Dynamic Email Organization via Relevance Categories

May 6, 1999

Kenrick Mock

Kenrick.J.Mock@intel.com

Intel Architecture Labs

Abstract

Many researchers have proposed classification systems that automatically classify email in order to reduce information overload. However, none of these systems are in use today. This paper examines some of the problems with classification technologies and proposes Relevance Categories as a method to avoid some of these problems. In particular, the dynamic nature of email categories, the cognitive overhead required to train categories, and the high costs of classification errors are hurdles for many classification algorithms. Relevance Categories avoid some of these problems through their simplicity; they are merely relevance-ranked lists of email messages that are similar to a set of query messages. By displaying messages as dynamic query results in lieu of fixed categories, we hypothesize that users will be less sensitive to errors in the Relevance Categories scheme than to errors in a fixed categorization scheme. To study the effectiveness of the Relevance Categories concept, we devised a performance metric for relevance ranking and used it to test an inverted index implementation on the Reuters-21578 test collection. The promising test results indicate the need for further work.

1. Introduction

Electronic mail and information overload has become a significant problem over the last several years. Time Magazine estimated that 776 billion email messages were sent in 1994, 2.6 trillion sent in 1997, and 6.6 trillion email messages will be sent in 2000 [1]. Today, it is not uncommon for users to receive hundreds of messages per day. To address this problem, many researchers have designed systems to automatically classify incoming email. Typically, the email is classified into folders. The folder hierarchy is usually flat and distinct; i.e. a message cannot belong to two folders, and the content of a folder is independent from the content of another folder.

2. Previous Work

Existing research has focused on a variety of learning algorithms to classify email into folders. First, the user is required to designate a set of messages that belongs in the folder. These messages are used as positive training examples for the classifier's learning algorithm. The user may also be required to specify messages that do not belong in the folder, i.e. negative training examples. Depending on the algorithms that are employed, the training process may be compute intensive. After the classifier is trained, new or existing email may then be evaluated through the classifier and placed into the folder if appropriate.

A commonly deployed email classification learning algorithm is based on vectors of term-frequency / inverse-document-frequency (tf-idf) values. These values are used to create a vector that represents both email messages and the contents of a folder [2,8]. Email vectors and folder vectors can then be compared to one another through the cosine metric or a dot product. An email message is classified into the folder whose vector most closely matches the vector for the message. Note that this system only allows for classification into a single folder. To support classification into multiple folders, which we will refer to as *categories*, a threshold value must be computed for each category. If the vector comparison exceeds the threshold, then the message is placed into the category. Unfortunately the computation of the threshold values is non-trivial and an open research issue.

In addition to tf-idf vector-based systems, many other learning algorithms have been investigated, ranging from the induction of decision rules [3,6] to Bayesian classifiers [4], support vector machines [5], and neural-network, case-based, or knowledge-based approaches

[9]. Some of these approaches are more expressive than others; for example, multi-layer neural networks are capable of non-linear classifications, unlike a naïve Bayesian classifier. Typically, the tradeoff for this flexibility is a dramatic increase in the computation required to train the classifier. All of these approaches have been shown to work reasonably well for fixed categories of email, such as finding mail belonging to a mail list, or mail that may be considered junk mail.

3. Difficulties with Email Classification

With the vast amount of interest and research that has been accomplished with automatic email categorization, why hasn't the concept been incorporated into existing mail readers? Does the concept fail to work in practice? To investigate the issue, we developed prototype email plug-in's based on tf-idf and rule induction classifiers for two popular email clients. The plug-in's are capable of classifying email into multiple categories as opposed to single folders.

Some of the ad-hoc usability obstacles that we encountered with respect to the classification technology are:

1. The need for constant re-training to keep up with dynamically changing categories.
2. Classification errors are puzzling and instill distrust on behalf of the users.
3. Insufficient data may be available as training examples.
4. It is difficult for a user to examine or manually edit a classifier.

Dynamic Nature of Categories

The first issue addresses the dynamic nature of email. It is not uncommon for a category to change over time as new messages are received. As with newsgroups, there may be "topic drift" as new threads are incorporated and added to a folder.

