation on the training set to create an upper bound for the classifier. We calculated the same metrics when evaluating on the training set. Also because the classifiers are being used in a business setting we also looked at the subjective accuracy. The tags predictions must be accurate but more importantly they must also be useful to the end user.

4 Related Work

There is a significant body of work in the application of classification models to different datasets from many areas in business applications. Most of the datasets are not focusted on developing classifiers with consideration to a production system. One such exception is the work done at IBM by Godbole and Roy where they applied classifiers to survey feedback data [9]. The IBM system utilizes similar tools to our system.

The keys to the system are using the UIMA [6] framework for handling the data processing elements, and WEKA [10] for the development of the classification. UIMA handled the collection and processing of the data. The system used UIMA to provide different classification methods for the data. Godbole and Roy utilized an expert rule based system to classify the low frequency high value classes for which the training data was insufficient. The rest of the classification was handled by models developed through WEKA. The key part of their system was the use of a feedback loop within the process that allowed a review of proposed classes for customer feedback documents.

Our system is similar to Godbole and Roy in that it utilizes similar software components. However our work differs in its goals, and data used to accomplish them. The data is relevant because it is from the enterprise support area, and is unformatted text. The sources for our data differ from Godbole and Roy because we are using the primary sources text rather than transcripts of phone calls. We faced similar problems to Godbole and Roy in generating the training data for the project. Deciding on the classes is a difficult problem that was faced by Godbole and Roy. Godbole and Roy did not have a predefined set of classes they wished to use. The metadata framework already in place prior to our project allowed us to move directly into the training of the classifiers.

Godbole and Roy faced problems with how to create accurate training data that reflected the true nature of the documents. Repeatability and reproducibility were issues faced in the development of their classifier, [9]. The repeatability problem is the consistency of an individual labeling documents over time, and reproducibility is the consistency of different labelers over the set of training documents. In our initial test case we did not have these issues because the documents were added over time, and had been reviewed by domain experts. However, this was an issue in subsequent training sets where the tags had not been reviewed, and had to be generated specifically for the purposes of training the classifier.

The Godbole and Roy work differs from ours in the types of classes being utilized. They were interested in the classification of documents related to customer satisfaction, which is more sentiment analysis rather than subject identification. Their goal was to understand what the customer felt, and classify it into a category such as “improper accent” or “poor voice quality” [9]. Identifying the subject of a document results in tags such as products or technology names. This difference in goals leads to more concrete accuracy judgments, and makes it easier to evaluate the accuracy. Because the goal of our work is subject identification we do not face as many issues related to repeatability and reproducibility.

In our datasets many of the documents are not exclusive to a single class;

therefore we explored the use of multi class classifiers to try and obtain a more relevant set of tags for each document. Katakis et al. [16] utilized a multipliable prediction process to recommend tags for a social media bibliography web site. The process for recommending those tags is very similar to the method we used to create our suggested tags when considering a multi label problem. Multi label classification trains data by using many tags for each document. The classifier is most commonly trained using a binary relevance decision. This means that a classifier, in their case a naive Bayesian classifier, is trained for each pair of tags, and is then used to produce the final prediction. Katakis et al. utilized a multi label prediction implementation in the open source package MULAN. Their work differs from mine in that it uses a user defined set of tags to learn from rather than a defined set that has been predetermined. The social aspect of the bibliography web site means that there are a number of issues associated with the definition of the tags themselves. In addition because our data contains primary tags the problem can be further simplified to a single class problem.

Related to this project is the field of tag recommendation in [8]. It seeks to recommend a tag for a document similar to what we strive to accomplish. However it is generally approached from the goal of incorporating social networks to inform the decision on the tag recommendations [8]. The focus of their work is to look at both the content of the document as well as the content of users. In [8] they utilize a dataset from the *flikr* website to recommend tags a new picture uploaded to by the user. They use the current tags entered combined with knowledge gained from other similar users to recommend new tags. The recommendations are based on similar machine learning models as we utilized. This area is rich, and can be explored in future work where tags can be correlated with forum users. Currently the focus of the classifiers is on the document content itself but in future work it would be possible to leverage the user data present in the forums.

5 Development Process

We developed predictive tagging systems for different document types used for customer support in a major networking company. The following shows the development process and results for each document type. Three distinct classifiers were developed for technical support documents containing engineer written content, customer service requests documenting how customer problems are resolved, and customer support forums where customers can discuss their problems. The results from these classifiers demonstrate the high accuracy that can be obtained through the use of classification in this space, providing advantages for the company.

5.1 Technical Support Documents

Technical support documents are written by engineers at the company for consumption by outside customers. The documents are typically written on support subjects relating to individual products. The documents are written by internal engineers who are experts in the products. When creating the documents the engineers give the document a metadata framework (MDF) tag that can be considered ground truth because of their expert status. The mapping of the technical support documents provides a test of the possibilities for classifiers in this space because of the availability of the ground truth. The key insight gained through the development process is how to overcome a sparse dataset to create a strong classifier.

