DIGIMIMIR: A Tool for Rapid Situation Analysis of Helpdesk and Support E-mail

Nils Einar Eide, Andreas N. Blaafadt, Baard H. Rehn Johansen and Frode Eika Sandnes
– Oslo University College

ABSTRACT

The system administrators of large organizations often receive a large number of e-mails from its users and a substantial amount of effort is devoted to reading and responding to these e-mails. The content of these messages can range from trivial technical questions to complex problem reports. Often these queries can be classified into specific categories, for example reports of a file-system that is full or requests to change the toner in a particular printer. In this project we have experimented with text-mining techniques and developed a tool for automatically classifying user e-mail queries in real-time and pseudo-automatically responding to these requests. Our experimental evaluations suggest that one cannot completely rely on a totally automatic tool for sorting and responding to incoming e-mail. However, it can be a resource-saving compliment to an existing toolset that can increase the support efficiency and quality.

User Dynamics: Problem Statement

Reading and responding to e-mails from disgruntled users in an organization takes up several hours of a system administrator's daily effort. At the Engineering faculty at Oslo University College, with some 1,500 users, system administrators often encounter an inbox with hundreds of messages in the morning when arriving at work. The task of responding to these e-mail messages can be daunting, time-consuming and tedious. Yet, timely and quality replies are immensely important for the individual user in order for the users to fulfill their role in the organization. Technical difficulties and setbacks, that may seem trivial to a system administrator, can be overwhelmingly frustrating and destructive for the user and can be unnecessarily costly for the organization. Further, it is important that anomaly reports from users get the attention of the system administrators early such that corrective and preventive actions can be taken and to minimize the damage and the repercussions of the anomalies. It is therefore important that the system administrators' inboxes are continuously monitored.

Until very recently there have been few general formal training programs for system administrators with the exception of product specific certifications. Yet, the existing training and certification programmes primarily focus on technical issues. User support is usually learned on the job and there are often few official procedures for how to handle user queries. Attempts at ISO-certifying and standardizing procedures in organizations may result in some of these queries being formalized, but often it is superficial bureaucracy that, in terms of timeliness and responsiveness, may actually reduce quality rather than improving it. Consequently, system administrators often develop their individual practices for handling user queries, and often these ''common sense'' practices work quite well in practice.

Organizations are dependent on a team of system administrators and these administrators can usually be reached through one point-of-call, such as support@fantasticCorp.com. E-mails sent to such an address are commonly forwarded to the entire team of system administrators that are responsible for user queries. Two obvious reasons behind this strategy are that it is easier for the users to remember and the administrator on-duty will receive the message. However, often individuals in an organization know one or more of the system administrators personally and may decide to send a message to this specific system administrator directly. This is (for the reasons stated) an undesirable practice.

The size of the inbox is dependent on a number of factors. Two important factors are when the inbox was last inspected and the occurrence of certain system events. According to queuing theory, e-mails can be modeled as arriving in a stream with a Poisson distribution, and therefore the size of the inbox is approximately proportional to the time passed since it was last read. This phenomenon is something that everyone coming to work in the morning or returning from lunch, meeting or even a business trip can testify. Further, the occurrence of certain events, such as a system failure or anomaly usually ignites a burst of e-mails. For example a printer breakdown on Friday afternoon does not go down well with users struggling to meet their deadlines. Once a system failure or anomaly affects the work carried out by the user, the users often chose to resolve the problem by contacting the system administrators by e-mail, as it may be hard to reach the system administrator by phone. The more users an anomaly affects the more e-mails the system administrator

receives. Occurrences of anomalous events are hard to predict but they themselves may be modeled using a Poisson-like distribution.

The messages from users can be coarsely classified into four categories: a) automatically generated mails from various processes such as cfengine, at, cron, etc., b) unsolicited e-mail and spam, c) general questions and d) urgent questions, notifications and anomaly reports. Automatically generated mail is easily sorted into assigned folders using simple keyword-based filters, and automatic spam filters are getting increasingly good at reducing the amount of junk in the inbox. General questions from users, such as how to do ''this or that,'' are often not urgent and the task of responding to the question can be delayed to a ''low-peak'' time. Often, users ask similar questions and the system administrator can retrieve an old answer written to a different user at a previous point in time. In this way, the administrator hopefully saves time if this archived message can be recalled with little effort. However, urgent e-mails from users reporting anomalies, faults and strange behaviour in the computer system should be read and dealt with immediately. Many system administrators in small and medium-sized organizations get a ''feeling'' for how to read and respond to e-mail. The purpose of this work is to assist the system administrator by simplifying e-mail management. DIGIMIMIR, based on text-mining techniques, automatically clusters and categorizes incoming e-mail into related topics and presents the e-mail categories in terms of identifiable and characteristic keywords. System administrators get a clear overview of the inbox contents, and can thus more easily identify urgent matters that need to be resolved at high priority. In addition, DIGIMIMIR can be connected to a reservoir of pre-answered questions such that the most suitable answers to commonly asked questions are found automatically.

Previous Work

Much has been written on helpdesk support [8, 11, 13, 29] and many commercial systems exist, such as E11 and IssueTracker. These systems primarily focus on tracking historic aspects of customer support, maintenance of searchable knowledge bases and the identification of recurring issues. Many of these products target general businesses. GNATS is a system specifically designed for tracking bugs in software and the maintenance of these in databases [28].

Trouble ticketing systems, such as OTRS (Open Ticket Request System), are useful tools for managing large inboxes, which may be handled by several system administrators concurrently. New requests that arrive in the inbox are given a ticket (e.g., a number) and an automatic reply informing the user that the request is received and will be handled. Other requests on the same issue are automatically grouped together with the existing correspondence related to the ticket. The different system administrators therefore have easy access to the entire history of the requests associated with a ticket. The responses of different system administrators can therefore be coordinated.