Topic drift poses a significant problem for many learning algorithms, which typically perform better on for static data. First, training time is often an issue. For example, inducting new rules or a decision tree may take many minutes, an unacceptable delay if this is required every time a category is changed. Second, most of these algorithms do not learn incrementally, and updates require a complete re-train based upon the original training messages. Vector-based learning algorithms are relatively quick to update, but thresholds may take more time to re-compute. Finally, fixed-length vectors have a vocabulary problem if the vector doesn't include new keywords that are necessary to properly learn the new meaning of the category.

Classification Errors and Trust

The second issue addresses common user expectations with regard to classification accuracy. Users tolerate relatively few errors and expect immediate results. Unfortunately, no classifier will be completely accurate and the re-training issues may prevent immediate results.

For example, consider a classifier that over-generalizes a category. If the user applies this classifier to all mail, the result may be a large number of messages put into the category that do not belong there. The user effort required to fix the error and re-train the category might outweigh the utility of the classifier to the point where a regular folder is less work. Additionally, user trust in the system will be severely impaired by these types of mistakes. A similar problem occurs with classifiers that are too specific and miss messages that should be included.

As another example, consider a vector-based classifier that fails to classify a new message because it is an outlier. Traditional vector-based classifiers are linear classifiers that classify messages close to its centroid. The user now uses the new message to train the classifier. As a result, the centroid for the category vector shifts slightly towards the vector of the new article. However, a single article is probably not enough to dramatically alter the behavior of the classifier[1]. Now, if the user receives another email almost identical to the previous email, it will probably not be classified into the category despite the user's efforts. The result is frustration on behalf of the user and a lack of trust in how the classifier works. Although the system will likely perform the correct behavior after a few more emails are received, our observation has been that users are extremely intolerant of these errors and expect an immediate correction after re-training with the new message. The more common response is "Why is the machine broken?"

Insufficient Data Available

The third issue addresses the need for large amounts of data in order to generate meaningful classifiers. Most of the learning algorithms are based on statistics, and

[1] Other classifiers, e.g. nearest-neighbor, will perform the desired behavior in this case, but may not generalize as well as the vector-based classifier or other classifiers.

for the algorithms to perform well, a large amount of data must be available.

Unfortunately, in many cases a large amount of email may not be available. For example, a common expectation is the capability to build a classifier using about a dozen training examples. This may be insufficient to generate meaningful and accurate classifiers, further exacerbating the classification inaccuracy of many statistical-based algorithms. Nevertheless, users expect the system to work even with a limited amount of email.

Classifier Viewing and Editing

Ideally, a classifier would be good enough so that a user will never have to manually fix or edit it. In practice, users may want to understand why the classifier is behaving in a particular manner and to perhaps alter its behavior. For example, with the rule-based classifier, the first question that many users ask after training is, "Can I see the rules?" Rule-based classification algorithms may be understood and modified by users relatively easily, but Vector or Bayesian classifiers may be extremely difficult for a user to comprehend, let alone edit. At a minimum, if users have a way to understand the underlying model and behavior of a classifier, then additional trust may be earned by the system.

4. Solutions to Classification Difficulties

Many of the difficulties described with classification may be alleviated through better classifiers, new classifier technology, and additional user-interface constructs to keep the user informed as to the state of the classifiers. We are currently investigating these techniques as to their feasibility and effectiveness.

Another way to resolve these difficulties is to sidestep the entire problem with an alternate technology. The remainder of this paper discusses one alternate technology, Relevance Categories, that addresses some of the same information management issues as automatic classification while avoiding many of the problems discussed in the previous section.

5. Relevance Categories

The difficulties explored in section 3 bring up several requirements for any proposed solution. First, the technology must be fast and capable of almost instantaneous re-training. Second, the system must either make few errors or operate in such a way that errors do not significantly impair usage. Third, the method must operate satisfactorily even when there is not much data available.

Relevance Categories are a simple way to address these three issues. The basic concept is to provide the same functionality as regular folders or categories. Users can assign mail to categories, or remove them from categories just like they are normally used to. The new addition to Relevance Categories is a query that is performed across all mail messages based upon the items the user has placed into the category. The results of the query are shown as a relevance-ranked list in a separate window or frame.