5.1.1 Data Set

The training data for the technical service document classifier was taken from the technical support documents. The technical support document data is sparse in that there are only a few documents corresponding to each tag. The training set of 5000 documents is spread across 1000 unique tags. This results in an average of 5 documents per tag; however, as shown in shown in figure 2 the tags are both sparse and unevenly distributed. There are some large categories but the majority of tags correspond to fewer than 25 documents. Because of the distribution, it is difficult to have enough data for each tag to make an accurate prediction.

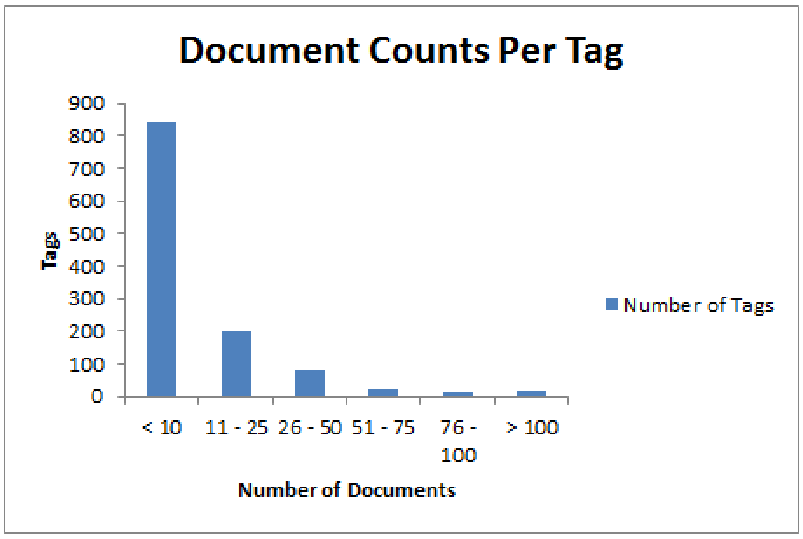


Figure 2: Technical Support Document Tag Counts

To combat this sparseness we experimented with different methods including augmenting the data set with data from product manuals, aggregating the data using the MDF tree, and utilizing thresholding. The product manuals increased the document count from 10,000 to 150,000, however this failed to increase the accuracy. The product manuals contained little variability in the titles of the books and sub headings. Because there was little variability in the features across the tags it is difficult to distinguish the tags. We attempted to solve this problem by processing the product manual titles by hand but this did not significantly improve any of the evaluation measures.

Another method for mitigating the sparseness problem was aggregating the tags by the MDF hierarchy. The tags were organized hierarchically and therefore it was possible to select the lowest relevant level in the hierarchy to reduce the tag set increasing the number of documents within each tag. To further reduce the tag set we set a threshold at different levels to take only the tags that had over the threshold number of documents within them.

Once the training set was processed it was possible to parse the documents into a format to train them.The documents contained titles as well as content that can be utilized for training classifiers. The titles were utilized along with selected text content. The text was parsed into a word vector representation through a whitespace tokenizer. The word vector representation treated the document as a bag-of-words where order and frequency are not preserved. This preprocessing was done through two steps. The first was taking the documents out of the database they were contained in and parsing out the relevant tags and data. In order to filter and train the classifiers the open source package Weka was used [10]. Weka contains classes for converting directory structures into the proper format to train the classifiers, because MULAN [16]is built on top of WEKA [10] the same training classes can be used.

5.1.2 Classifier

The accuracy of the classifier varied based on the techniques used to develop it. The most accurate model evaluated on the training set was a boosted Bayesian model because the features can be assumed to be independent in such a short title. The boosting significantly increased accuracy on the training set by retraining the classifier on just the errors from the previous classifier. Theoretically this technique can create a perfect classifier on the training set. The danger of the boosting process is that it will over fit the data, and when it is applied to new data the accuracy decreases. There was some over fitting but the gain in accuracy on the training set was worth the slight loss in the cross validation tests. The accuracy of all classifiers was significantly less when evaluated through cross validation. Cross validation randomly breaks the training set into test and training sets and then iterates. The statistics for precision and recall are then averaged over the iterations producing an evaluation that better represents how the classifier would predict for new data. The reason for the relatively low precision and recall of the classifier lies in the data used to train it. The data for the training set contained a very high ratio of tags to total documents. The documents were also represented by their title which is a relatively short string to classify on. Overall the classifier is highly accurate within the training set. This high accuracy validates the usefulness of a classifier within this domain by proving that it is possible to assign tags in a meaningful way, similar to that of a human expert.