Expert system based helpdesk systems have also been explored [35], where the staff running the helpdesk is guided through sequence of decision rules to solve the particular difficulties. A problem with expert systems is the nontrivial establishment of the decision rules. Another strategy is case based reasoning which is especially suited at detecting recurring problems [4]. In the spoken domain recurrent neural networks have been used to route helpdesk calls automatically based on utterances [9].

The difficulties of handling large numbers of requests are commonplace in large organizations. Research effort has gone into the automatic retrieval of answers from existing question-answer lists based on queries, such as the VIENA classroom system, which uses lexical similarity [32, 33], the FAQFinder system [1, 20] which uses semantic knowledge and the FAQIQ system which uses case based reasoning [22]. In a different approach [30] a template strategy is used to answer questions based on information in relational databases. Common to all these strategies is that they are based on already existing knowledge bases.

In addition to the distribution of answers it is also necessary to categorize questions. The idea of automatically sorting e-mail into predetermined categories has been examined by several researchers. In one theoretical study [36] web-mining agents are assessed as a means of automatically sorting e-mail using an uncertainty probability based sampling classifier and rough relation learning.

Recently, there has been a huge public interest in the problems associated with spam, and a substantial effort has gone into developing spam-filtering technology [1, 34]. Notably, by far the most efficient strategy is the statistically based nave Bayesian classification [34]. A nave Bayesian system is trained using a large corpus of spam e-mails and non-spam e-mails and a word signature vector is established for both groups. When a new message arrives, its word signature is compared to both that of the spam and the non-spam signatures and the one that yields the best match determines its category. The spam-filtering problem is related to the problem we are addressing in this paper. However, it is different on two major accounts. Firstly, spam-filtering entails classifying messages into two groups, either spam or non-spam. Second, these categories are defined a-priori. However, in document classification there are many (usually more than two) categories and the categories might not be known and therefore have to be established at run-time. Bayesian classification has in fact also been studied in the context of general document classification [24]. One exciting practical development that has occurred in

parallel with the development of DIGIMIMIR is POPFILE, which is a Bayesian based e-mail classification program [12]. POPFILE works directly on POP3 e-mail accounts and uses Bayesian classification to sort incoming messages into predetermined categories. POPFILE as a software product has reached some maturity, but still suffer from major shortcomings – the most notable (at time of writing) is the lack of IMAP support. The IMAP protocol is particularly applicable in the context of automatic mail sorting applications since mail folders can be managed on the server [5], and it is therefore possible to achieve e-mail client independence. Further, POPFILE is designed only to work in supervised mode and is reliant on training data and therefore cannot be deployed in unsupervised situations where new categories are created dynamically on the fly.

Another exciting and controversial new development is the announcement of Google's new e-mail service Gmail [10]. This is a novel strategy in terms of managing e-mail. The idea is that the users get a large area to store their mails and that e-mails rarely have to be deleted. Document classification techniques are therefore used to navigate and search through this huge reservoir of e-mails. This service is not yet available to the general public, but this new thinking with regards to dealing with e-mail management may prove to be useful in terms of support and helpdesk e-mail management.

In fact, the largest success of text mining and document classification and retrieval technologies has been within the areas of web search engines [21, 25] and search engine technologies have developed at a rapid pace over the last few years. However, there is a number of profound differences between the clustering and classification requirements for indexing web pages and handling streams of incoming e-mail. First, the largest difference is probably the volume of text. The web is huge and the size can be exploited to up the quality of the clustering and classification. Personal e-mail collections or an organization's e-mail collection remain small by comparison. Second, indexing of web pages can be done offline and there are no critical time-restrictions on how fast the pages are indexed. Indexing is often done offline and there is a significant turnaround time from when a change is being made on a document on the Internet until it is being reflected in the search results of the search engines. We could perhaps describe this as an index epoch. However, in order for an e-mail sorting system to have a value, the classification and clustering needs to be continuous, instant and real-time. Third, in the indexing of web pages quality is the prime importance, while in the clustering and classification of e-mail messages quality can be traded for speed. Until now, most of the research into text mining and document classification and retrieval primarily focus on clustering and classification quality and they pay less attention to the real-time operational constraints and the incrementally growing document collections (inboxes).

Finally, some system administrators wisely believe that the best support policy is self-reliance and that users should be able to resolve their own problems as much as possible [7] and hence reduce the need for support.

DIGIMIMIR and Text-mining

DIGIMIMIR employs techniques borrowed from web mining (see [3] for a general introduction to web mining) and terminology mining based on text corpora (see for example [15]). DIGIMIMIR takes a set of messages as input and produces a set of message clusters as output, i.e., related messages are clustered together, and unrelated messages are placed in different clusters. The step comprises several phases. First, the system must retrieve the messages, then each message is pre-processed and transformed into a text vector. The set of text vectors representing the set of messages are used as input to the clustering algorithm so as to compute the most suitable clusters and finally the results are presented to the recipient (user).

A dedicated e-mail account is set up and DIGIMIMIR polls the inbox at regular intervals to check for new incoming messages. The inbound messages are processed as follows: Each message is treated as a separate entity and used as a basis for computing a word vector. A word vector represents a set of words as a vector, where each word in a dictionary is assigned a specific position in the vector, thus the size of the vector equals the size of the dictionary used. The presence of a word is marked by a positive non-zero integer, where the value represents the number of times the word occurs in the text. A zero denotes the absence of a specific word. Cleary, different messages containing different words have different word vectors in the word space. Hence the phrases ''nuts and bolts'' and ''bolts and nuts'' would both yield the same word vector.