This method results in an approximation of categories and gives the user a way to store persistent queries into the database of mail. If the relevance algorithm is perfect, then all relevant items that belong to the category will be listed first, and less relevant items listed last. Thread, date, sender, or other fields could be used to sort the relevant items to increase their accessibility. The advantages of this approach to traditional classification are:

1. Relevance ranking can be performed quickly, with no training. This bypasses the problems that classifiers have with regard to training time and the necessary user overhead to fix classification errors. Since there is no classifier, there are no centroids to move or over-generalizations to worry about.

2. Errors are more likely to be tolerated by the user. As long as items relevant to the category are ranked near the top, the user will be able to find them. A few false hits will increase the amount of noise, but not significantly as long as the false hits are mostly ranked below the relevant items. This is similar to the behavior of search engines. They may produce some false hits, but the results are still immensely useful as long as relevant items are near the front of the list.

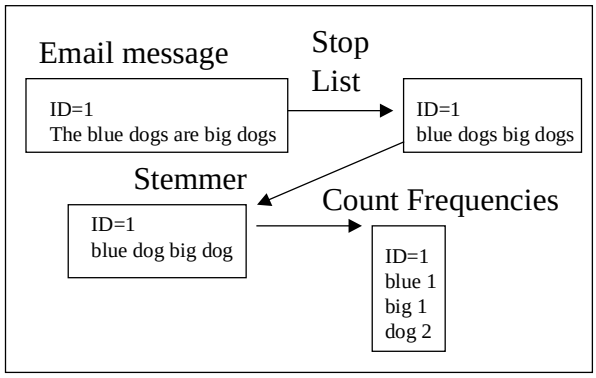
3. Relevance Categories are guaranteed to preserve the contents of existing categories and folders. Since the Relevance Categories are an add-on to existing categories, they could be ignored and used exactly like a normal category without impacting performance. In contrast, a poorly performing classifier can potentially make a folder more difficult to use than if no classification was done at all.

4. Relevance rankings are still possible even in the presence of sparse data. As more data becomes available, the relevance should improve.

Implementation of Relevance Categories

Relevance Categories could be implemented through any means as long as the constraints of section 5 are met. We constructed a very simple implementation based upon an inverted index with integrated tf-idf values. The inverted index provides quick access to potentially relevant messages while the tf-idf values provide the relevance metric.

The first step in the implementation of the inverted index is to parse, stop-list, and stem each email message. Each message is then treated as a bag of



words, and the frequencies are counted for all remaining words. The top N words, or terms, are saved as a representation of the message. In these experiments, N was set to 50. The process is depicted in figure 1.

Figure 1. Stop-listing, Stemming, Frequency Counting

The next step is to index each message into the inverted index. The architecture of the inverted index is shown in figure 2.

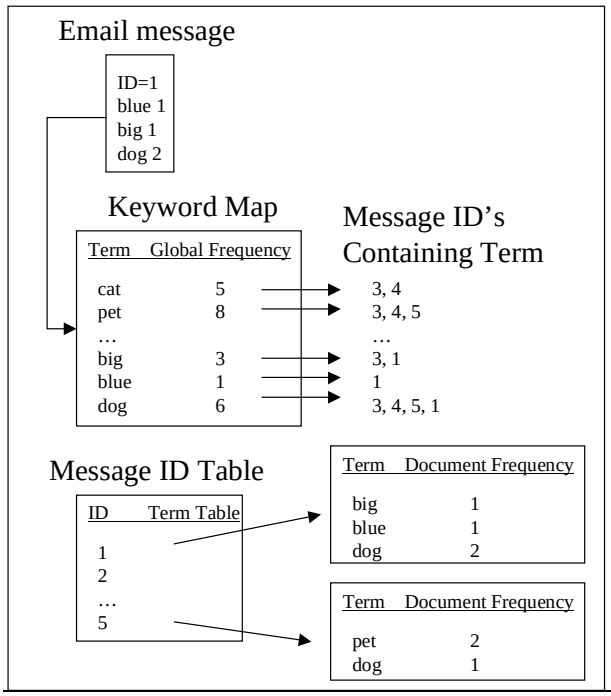


Figure 2. Inverted Index Architecture.