5.1.3 Classification Results

Metric	Training	Cross Validated
F-Measure	.88	.585
Precision	0.9	.669
Recall	0.88	.596

Table 2: Technical Support Document Classifier Results

The F-measure in the boosted Bayesian model on the training set was .88, whereas the F measure on the cross validated boosted Bayesian model was .596. For titles that are exactly the same the classifier performs well because of the boosting. As discussed, boosting can result in over fitting of the data, and a loss of accuracy over a non boosted model, but in this case the effects of the over fitting were minimal. The cross validated classifier represents how the classifier performs with truly new data. Given the ratio of the tags to documents the low F-measure for the cross validated classifier is to be expected. While the cross validated classifier F measure is low it is still useful as a proof of concept for the tagging process where most of the subjective validating was done through using the training set. The false positive rate is very low at .01, meaning that the average ratio of wrong predictions to the size of the class is low. Increasing the accuracy would be possible as the classifier is used in production when there are new documents created. However there is a low frequency of document creation within the technical support document repository, around 10 documents a year, so there is little to gain in business value through that process.

Classifier	F Measure
Bibtex Multilabel Binary Relevance with Naive Bayes [12]	0.0942
Multi Label Binary Relevance with Naive Bayes	0.13
Naive Bayes with Single Classifier	0.606

Table 3: Comparison of Single and Multi label Classifiers

Several tests were conducted on the use of all tags, both primary and secondary, and the use of the single primary tag. The results shown in table 3 demonstrated the F measure of both. The multi tag method performed better than [12], however was not as accurate or as high an F measure as the single tag methods. The multi-tag classifier was still slightly more accurate than the best in [12]. The naive Bayes model with the single tag classifier was thresholded with only tags that contain more than 50 documents. This resulted in a tag set of 110 of the most frequent tags. It was possible to simplify the multi tag problem to a single tag in our case because the engineers denote a primary tag for each document. The F measure in this case is more a relative measure. The difference between the primary tag method and the single tag method is large, indicating the primary tag method as the better method.

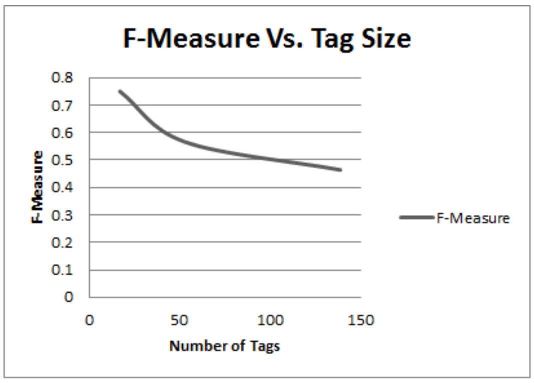


Figure 3: F Measure Vs. Number of Documents in Each Tag

The single tag methods utilized only the primary tag of the document. The accuracy of the classifier was heavily dependent on the number of tags used, as shown in figure 3. As the number of possible tags increased, the F-measure was reduced. This decrease can be seen in general for classifiers; however in this case it is more prevalent due to the ratio of tags to documents, which overall is 6.8 documents per tag without aggregation. By using aggregation the ratio was increased to an average of 71 documents per tag. This resulted in increased accuracy as well as usability because the tags are more relevant. This relevancy comes from using a smaller pool of tags that correspond to the most useful tags to begin the thresholding process. Because all tags in this classifier are from the software family, hardware product series or technology space they only contain meaningful tags.

5.2 Customer Service Requests

The customer service request data set is made of service requests from customers who experience problems with their products, and have service contracts with the company. This classifier builds on the TacWeb classifier and shows the benefits of quality training sets over the size of a training set. Also, if expanded, the service request classifier could provide internal support engineers with an advantage when looking for similar service requests.

5.2.1 Data Set

The customer service request domain is the next area to which we expanded the technical support document. The service requests records the full interaction between a customer who reports a problem, and the engineers that are trying to solve it. The requests contain a high level of irrelevant and repetitive material. Eliminating this noisy data programmatically is a difficult challenge currently being worked on by others. To quickly develop a training set we handpicked representative requests of each tag by using the company's internal search tool to look up requests relevant to each tag. By picking the training set by hand we eliminated much of this extraneous information. This resulted in a training set that was more relevant to training the classifier. The goal of this classifier is to investigate whether quantity or quality has a greater impact on the accuracy of the classifier. The quality of this data is much higher since it is handpicked and the documents contain much more than a title. The documents range from a sentence to a few pages of data that is highly relevant to the tag. This eliminates the noise found in the other training set. It is possible to increase the service request data because there are hundreds of thousands of unique service requests that could be added to the training set. The key for this process would be to utilize a programmatic process to do this. Another test to run would be to see how using the full service requests with the noise would effect the accuracy. This would simplify the additions. To make the service request classifier more relevant for business use a new set of tags should be created based on the current most popular tags. However, the majority of the technical support document data is around ten years old, resulting in much of the tag set being out of date, and providing classification for products that are no longer in service.