To generate a word vector the text is organized into individual words. The first step is to filter high frequency words, also known as stop words [31, 14]. This is achieved using a stop word dictionary computed from word frequency lists [17]. Next, word stemming is employed to obtain the general form of words [27] with the purpose to reduce the size of the word vectors. Then, a dictionary-based spell checking technique is used. I.e. a reference dictionary comprising of all the possible valid words in the language in all grammatical forms is used. If an entry in the text dictionary cannot be found in the reference wordlist then the entry is tagged as a potential incorrect spelling. All entries marked as potential incorrectly spelled words are crosschecked against the reference wordlist using Metaphone [26]. Metaphone is a technique, inspired by SOUNDEX for matching words based on their English phonetic sound and it is particularly suitable for spell checking applications. Entries with no Metaphone match in the dictionary are

considered special terms. Entries with a metaphone match are most likely incorrectly spelled words (see [19] for an excellent survey of automatic spelling correction techniques). Then, each instance of the remaining words is counted and represented in the word vector. The word vector therefore represents the potentially interesting words that are characteristic of the question [6]. Further, the word vectors can be large and contain large amounts of noise. Research into text-mining often strives to reduce the dimensionality, or size, of the word vectors by the means of some transformation [34]. In DIGIMIMIR a simple word vector reduction technique is employed, namely word masking. For each new vector that is being presented to the system only the words present in the given word vector are considered when computing the distance to the other words in the clusters. I.e. all words that are present in documents to be compared are discarded if they are not also present in the document they are compared to. This mechanism prevents unimportant, and most probably, unrelated words to influence the distance measure. Without dimensionality reduction, or word masking, then the auxiliary words may unnecessarily add to the distance between two word vectors that in practice are quite similar.

The word vectors are clustered using the K-means algorithm – a classic and widely known and effective clustering algorithm (see [6]). In clustering the words are represented as vectors in the word space, and the purpose of the clustering algorithm is to assign word vectors that are similar to the same cluster in the vector space and assign word vectors that are different to different clusters. Two vectors are similar if the distance between the two vectors is small, and two vectors are dissimilar if their distance is large. One popular distance measure is the Eucludean distance. The K-means algorithm works as follows: a set of vectors is to be clustered into K clusters. Initially, the vectors are assigned arbitrarily to the K clusters. Then the mean vector for each cluster is computed. Next, the vectors are reassigned to the cluster to which they are closest and the cluster means are recomputed. The process is repeated until some convergence criteria are met.

Finally, the results of the clustering algorithm are presented to the user as a pre-catalogued set of messages. Each time a new message or a group of messages arrives into the system, the process is repeated incorporating the new messages into the clusters.

Quality Measurements

It is relatively easy to assess the quality of classification tasks when they are applied to a training set, as this is a form of supervised learning, where the categories or clusters are predetermined. One can simply compute the success rate as the number of messages that are correctly assigned a cluster, and the error rate as the number of messages that are incorrectly assigned. However, assessing the quality of clustering is more difficult as there is no given mapping between the training category and the assigned cluster. We therefore deployed the following strategy: Each document d_i belongs to a training category c_i and after the algorithm is deployed it is assigned a cluster k_i. After training a category-to-cluster matrix is established where the columns represent the training categories and the rows represent the assigned clusters. An element at column i and row j denotes the number of documents from category c_i that has been assigned cluster k_j.

The quality measures are then computed in three steps. First, for each category the entire column is scanned for the largest value and this is marked as a category-to-cluster mapping. Second, for each cluster the entire row is scanned for the largest value which again is marked as a category-to-cluster mapping. If other elements in the row are also marked as a category-to-cluster mapping (in the first step) then these are remarked as ‘‘undecided.’’ Third, the three quantities are computed as follows: the success rate is computed by summing all elements marked as category-to-clusters and dividing by the total number of documents, the failure rate is the sum of all non-zero unmarked elements divided by the total number of documents and finally the ratio of ambiguous messages is the sum of all elements marked as ‘‘undecided’’ divided by the total number of documents.

Test Suites

In order to assess and document the effectiveness of the system through a repeatable experiment, one is dependent on a test suite with pre-categorized messages. The Reuters-21578 Text Categorization Collection [23] is a well-known and widely cited test suite, comprising of Reuters news articles from 1987 that have been classified and indexed by experts and later made available for research purposes. These news articles comprise medium to long, well written, pieces of English text. A news report is long compared to e-mail messages that often are short and poorly written with spelling mistakes and various abbreviations. The Reuters collection is therefore not completely representative of the problem domain. Further, we were unable to use this test suite as the current implementation of DIGIMIMIR is optimized for Norwegian. To manually categorize messages is time-consuming, difficult and error-prone. We therefore deployed three strategies for obtaining test-suites:

First, a small hand crafted test suite, comprising of 100 messages, was used for early testing. This set contains manually categorized fictitious messages, characterized as being easy to cluster and classify.

Second, the students at Oslo University College were used to create the second set of messages from an online ‘‘quiz,’’ comprising 160 messages. Four themes with 10 entries each were created. Each entry comprised a picture and a statement, such as a picture of well-known politicians or some hi-tech device. The

students were asked to submit a question related to the entry via a web form. Thus the questions received could therefore be tagged with the given category.

Third, a set of messages from our local UNIX system administrator was collected and manually organized into natural categories and labeled. Unfortunately, this test set was small and contained a diverse set of messages, since the system administrator limited the selection (out of concern for privacy and security) and considered the task low priority. Secondly, as this paper was finalized during late spring early summer, there was not that much traffic as the peak usually occurs in the autumn when new students enroll onto the college courses and acquire accounts on our computer system.

Implementation Details

The implementation consists of the following highly configurable Java components: Controller-module: Controls the flow of information through the system as the modules can be interconnected in an arbitrary manner, i.e., the output of one module can sent to the input of another module etc. Messages are processed as they travel through the various modules on their path to their destination. A typical message-path comprises an input-module, a set of pre-parsers, a parser-module, a set of filters, a sorter-module and an output-module. Multiple instances of a module can be created and used in different parts of the message chain.