The frequency for each term in a message is collected in a global keyword map. This map is accessible by keyword. A pointer for each entry leads to a list of all messages that contain that term. This provides quick access to all messages that contain a particular term. In addition to the keyword map, there is also a message ID table. This table stores all extracted terms for each message along with their document frequencies. These values can be combined with the global frequencies to obtain tf-idf coefficients. Updating the data structures to maintain the inverted index requires only O(n) time, where n is the number of extracted terms.

Messages are retrieved from the inverted index based upon a query. A query consists of a set of terms and their associated frequencies. This query could be from an individual message or aggregated from a set of messages. From the inverted index keyword map, the set of documents that contains any terms in the query is determined. This has the potential to substantially cull the list of relevant messages so that all messages do not need to be examined. Then, each message document is compared to the query using a similarity metric. In this experiment we used the Dice coefficient, which is given by:

$$Sim(Query, Doc) = \frac{2\sum_{i=1}^{\#Terms} TfIdf(Query(i)) \bullet TfIdf(Doc(i))}{\sum_{i=1}^{\#QueryTerms} TfIdf(Query(i))^2 + \sum_{i=1}^{\#DocTerms} TfIdf(Doc(i))^2}$$

The Dice coefficient returns a normalized value between 0 and 1, where 1 indicates exact similarity, and 0 indicates no similarity.

Inverted Index Usage

After all messages have been indexed, the next step is to create queries. Queries are created for each category and are based upon messages that the user places into the category. The messages are concatenated and treated like a single document. Then, the N most frequent terms and term frequencies are extracted. In these experiments, N was set to 50. The resulting terms comprise a query for the category that it represents. Note that as the set of messages changes, the queries are simple to update. All that is required is to re-compute the term frequencies.

Once the query terms and frequencies are determined, messages are evaluated using the inverted index and the Dice coefficient. The messages are sorted by relevance value and displayed to the user in a list. The user can then browse through the list and open messages of

interest. If the algorithm works properly, the mail most similar to the items placed in the folder will appear at the top of the list. The entire process is very quick; our implementation for a thousand messages required only a few seconds to compute on an Intel® Pentium® II processor-based system.

6. Evaluation of the Algorithm

To examine the effectiveness of the Relevance Categories concept, we conducted a test using the Reuters*-21578 corpus.

The Reuters-21578 collection is a corpus of Reuters news articles originally published in 1987. The content of the 21,578 articles ranges from economic to agricultural news. A majority of the articles are classified into one or more of 135 categories. However, the classifications are not perfect; there are some well-known ambiguities and inconsistencies. The entire corpus was cleaned up and annotated by David Lewis in 1997 as a standard test collection so that machine learning algorithms could be fairly compared to each other on this corpus [7]. This corpus is particularly challenging for classification since there are multiple non-overlapping and non-exhaustive categories.

A number of researchers have used different splits of the data for the training and test sets. For our experiments we used the "ModApte" split. The split consists of 7,775 training articles and 3,299 test articles. The number of training articles is slightly lower than Lewis' numbers since we threw out those training articles that had no assigned category topic. Some of the test articles also have no category topics, but these were not thrown out.

Experimental Methodology

Since the Reuters-21578 collection was designed for classification tasks and Relevance Categories are designed to provide ranked lists of documents, the standard evaluation metrics of precision and recall do not apply to this task. To evaluate the effectiveness of Relevance Categories, we used a new metric, the Goodwin Relevancy Metric (GRM), named after its inventor.

In the GRM, we start with a ranked list of relevant messages for a particular category. From the tags in the test data, we know which of these messages belong to the category. These messages are sometimes referred to as "classified test messages." The ideal organization of the relevancy list is defined as the case when all categorized test messages are located sequentially at the top of the list, as shown in figure 3 for the sample category "Cocoa". In this example, *K*=3 indicates the number of classified test messages in the category, while *N*=5 indicates the total number of messages returned in the Relevance Category.