5.2.2 Classifier

Metric	Training	Cross Validated
F-Measure	.99	.706
Precision	.99	.717
Recall	.99	.71

Table 4: Service Request Results

The service request classifier works on the same software framework as the technical support document classifier. The classification scheme used is a tree based model. The tree based model was used in order to increase the accuracy over a Bayesian model because the data contained features that could easily be branched. Utilizing a boosted tree created a much stronger overall classifier when evaluated against both the training set as well as the cross validated set. Much of these were explicit model names and series that are contained in the training data. Text features contained in the document such as “catalyst 6500” for the “Catalyst 6500 Switch” tag make it easier to branch on and then use the rest of the document to further increase the certainty of the final prediction. The targeted data extraction in the service request classifier resulted in a stronger training set with less documents. The classifier had an F-measure of .706 which is higher than than the technical support document classifier for the same number of tags. This can again be attributed to the hand selected data set. The false positive rate was slightly higher at .005. This indicates that the classifier was more likely to identify something with the wrong class. However this is not a large change, and does not affect the overall performance of the classifier. The F measure of .706 is high enough that the majority of the time the predicted tag is correct. The precision and recall are close and because of this the F measure indicates that not only will the majority of predicted tags be accurate, but also the majority of the available tags will be tagged.

5.3 Customer Support Forum Messages

The customer support forum classifier represents the highest potential for business gains and impact with customers. As it currently stands, there is no method to correlate the messages and threads contained in the public support forums to the internal MDF tags. Correlating the messages to the MDF tags will provide the ability for customers to be presented with other relevant documents and messages that would be otherwise difficult to find. By utilizing the same framework as the other classifiers we were able to implement a multilevel classifier to predict the class for a given message or thread. The key contributions developed through this classifier are the generation of the training data by automatically assigning tags through a keyword based search method and utilizing a multilevel hierarchical system of classifiers to increase the overall accuracy of the classifier.

5.3.1 Data Set

To develop the training data for the forum classifier we were faced with many challenges because no tagged data were available. Developing the training data in a time efficient manner required a unique approach utilizing automation in the training process. We utilized an in-house tool with the ability to query the forum repository with a text string and retrieve forum messages back. We queried the forms with the MDF text strings and assigned the returned messages the MDF query tag. This automated process shrank the development time from weeks to days over creating the training set for the service request classifier, while maintaining accuracy. As shown in [9] the development of a training set is difficult even when using trained engineers. The effectiveness of the automated process is demonstrated when compared with other classifiers. Even though the tags were assigned through an automated process the predictions are highly accurate, and when implemented the continual updating of the classifier can eliminate much of the inconsistencies in the training data.

The tag set for the classifier was intended to be as broad as possible. It incorporates the majority of the software family, product series and technology tags

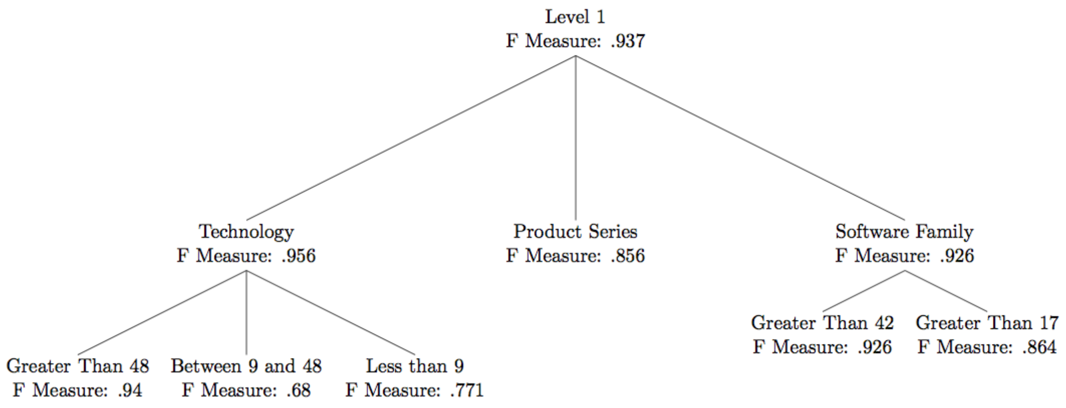


Figure 4: Forum Classifier Structure

that are present in the MDF. The other classifiers developed only incorporate a small portion of the total MDF because they had limited data available. The forums contain tens of thousands of unique messages that can be searched for individually.

5.3.2 Classifier

The breadth of coverage of the MDF tree necessitated a unique solution for reducing the number of tags for the classifier so that it could maintain high levels of accuracy. In order to create the highest accuracy the classifiers were broken up by their document count. As the technical support document classifier results in figure 3 show the accuracy diminishes as the tag count increases. From the previous experiments shown in figure 3 the ideal tag count for this type of document is between 100 and 150. Therefore the classifiers were trained on the data whose counts fell into that range.