The modules are glued together by the means of RMI (Remote Method Invocation) and a global configuration file, allowing various modules to reside on different machines in a distributed manner. This feature allows systems with a high degree of interactivity to be configured, i.e., the system can be configured to provide immediate feedback to instant inbound messages. A special message-path configuration is required, which does not contain conventional input and output modules. This implementation also allows modules to be interconnected to form a tree-structure where all non-leaf nodes are routers that forward messages to yet other modules. For instance, a controller can be configured to identify the language of a message and forward the message to the corresponding controller for the target language of that message. Messages can be rejected and re-routed to different branches of the graph if problems are detected by modules higher up in the tree.

Input-modules retrieve and convert external data into the internal representation used by the system. The current set of input modules can read POP3 (Post Office Protocol) and IMAP (Internet Mail Access Protocol) mailboxes, and external JDBC-compliant database tables provided the correct fields are appropriately configured.

Preparsers modify the incoming messages before they are tokenised into sentences and words. Removal of HTML-tags is a common pre-parser filter operation. Parsers tokenise texts into sentences and words. A message must be parsed before further processing can take place.

Filters alter or remove words or phrases from messages. For example, a spellchecker replaces incorrectly spelled words with their corresponding correctly spelled words, while a stop-word filter removes words that are not relevant to content of the message.

Sorter-modules attach answers to incoming messages, or signals to the controller that there are no matching answers to the incoming question.

Output-modules return messages back to their source provided they belong to the same category. For example, POP3 and IMAP input messages are returned via the SMTP (Simple Mail Transfer Protocol) output module. Further, the database input messages can be returned to another external JDBC-compliant database using references created by the input module.

GUI (Graphical User Interface)-controllers are equipped with swing-based GUI components which allows for direct user-controller-interaction. The current GUI controller can be connected to multiple controllers simultaneously. A HTML-based web interface is not currently provided, but is planned for a future release. The application is completely written in Java, currently using MySQL 3.23 for persistent storage via Hibernate. The application is developed and tested using Sun's JRE 1.4.2 (Java Runtime Environment).

Results

Figures 1, 2, 3, and 4 show the results obtained running DIGIMIMIR on two datasets. The figures illustrate the accumulative performance of the system, i.e., how the system state changes as messages are added to the system. The horizontal axes indicate the number of messages in the system and the vertical axes represent percentage correct answers, incorrect answers, ambiguous or uncertain answers and new clusters. Figure 1 shows the results obtained using the quiz dataset in unsupervised mode (160 messages), i.e., without training, and Figure 2 shows the results using the quiz dataset in supervised mode (80 messages), i.e., with training. One half of the dataset was used for training and the other half was used during testing. Figure 3 shows the handcrafted dataset in unsupervised mode and Figure 4 shows the hand-crafted dataset in supervised mode. The messages were shuffled into pseudo random order for all the test runs.

All the figures show similar trends, namely that the success rate, the error rate and the percentage of new clusters converge as more messages are added to the system – and this is adhering to expectations. The quiz data reveals a clear difference between the unsupervised and the supervised runs. In unsupervised mode we achieve a successful classification rate of nearly 50% and an error rate of just below 40%. The

percentage of ambiguous or uncertain messages is converging quickly to approximately 15%. Further, the probability percentage of generating a new cluster converges early at just over 10%. The results achieved in supervised mode are nearly twice as good. First, the success rate converges at around 80%, which is nearly a doubling in quality compared to unsupervised mode. This result is consistent with similar experiments reported in the literature on document classification. Second, the failure-rate is converging at 20%, which again is a halving in the number of incorrectly classified messages compared to unsupervised mode. Further, the rate of ambiguous or uncertain messages converges early at around 10%. It is also interesting to observe that the probability of generating a new cluster is converging much slower in supervised mode compared to unsupervised mode and that it is converging at a higher value of approximately 20% compared to 10% for unsupervised mode.

The shape of the graphs in the figures smoothly either decrease (error-rate and clustering probability) or increase (success-rate), but at certain points there

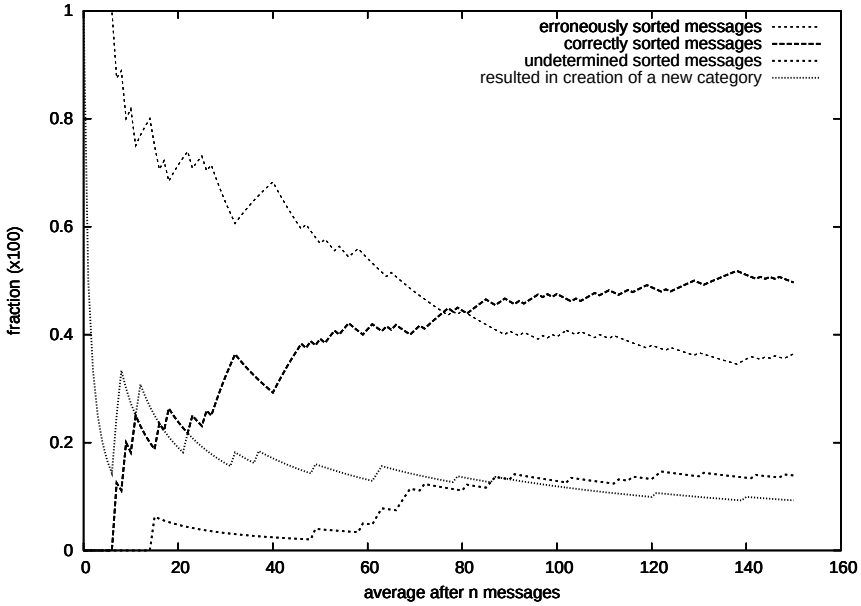

Figure 1: Cumulative success rates, error rates, uncertainty rates and clustering probabilities using the quiz test suite in unsupervised mode.