Ideal Ranking For Category Cocoa
K=3, N=5

Rank	Actually in Cocoa	Doc ID
1	Yes	54
2	Yes	43
3	Yes	33
4	No	21
5	No	12

Figure 3. Ideal Ranking for Cocoa.

In contrast, the worst possible ordering for the rankings is if all of the messages that belong to the category are ranked at the bottom of the list. This case is illustrated in figure 4.

Worst Ranking For Category Cocoa
K=3, N=5

Rank	Actually in Cocoa	Doc ID
1	No	21
2	No	12
3	Yes	54
4	Yes	43
5	Yes	33

Figure 4. Undesirable Ranking for Cocoa.

In practice, we are more likely to have a case in between the best and worst possibilities. Such a case is depicted in figure 5.

* Third party marks and brands are the property of their respective owner.

Actual Ranking
for Category Cocoa
K=3, N=5

Rank	Actually in Cocoa	Doc ID
1	Yes	54
2	Yes	43
3	No	21
4	Yes	33
5	No	12

Figure 5. Typical Category Ranking

We can measure how close the actual ranking is to the desired ranking by scaling sums of the rankings in the best and the worst cases. The resulting expression is the GRM value. In this expression, *N* is the total number of messages in the list, *K* is the number of classified test messages that are actually in the category, and R(*i*) is a function that returns 0 if message *i* is not in the category, and *i* if message *i* is in the category.

(Eq. 1) $$GRM = \frac{\sum_{i=n-k+1}^{n} i - \sum_{i=1}^{n} R(i)}{\sum_{i=n-k+1}^{n} i - \sum_{i=1}^{k} i}$$

The denominator subtracts the best possible rank from the worst possible rank, while the numerator subtracts the actual rank of the messages in the category from the worst possible rank. A perfect ranking results in a value of 1, while the worst possible ranking results in a value of 0. The metric scales linearly for cases in between; for example, if all messages are in the middle, GRM=0.5. In the example of figure 5, GRM = (12-7)/(12-6) = 0.83.

While the GRM is effective, it may lead to misleading results depending on the data. With the Reuters data, many categories have only a handful of classified test messages for a category but thousands of messages that do not belong to the category. For example, if *N*=3299 and *K*=1, the single classified test message could be ranked as low as 300 and still result in a high GRM of 0.9. The large number of non-classified messages results in a skewed evaluation.

To address this problem, we truncated the list of top relevant messages to a truncation threshold, *T*. In the experiment, we set *T* to 100. This threshold was determined based upon the number of messages in the list that a user might be willing to scroll through. We estimated that a user may feasibly browse the first 100 messages, but is unlikely to expend the effort to look further. In practice, the number may be lower. In any event, a user will certainly not scan through thousands of messages.

The act of truncating the relevance list complicates the GRM computation. The following cases must now be addressed:

1. *K*, the number of messages that are in the category, is greater than the truncation threshold, *T*.
2. *K* is less than the truncation threshold, but the number of messages ranked by the relevance algorithm within the first *T* messages may be less than *K*.

The first case is handled by setting the optimal ranking to include all classified messages for the first *T* messages. The modified GRM value is calculated via:

$$GRM_1 = \frac{\sum_{i=1}^{T} (T - R(i) + 1)}{\sum_{i=1}^{T} i}$$

The modified metric penalizes the algorithm for missing any rankings in the top *T* with a higher penalty for top rankings vs. bottom rankings.

The second case may cause normalization problems if it is not accounted for. To scale the metric appropriately, the following equation is used. In this expression, f is the number of classified messages found by the relevance algorithm within the first *T* messages and *K* is still the total number of classified messages in the category:

$$GRM_2 = \frac{\sum_{i=T-f+1}^{T} i - \sum_{i=1}^{T} R(i)}{\sum_{i=T-f+1}^{T} i - \sum_{i=1}^{f} i} \bullet \frac{f}{K}$$

If *f=K* then this expression is identical to equation 1. Otherwise, the GRM value is scaled down proportionally by the number of messages that were actually found vs. the number of messages that would ideally have been found. The resulting value ranges between 0 and 1.