The forum classifier structure shown in figure 4 shows how the classifiers are broken up. The lowest level classifiers are designated by the document counts they contain. The higher the document count the more accurate the classifier. The grouping of tags by their document counts was chosen because inherently in this service document domain the lower the number of training documents the less accurate the classifier is. To maximize the level of training data, and therefore the chance of distinguishing the different tags, breaking by the count was chosen. Future work in this area could include clustering the tags, and creating classifiers on the most similar sets. This could be a method for increasing accuracy. To provide a base for comparing each type of document, the support forum messages were used to train a single classifier which was then evaluated against the same tags contained in the technical support document data set the accuracy of this classifier is as shown in table 5.

6 Discussion

The individual performance of the classifiers is one aspect of the project. We also evaluated the different classifiers across datasets. We evaluated the classifier utilizing the same set of MDF tags to compare the effects of the different datasets on their accuracy. The datasets differ in both the quantity and quality of the data contained. The insights gained from this comparison showed that utilizing a larger set of data containing both a large number of documents and with significant content resulted in the greatest accuracy. The support forum data was more accurate than the service request data that contained more content per message as well as the technical support document data set which contained high numbers of individual documents but lacked length.

6.1 Evaluation of Classifiers on the Same Tag Set

Classifier	F Measure: Training Set	F Measure: Cross Validated
Technical Support Document	.891	.588
Service Request	.902	.592
Forum	.93	.78

Table 5: Support Forum Results Using Technical Support Document Tag Set

As shown in table 5 the classifier accuracy improved as more quality data was introduced. The F measures shown are for unboosted models because they provide a fair comparison of the actual accuracy of the classifier. The three datasets were evaluated against the same tag set. The technical support document classifier was the lowest because of its scattered dataset. The technical support document set, although labeled by experts, was the least accurate over the test set. Also the technical support document set contained the most documents. The low accuracy can be explained through the repeatability and consistency issues faced in [9]. When compared with the small but quality dataset handpicked for the service request classifier the technical support document classifier cannot compete. This dataset contained less noisy data and also contained more per document than the technical support document set. This focused selection of the most relevant part of a document resulted in higher accuracy. The final dataset the forum classifier was the most accurate because of both the quantity and quality of the data. The dataset had more instances than the service request dataset and, although they were not handpicked, the automated method returned a consistent set that could be used for high accuracy. The messages in the forums contain more focused information than the technical support document set, and resemble more of the information in the service request set.

6.2 Combining Classifiers

In order to increase the usefulness of the classifiers we attempted to combine the data sets in different ways. The first method was simply combining the data from the technical support document and service request data. The data was combined by putting the documents into their corresponding tags regardless of the source. This resulted in a loss of accuracy with the F measure going to .42. The loss can be attributed to the difference in the data sets. The service requests are written by a different set of engineers for an internal engineering audience as opposed to the technical support document data that is written for customer consumption. This results in a different point of view as well as more utilization of acronyms for different products and types. The data sets represent two disjoint sets that benefit from separate classifiers.

The second method we tried was to take advantage of the accuracy of the more specific classifiers developed for each data set by utilizing a hierarchical classifier shown in figure 5. To do this we trained a Bayesian classifier on the binary classes of service request or technical support document. The differences in the

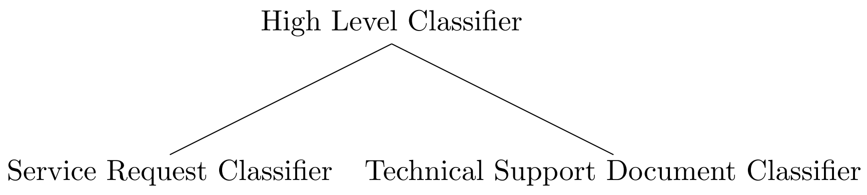

High Level Classifier
Service Request Classifier
Technical Support Document Classifier