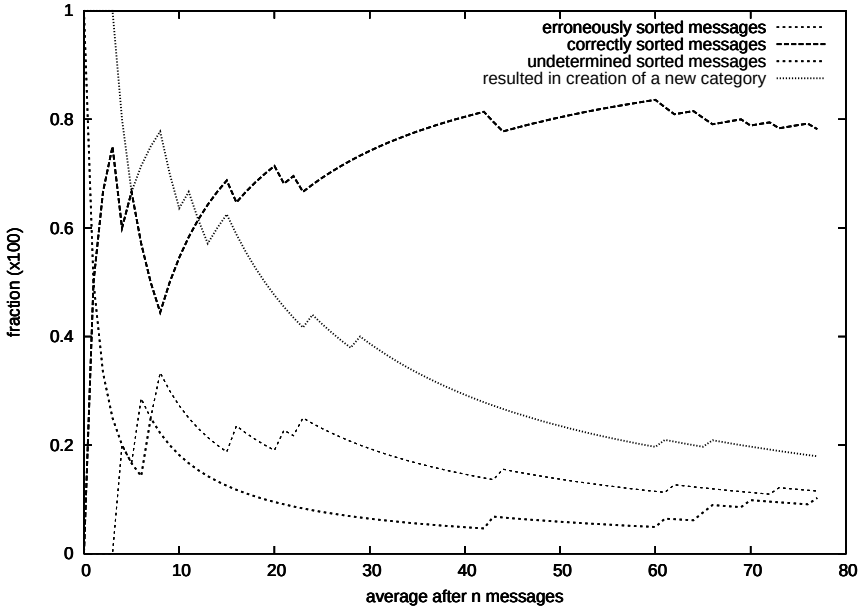

Figure 2: Cumulative success rates, error rates, uncertainty rates and clustering probabilities using the quiz test suite in supervised mode.

appear to be discontinuities in the graphs resulting in temporal setbacks. These discontinuities mark the arrival of very dissimilar messages that result in new clusters being established. When new clusters are added the classification landscape is altered and leads to a temporary classification instability and slightly lower success rates. These messages can be genuine and naturally belonging in new clusters or they may simply be irrelevant or noisy messages.

Similar observations can be made in Figures 3 and 4. Figure 3 shows the accumulative success, error and clustering rates for the custom-made test data in unsupervised mode with very high success rates, and Figure 4 shows the same dataset in supervised mode.

Operational Issues

It would seem natural to integrate DIGIMIMIR with a trouble ticket system such as OTRS. In fact, a trouble ticketing system could be a good front end to DIGIMIMIR. At the entry point all incoming messages are passed through a spam-filter, to remove noise in the input stream, and then the e-mails are

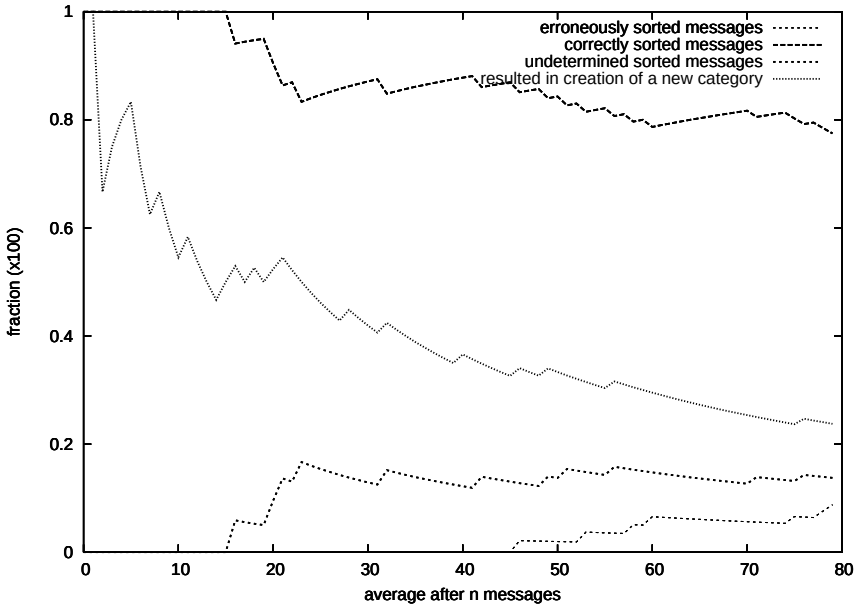

Figure 3: Cumulative success rates, error rates, uncertainty rates and clustering probabilities using the hand crafted test suite in unsupervised mode.

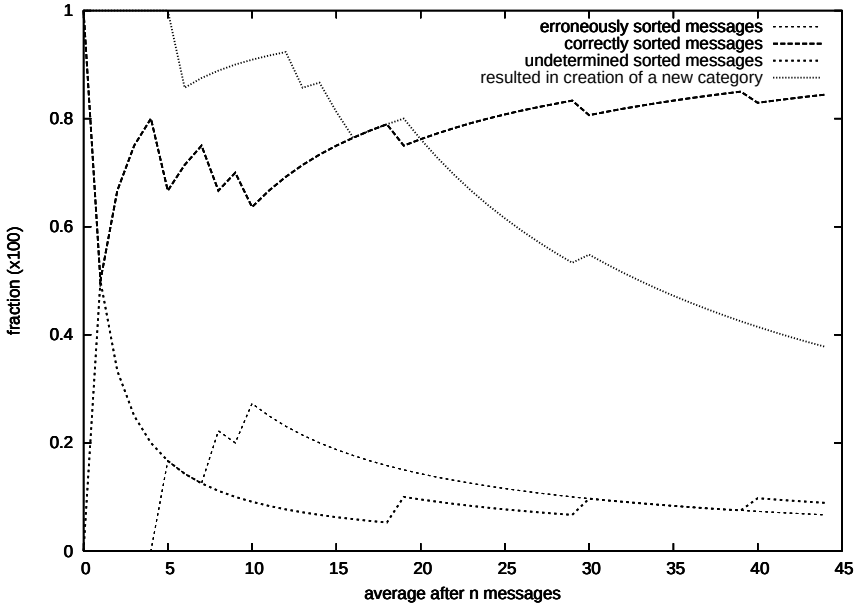

Figure 4: Cumulative success rates, error rates, uncertainty rates and clustering probabilities using the hand crafted quiz test suite in supervised mode.