Experimental Procedure and Results

To test the system, the 7775 Reuters training articles were used to generate term frequencies for each category and the 3299 test articles indexed. The relevancy rankings for each category were generated and the GRM metric computed. Categories that

contained no classified test messages were ignored; this left 90 total categories.

A summary of the GRM results averaged across all categories is shown in Table 1.

Mean	0.78
Median	0.80
Standard deviation	0.18

Table 1. Summary of GRM results averaged over 90 categories.

The results indicate good, although not stellar, performance. On average, the classified test messages definitely appeared toward the top of the relevance list.

To interpret the results, consider a relevance ranking that returns 100 messages, K=1, and has a GRM=0.8. The message that belongs to the category will be ranked 20th in the list. For K=2 and GRM=0.8, the two messages that belong to the category will be spaced around the 20th rank in the list. For example, either both messages are 19th and then 20th, or one may be 1st and the other 40th. While this is less than ideal, it may be adequate to provide sufficient utility in finding related articles and is far superior to scanning all mail messages.

7. Conclusion and Future Work

While this paper has focused on email, the problems and solutions discussed are equally applicable to other dynamic domains such as news stories or web pages. The goal of the work has been to examine hurdles in the space of automatic classification algorithms with respect to common applications, and discover ways that these hurdles may be overcome.

The concept of Relevance Categories is really a step back from pure categorization. Unfortunately, existing algorithms for classification may require too much training time for dynamically changing categories and produce too many errors that break user trust. Until these issues can be resolved, an alternate approach is to use a different technology that avoids some of these issues. Relevance ranked lists appears to be one such candidate. The ranked lists are quick to compute and errors, while a hindrance to productivity, do not produce the same consequences with respect to folder pollution and re-training that is necessary with traditional categorization. Moreover, ranked lists may be easily generated for multiple or overlapping categories.

To further validate the approach, the next step is to build and integrate Relevance Categories into an email application and to conduct user studies. Additionally, more work can be done to produce better rankings. For example, better term selection, noun phrase extraction, the use of more terms, variation of test parameters and assumptions, and different similarity metrics might significantly improve relevancy performance. Visualizations can also be constructed that are superior to the simple list view. For example, a 3-D visualization might take advantage of threads combined with relevance values to quickly depict the contents of a category. Finally, additional work is required to quantify the performance of current classification algorithms in the email domain with both test data and user studies. When user expectations are closely aligned with the capabilities of the underlying technology, information agents that organize and classify streams of data will become more effective and widespread.

This work has been possible thanks to the contributions from the following people: David Goodwin, Dhan Keskar, Brian Bird, Alan McConkie, Robert Adams, Dave Atkinson, and Mic Bowman.

8. References

[1] Gwynne, S. and Dickerson, J. Lost In The E-Mail. *Time Magazine*, April 21, 1997.

[2] Boone, G. Concept Features in Re: Agent, an Intelligent Email Agent. *Proceedings of the Second International Conference on Autonomous Agents*. Minneapolis/St. Paul, May 10-13, 1998.

[3] Cohen, W. Learning rules that classify e-mail. *Proceedings of the 1996 AAAI Spring Symposium on Machine Learning in Information Access.* 1996.

[4] Sahami, M., Dumais, S., Heckerman, D., Horvitz, E. A Bayesian approach to filtering junk e-mail. AAAI'98 Workshop on Learning for Text Categorization, Madison, WI, July 1998.

[5] Dumais, S., Platt, J., Heckerman, D., Sahami, M. Inductive learning algorithms and representations for text categorization. *Proceedings of ACM-CIKM98*, November, 1998.

[6] Apte, C., Damerau, F., & Weiss, S. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12, 233-251. 1994.

[7] Lewis, D. The Reuters-21578 Test Collection, Distribution 1.0. http://www.research.att.com/~lewis/reuters21578.html 1997.

[8] Mock, K., Adams, R., Spangler, L. Venice: Content-Based Information Management for Electronic Mail. 1997 Intel Software Developers Conference, Portland, Oregon. 1997.

[9] Mock, K & Vemuri, V. Information Filtering via Hybrid Techniques. *Journal of Information Processing and Management*, Permagon Press, v33, n5, pp 633-644. 1997.