Figure 5: Hierarchical Combination of Classifiers

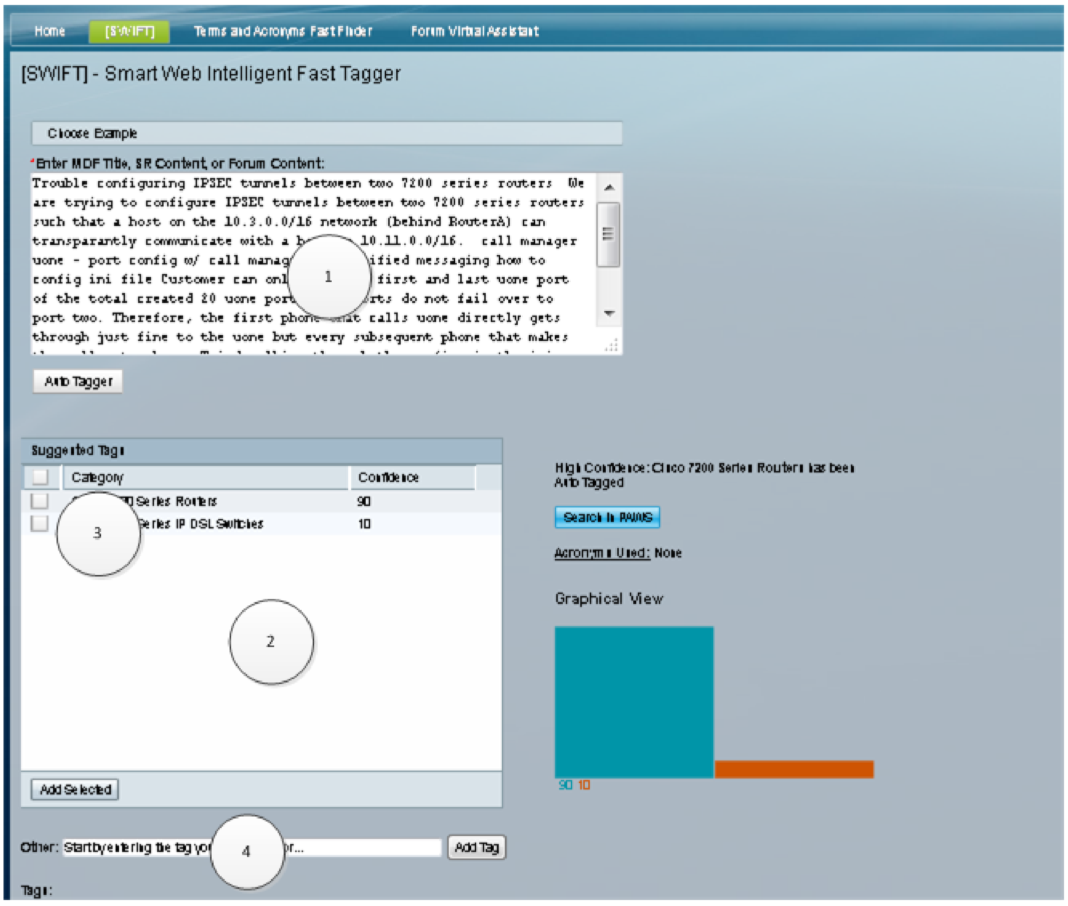
word choices of the documents are evident because the classifier had an F-measure of .98. This is sufficiently high enough to allow this binary classifier to first classify a document as one of the two types of documents, and then pass them to the other classifiers to distinguish the tags. Because of the high accuracy of the classifier there is no loss over hand selecting the document type.

7 Application of Classifiers

One of my key contributions is the use of these classifiers in business applications. The following use cases show the usefulness of being able to connect data across repositories. There are currently three methods of interacting with the classifiers. The following applications all rely on the integration between UIMA and Weka that I developed. The proof of concept interface allows users to test each classifier and evaluate its performance on tagging each document. The forum interface is a practical customer-facing use case. This page simulates the process that a customer follows when creating a forum post, and can suggest related material to their post. Additionally programmatic access is provided through a REST interface allowing other engineers to develop their own projects utilizing the classifiers we developed.

7.1 Proof of Concept Interface

The user interface was developed using a rapid prototyping approach changing over time in order to meet the needs of the engineering team. By utilizing user interface components already available at the company, we were able to quickly respond to changes in the interface. The initial prototype did not contain the graphical representation of the different results, or the application of the other analysis engines developed. The goal of the tagging began with the idea of suggesting tags for en-



[SWIFT] - Smart Web Intelligent Fast Tagger

*Enter MDF Title, SR Content, or Forum Content:

Trouble configuring IPSEC tunnels between two 7200 series routers We are trying to configure IPSEC tunnels between two 7200 series routers such that a host on the 10.3.0.0/16 network (behind RouterA) can transparantly communicate with a h 10.11.0.0/16. call manager uone - port config w/ call manag ified messaging how to config ini file Customer can onl first and last uone port of the total created 20 uone port rts do not fail over to port two. Therefore, the first phone at calls uone directly gets through just fine to the uone but every subsequent phone that makes

Suggested Tags

	Category	Confidence
☐	0 Series Routers	90
☐	eries IP DSL Switches	10

High Confidence: Cisco 7200 Series Routers has been Auto Tagged

Acronyms Used: None

Other: Start by entering the tag yo r...

Tags:

Figure 6: Proof of Concept

gineers to add to their documents. The focus has since shifted to automatically tagging the documents if sufficient accuracy is obtained. This automatic tagging is indicated by the “High Confidence:” message. This can be used in the future to place these highly confident tags into a repository.