directly handled by DIGIMIMIR. DIGIMIMIR would immediately classify the message and identify if there are any suitable categories, i.e., a match with a high probability of belonging to the particular cluster. If there is a good match then DIGIMIMIR would respond with a auto-reply if one exists in the system. If one of the tickets associated with a message in the cluster is assigned a response, then that response is also returned to the client. If there is no suitable response, or the probability is below a given threshold then the message is assigned a cluster and a standard ticket notice is returned to the user. These unprocessed messages are later addressed by a second line of system administrators. Each cluster is therefore analogous to a work queue, and once one general answer is generated for the messages in the cluster, all the recipients are forwarded the general reply. There should also be room to give specific and independent responses to a particular ticket in a queue and mark it as such. This would prevent a specific answer to be used as a general answer for automatic distribution. Further, is should also be easy to move a ticket manually to a different cluster if it is detected that DIGIMIMIR has misclassified the message. Further, it should be possible to manually establish new clusters and manually combine existing clusters that have been automatically established.

Future Work

There are many improvements that can be made to DIGIMIMIR. Firstly, the language specific modules need to be internationalized. An absolute medium addition is support for the English language. Ultimately, the tools should be easily extended to support any western language, such that it can be configured using only a standard wordlist for the language, for example the wordlists found on most Unix/linux-type systems, and a stop word list. These are all relatively easily accessible. The main challenge is the word stemming algorithms that must be tailor made for each language. Stemming algorithms have been published for most western languages, but one needs to know the language at least in order to evaluate the effectiveness of these algorithms. Another challenge is pictographic-based scripts such as Chinese. Chinese computer lingo is a good mix of Chinese characters and English terms, at least judging from Chinese Computer Magazines. There is a vast literature on Chinese language processing, and some answers may be found there. POPFILE claims to handle Chinese messages.

Further, we only had the opportunity to experiment with a few clustering techniques. Document classification is an ongoing research topic and better algorithms are continuously published. And the DIGIMIMIR tools will most certainly benefit from improved clustering and classification strategies.

A crucial next step of development is extended IMAP support such that the folders facilities on IMAP servers can be exploited. In addition to polling messages off the IMAP server, DIGIMIMIR should use the messages in the existing folders on the server during the classification and consequently move messages from the inbox to the correct folders on the server. When new clusters are established, DIGIMIMIR should create new folder on the server too. This will allow standard IMAP clients, such as Thunderbird, to be used as front ends to DIGIMIMIR and users would require no additional training. The users will see the new messages in the respective folders and the newly established folders. Further, by manually moving messages from one folder to another, the user can correct the classification engine and manually affect the classification and clustering. This would abolish the need for the DIGIMIMIR GUI component. However, we have had experiences with IMAP clients that do not register folder changes on the IMAP server that are made by other IMAP clients.

A crucial next step of development is extended IMAP support such that the folders facilities on IMAP servers can be exploited. In addition to polling messages off the IMAP server, DIGIMIMIR should use the messages in the existing folders on the server during the classification and consequently move messages from the inbox to the correct folders on the server. When new clusters are established, DIGIMIMIR should create new folder on the server too. This will allow standard IMAP clients, such as Thunderbird, to be used as front ends to DIGIMIMIR and users would require no additional training. The users will see the new messages in the respective folders and the newly established folders. Further, by manually moving messages from one folder to another, the user can correct the classification engine and manually affect the classification and clustering. This would abolish the need for the DIGIMIMIR GUI component. However, we have had experiences with IMAP clients that do not register folder changes on the IMAP server that are made by other IMAP clients. Another long-term improvement would be to introduce the idea of collaborative DIGIMIMIR systems, perhaps via a central body or directory, analogous to how virus filters update themselves on a regular basis. Many of the problems faced by system administrators are independently of organization, but rather dependent on a particular version of a product, for example a security patch for the apache web-server. As system administrators encounter problems and manually establishes categories for these problems, the information can then be shared with other DIGIMIMIR clients via one or more central word-vector reservoirs. Then, when other system administrators encounter similar problems at a later date, they can benefit from the work already carried out by the first system administrator and automatically obtain the new category, perhaps even with a suggested response. However, there are obvious privacy and quality issues that need to be addressed in order to deploy such a strategy.

Another, interesting possibility would be to integrate the system with a network monitoring and

alerting system such as IPSentry. Such system often has many channels of notification including e-mail. Obviously, a message from such a system is very easily detected by DIGIMIMIR and is correctly classified as such a system produces messages with uniform wording and format. More interestingly, notifications from network monitoring and alerting systems could be correlated with user messages. This could greatly help a system administrator more easily assess the impact of a given network anomaly.

Finally, DIGIMIMIR could benefit from a thorough review with regards to efficiency. Currently, the system has been tested with hundreds of messages with no apparent performance bottlenecks. However, large organization could easily receive a thousand messages each day, and maybe keep millions of messages on record. There are no apparent time or space complexity issues that prevent the system from scaling. The major bottleneck of the system is the k-means clustering algorithm, which has a linear time-complexity with respect to the number of messages ($O(c\ k\ d\ n)$, where k is the k-value, d is the number of dimensions for the documents, n is the number of documents and c is the number of iterations required) [16]. Further, the distributed nature of the DIGIMIMIR framework allows it to be configured to run in a distributed manner across a network of workstations, such that the inherent parallelism can be exploited.