The following are the steps an engineer goes through in utilizing the interface:

1. The engineer enters the text they wish to tag into the text box.

2. The system presents the engineer with the sorted set of tags meeting a minimum prediction confidence level of .001

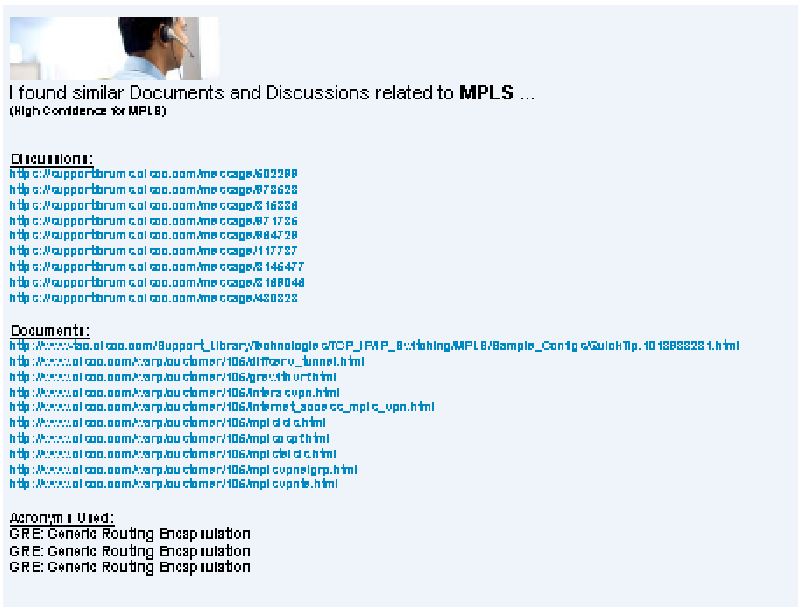
3. The engineer selects the applicable tags in any amount, however denoting one as the primary tag

4. If the engineer believes that none of the tags displayed from the classifier are

correct they have the option of selecting from an “other” category hat contains the set of all possible tags

7.2 Forum Interface

Building on the first two interfaces, we developed the forum classifier interface shown in figure 7. The goal of the forum classifier interface is to provide the end user with recommendations of related customer support forum messages as well as technical support documents. When the user enters a question or topic for a post it is then fed through the forum classifiers through the REST interface, and a class prediction is returned. The returned value is shown as “I found similar documents to” the predicted tag. To increase the actual usability of the system the user is presented with documents and messages that were also tagged with the same MDF tag. This correlation is made to the other support forum messages as well as technical support documents.


I found similar Documents and Discussions related to **MPLS** ...
(High Confidence for MPLS)

Discussions:
https://supportforums.cisco.com/message/602288
https://supportforums.cisco.com/message/873628
https://supportforums.cisco.com/message/3 16338
https://supportforums.cisco.com/message/871786
https://supportforums.cisco.com/message/884728
https://supportforums.cisco.com/message/117787
https://supportforums.cisco.com/message/3146477
https://supportforums.cisco.com/message/3188048
https://supportforums.cisco.com/message/430828

Documents:
http://www-tac.cisco.com/Support_Library/Technologies/TCP_IP/IP_Switching/MPLS/Sample_Configs/QuickTip.10188823 1.html
http://www.cisco.com/warp/customer/106/diffserv_tunnel.html
http://www.cisco.com/warp/customer/106/grewithvrf.html
http://www.cisco.com/warp/customer/106/interasvpn.html
http://www.cisco.com/warp/customer/106/internet_access_mpls_vpn.html
http://www.cisco.com/warp/customer/106/[illegible].html
http://www.cisco.com/warp/customer/106/[illegible].html
http://www.cisco.com/warp/customer/106/[illegible].html
http://www.cisco.com/warp/customer/106/mplsvpneigrp.html
http://www.cisco.com/warp/customer/106/mplsvpnte.html

Acronyms Used:
GRE: Generic Routing Encapsulation
GRE: Generic Routing Encapsulation
GRE: Generic Routing Encapsulation

Figure 7: Forum Classifier Interface

7.3 REST Interface

REST provides programmatic access to the different analysis engines that we have created in UIMA. The REST functionality is built into UIMA and can be deployed simply. This deployment strategy is effective in allowing the flexibility, and openness that is necessary to allow expandability in the future. Because the REST interface is standard, it does not depend on the client system that is accessing the analysis engine. This increases the overall business impact as other groups can access the system without needing to have access to the code. The REST interface provides the back end of the system for the other user interfaces.

8 Future Work

This work can be widely expanded both internally to this company as well as to the greater community. Because it is built on the modular UIMA platform the expansion and addition of more classifiers and other natural language processing elements is very simple. There are many more unstructured data repositories contained within the company. The more links that can be built among these different repositories the deeper the insights that can be learned. It is possible to develop further classifiers in a similar way to the current classifiers. Within the forums there is more interesting processing that can be done when the data on users is considered. The user data can be linked to MDF tags providing answers to questions such as which users are experts in a certain tag or what is the most popular document associated with a user.

The users can also form a critical part of the system by initiating a feedback loop. By providing a method for the users to choose whether the predicted tag is correct the classifier can be improved. The feedback loop continuously improves the results of the system. To implement this feedback the model must be rebuilt to accommodate changes. For a Bayesian classification model this is simple because the counts of the features effected are the only changes that need to be made. When the classifier is updated the new model is more likely to predict correctly. As shown in [9] the feedback loop is essential for creating usability. Because the support forums experience a high volume of traffic the classification can significantly improve.

Another use case can be further sorting the recommendations given for forums through the use of search. This would provide a ranked list of similar documents as opposed to showing a list of the documents. The unsorted list is useful but a better correlated list will be more useful to individual problems being experienced by the user who is creating the post or submitting the text.

Generalizing this approach of utilizing classifiers to unify repositories is also useful outside of the enterprise services area. Because of the modularity of the system the framework can be applied anywhere a suitable dataset can be found. The biggest challenge would be to create the hierarchy of tags. In this case the tagging system was preexisting. Once the tag system is created the training and application process is straightforward, and could be applied in various situations. The types of data that would be suitable could be emails, phone transcripts or other free form messages that are not part of a system.

9 Conclusions

We have shown through the application of classification to three unique datasets that it is possible to accurately predict labels across different types of documents. We have shown that increasing accuracy for classifiers can be accomplished through the use of hierarchical classification. The difficulties of working with a sparse data set have been shown, particularly enterprise service data. The results of using classifiers has enhanced both the search and data mining capabilities of the corporation by providing the links across different repositories. The interfaces created have the ability to enhance user experience with the customer support forums by showing relevant documents that was previously not possible, thereby increasing customer satisfaction. With increased satisfaction comes increased sales leading to measurable benefits for the company.

References

[1] Kjersti Aas and Line Eikvil. Text categorisation: A survey. http://citeseer.nj.nec.com/aas99text.html, 1999.

[2] Nahla Ben Amor, Salem Benferhat, and Zied Elouedi. Naive bayes vs decision trees in intrusion detection systems. In *Proceedings of the 2004 ACM symposium on Applied computing*, SAC '04, pages 420–424, New York, NY, USA, 2004. ACM.

[3] K Cukier. Data everywhere. *The Economist Newspaper Limited*, Feb 25th 2010.

[4] Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40:139–157, 2000. 10.1023/A:1007607513941.

[5] David Ferrucci. Build watson: an overview of deepqa for the jeopardy! challenge. In *Proceedings of the 19th international conference on Parallel architectures and compilation techniques*, PACT '10, pages 1–2, New York, NY, USA, 2010. ACM.

[6] David Ferrucci and Adam Lally. Uima: an architectural approach to unstructured information processing in the corporate research environment. *Nat. Lang. Eng.*, 10(3-4):327–348, September 2004.

[7] Yoav Freund and Robert Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In Paul Vitanyi, editor, *Computational Learning Theory*, volume 904 of *Lecture Notes in Computer Science*, pages 23–37. Springer Berlin Heidelberg, 1995.

[8] Nikhil Garg and Ingmar Weber. Personalized, interactive tag recommendation for flickr. In *Proceedings of the 2008 ACM conference on Recommender systems*, RecSys '08, pages 67–74, New York, NY, USA, 2008. ACM.

[9] Shantanu Godbole and Shourya Roy. Text classification, business intelligence, and interactivity: automating c-sat analysis for services industry. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 911–919, New York, NY, USA, 2008. ACM.

[10] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[11] Liangxiao Jiang, Zhihua Cai, Dianhong Wang, and Siwei Jiang. Survey of improving k-nearest-neighbor for classification. In *Fuzzy Systems and Knowledge Discovery, 2007. FSKD 2007. Fourth International Conference on*, volume 1, pages 679 –683, aug. 2007.

[12] Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. Multilabel Text Classification for Automated Tag Suggestion. In *Proceedings of the ECML/PKDD 2008 Discovery Challenge*, 2008.

[13] David Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In Claire Nédellec and Céline Rouveirol, editors, *Machine Learning: ECML-98*, volume 1398 of *Lecture Notes in Computer Science*, pages 4–15. Springer Berlin / Heidelberg, 1998. 10.1007/BFb0026666.

[14] M. and du Plessis. Drivers of knowledge management in the corporate environment. *International Journal of Information Management*, 25(3):193 – 202, 2005.

[15] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[16] Grigorios Tsoumakas, Eleftherios Spyromitros-Xioufis, Jozef Vilcek, and Ioannis Vlahavas. Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, 12:2411–2414, 2011.