Implications of Automated E-mail Support

The deployment of automatic document classification technology in the context of sorting incoming e-mails and automatically providing answers must be done with care. Nearly all requests are unique and the quality of the responses is best maintained by handling the requests manually. However, the quality of a support service is a tradeoff between the quality of the response content and its timeliness. A support department that does not respond to requests or the responses are answered weeks after they were originally sent can be frustrating to the users as they do not feel that their request is taken seriously. On the other hand, a rapid meaningless or obviously auto-generated incorrect reply can be equally frustrating and infuriating to the user and embarrassing to the organization. This may result in aggressive users. It is very difficult to make a foolproof system, i.e., a system that never incorrectly classifies a message and respond with the answer to a different question. To minimize the damage one can adopt psychological techniques, such as using humble wording in the responses. For instance, in DIGIMIMIR we used the following careful wording in the auto-generated replies: ''We found that your question X is very similar to question Y. One answer to question Y is Z. Note that this response is automatically generated and a human will evaluate this response hopefully quite soon.''

The optimal policy is probably a mixture of manual responses and automated responses. Manual responses should be used during low-peak periods for messages with a lower classification probability, while the automatic response mechanisms are to be used during peak hours when there is not sufficient manual resources to handle the stream of incoming messages. The administrator can then later inspect the requests that arrived during the peak time and assess the responses, and then send corrections to users when appropriate. Such a post-peak period inspection is still more efficient than having to respond to every message manually. It will in some situations be easier to spot a message that is out-of-place when it is placed together with other messages that are related. Ultimately, the inspection cycle is necessary as the 10 to 20% messages are incorrectly classified and needs to be handled manually. If one receives 500 messages a day, then this will account to 50 messages, and if the organization receives 5000 messages a day, this will obviously account to as much as 500 messages.

Politically speaking, an automated system is desirable as it is resource-saving. As long as the financial gains of deploying such a system are larger than the negative impacts of the errors introduced, an organization is likely to embrace such technology. A consequence of this is that in the next instance the system administrators may be budgeted with even fewer resources by the decision makers.

Availability

DIGIMIMIR is constantly under development and is released under a GPL license. Its binaries, source code and documentation can be downloaded from http://www.digimimir.org/.

Conclusions

In this paper we have addressed the problem of e-mail helpdesk support. Text-mining techniques were explored as means of partially automating the support tasks and a tool dedicated to this task was presented. Our experiments confirm that it is possible to achieve 50% or more correctly classified messages in unsupervised mode and 80% correctly classified messages in supervised mode. This can greatly help support staff reduce their workload, especially when combined with an auto-response feature. In operation, the system can exploit a mixture of supervised and unsupervised mode. Messages that are manually approved or classified messages can be used in a supervised manner, while new clusters can be established dynamically and unsupervised in order to get clear overview reports of totally new situations. We believe that document classification technology to a greater extent will be incorporated into the broad range of e-mail handling systems in the years to come due to the great potential for reducing the workload, but to ensure quality there should be a human in the loop. Further, such technology could also help reduce the emergency response time of a support team as emergency messages can be

more quickly identified and separated from less urgent requests.

Acknowledgments

The authors are grateful to the system administrator Sigmund Straumsnes of the Engineering faculty at Oslo University College for providing sysadmin related e-mails. Further, the authors are grateful to the staff and students at Oslo University College who participated in the quiz, and H-L Jian of National Cheng Kung University who provided some of the text mining papers. Finally, the authors acknowledge the useful comments of the anonymous reviewers and Jeffrey Sheldon of the Mathematics Department at Louisiana State University, who has greatly helped us improve the quality of this manuscript.

Author Information

Nils Einar Eide has recently graduated with a B.Sc. degree from the Computer Science Department at Oslo University College (2004).

Andreas Nordeng Blaafladt is an undergraduate student at Oslo University College. He is also a consultant with experience in Unix and network administration.

Baard H. Rehn Johansen is currently working towards a Masters degree at the University of Oslo, Department of Informatics. He received his B.Sc. from the Oslo University College in 2004. Reach him electronically at baard@rehn.no .

Frode Eika Sandnes received a B.Sc. in computer science from The University of Newcastle Upon Tyne, U. K. in 1993 and a Ph.D. in computer science from The University of Reading, U. K. in 1997. Dr. Sandnes research interests include parallel processing and especially multiprocessor taskgraph scheduling, human interaction with mobile devices, error correction codes and system administrations research. He is currently an Associate Professor at the Department of Computer Science at Oslo University College. Reach him electronically at frodes@iu.hio.no .

References

[1] Androutsopoulos, J. Koutsias, K. V. Chandrinos, G. Paliouras, and C. D. Spyropoulos, ‘‘An Evaluation of Naive Bayesian Anti-Spam Filtering,’ in *Proc. of the Workshop on Machine Learning in the New Information Age*, 2000.

[2] Burke, R., K. Hammond, V. Kulyukin, S. Lytinen, N. Tumuro and S. Schenberg, ‘‘Natural Language Processing in the FAQ Finder System: Results and Prospects,’’ in *Working Notes from AAAI Spring Symposium on NLP on the WWW*, pp. 17-26, 1997.

[3] Chakrabarti, S. *Mining the Web: Analysis of Hypertext and Semi Structured Data,* Morgan Kaufmann, 2002.

[4] Chang, K. H., R. Raman, W. H. Carlisle, and J. H. Cross, ‘‘A self-improving helpdesk service system using case-based reasoning techniques,’’ *Computers in Industry*, Vol. 30, Num. 2, pp. 113-125, 1996.

[5] Crispin, M., ‘‘Internet Message Access Protocol – Version 4rev1,’’ *RFC 3501*, http://www.imap.org/ .

[6] Dunham, M. H., ‘‘DATA MINING: Introductory and Advanced Topics,’’ pp 140-142, Prentice Hall, 2002.

[7] Elling, R. and M. Long, ‘‘User-setup: A System for Custom Configuration of User Environments, or Helping Users Help Themselves,’’ *Proceedings of the Sixth Systems Administration Conference (LISA VI)*, p. 215, USENIX Association, Berkeley, CA, 1992.

[8] Foo, S., S. C. Hui, and P. C. Leong, ‘‘Web-based intelligent Helpdesk-Support Environment,’’ *International Journal of Systems Science*, Vol. 33, Num. 6, pp. 389-402, 2002.

[9] Garfield, S. S. and Wermter, ‘‘Recurrent Neural Learning for Helpdesk Call Routing,’’ *Lecture Notes in Computer Science*, Num. 2415, pp. 296-302, 2002.

[10] Google.com, ‘‘Welcome to Gmail,’’ http://gmail.google.com/, 2004.

[11] Govindarajulu, C., ‘‘The status of helpdesk support,’’ *Communications of the ACM*, Vol. 45, Num. 1, pp. 97-100, 2002.

[12] Graham-Cumming, J., ‘‘POPFILE Trust the Octopus,’’ http://popfile.sourceforge.net/ .

[13] Guy, T., ‘‘Backstage at the Helpdesk,’’ *Proceedings of the 1999 ACM User Servive Conference,* ACM press, pp. 225-227, 1999.

[14] Ho, T. K., ‘‘Fast Identification of Stop Words for Font Learning and Keyword Spotting,’’ *Proceedings of the Fifthth Int’l Conference on Document Analysis and Recognition*, pp. 333-336, 1999.

[15] Justeson, J. & S. Katz, ‘‘Technical Terminology: Some Linguistic Properties and an Algorithm for Identification in Text,’’ *Natural Language Engineering*, Vol. 1, Num. 1, pp. 9-27, 1995.

[16] Kantabutra, S. and A. Couch, ‘‘Parallel K-Means Clustering Algorithms on NOWs,’’ *Nectec Technical Journal*, Vol. 1, Num. 6, pp. 243-248, Thailand, 2000.

[17] Kilgarriff, A., ‘‘Putting Frequencies in the Dictionary,’’ *International Journal of Lexicography*, Vol. 10, Num. 2, pp. 135-155, 1997.

[18] Kilgarriff, A., ‘‘Which Words are Particularly Characteristic of a Text? A Survey of Statistical Approaches,’’ *Proceedings of AISB Workshop on Language Engineering for Document Analysis and Recognition*, Sussex University, pp. 33-40, 1996.

[19] Kukich, K., ‘‘Techniques for Automatically Correcting Words in Text,’’ *ACM Computing Surveys*, Vol. 24, Num. 4, pp. 377-439, 1992.

[20] Kulyukin, V. A., K. J. Hammond, and R. D. Burke, ''An Interactive and Collaborative Approach To Answering Questions for an Organization, Technical report TR-97-14,'' Dept. Computer Science, University of Chicago, 1997.

[21] Lai, Y. S., K. A. Fung, and C-H. Wu, ''FAQ Mining via List Detection,'' *Proceedings of the COLING Post-Conference Workshop on Multilingual Summarization and Question Answering*, Taipei, Taiwan, 2002.

[22] Lenz, M. and H. D. Burkhard, ''CBR for Document Retrieval – The FAllQ Project,'' Case-Based Reasoning Research and Development (ICCBR-97), *Lecture Notes in Artificial Intelligence, Num. 1266*, pp. 84-93, Springer-Verlag, Berlin, 1997.

[23] Lewis, D. D., ''Reuters-21578 Text Categorization Test Collection Distribution 1.0,'' http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html, 1997.

[24] Nigam K., A. K. McCallum, S. Thrun, and T. Mitchell, ''Text Classification from Labeled and Unlabeled Documents using EM,'' *Machine Learning*, Vol. 39 Num. 2-3, pp. 103-134, May 2000.

[25] Moldovan, D. and A. Novischi, ''Lexical Chains for Question Answering,'' *Proceedings of COLING 2002 International Conference on Computational Linguistics*, Taipei, Taiwan, 2002.

[26] Phillips, L., ''Hanging on the Metaphone,'' *Computer Language*, Vol. 7, Num. 12, pp. 39-43, 1990.

[27] Porter, M. F., ''An Algorithm for Suffix Stripping,'' *Program*, Vol. 14, Num. 3, pp. 130-137, 1980.

[28] Salvadori, L., ''GNATS Revisited,'' *Sys Admin: The Journal for UNIX Systems Administrators*, Vol. 6, Num. 8, pp. 53-55, 1997.

[29] Shalhoup, R., ''A Web-Based Helpdesk,'' *Sys Admin: The Journal for UNIX Systems Administrators*, Vol. 6, Num. 9, pp. 41, 42, 44-46, 1997.

[30] Sneiders, E., *Automated Questioning Answering: Template Based Approach*, Ph.D. Thesis, Dept. Computer and System Science, Stockholm University, 2002.

[31] Wilbur, J. W. and K. Sirotkin, ''The Automatic Identification of Stop Words,'' *Journal of the American Society for Information Science*, Vol. 18, pp. 45-55, 1992.

[32] Winwarter, W., ''Collaborative Hypermedia Education with the VIENA Classroom System,'' *Proc. First Australasian Conf. on Computer Science Education, ACSE'96*, pp. 337-343, 1996.

[33] Winwarter, W., O. Kagawa, and Y. Kambayashi, ''Applying Language Engineering Techniques to the Question Support Facilities in VIENA Classroom,'' *Database and Expert Systems Applications,* pp. 613-622, 1996.

[34] Yerazunis, W. S., ''The Spam-Filtering Accuracy Plateau at 99.9% Accuracy and How to Get Past It,'' *Proceedings of the 2004 MIT Spam Conference*, January, 2004.

[35] Zhao, M., C. Leckie, and C. Rowles, ''An Interactive Fault Diagnosis Expert System for a Helpdesk Application,'' *Expert Systems*, Vol. 13, Num. 3, 203-217, 1996.

[36] Zhong, N., T. Matunaga, and C. N. Liu, ''A Text Mining Agents Based Architecture for Personal E-mail Filtering and Management,'' *Intelligent Data Engineering and Automated Learning – IDEAL 2002*, Lectures Notes in Computer Science 2412, pp. 329-336, 2002.