# Email Classification with Temporal Features

Svetlana Kiritchenko[1], Stan Matwin[1], and Suhayya Abu-Hakima[2]

[1] School of Information Technology and Engineering, University of Ottawa, Ottawa, ON, Canada K1N 6N5

[2] AmikaNow! Corporation, 700 March Road, Suite 203, Kanata, Ontario, Canada K2K 2V9

**Abstract.** We propose a novel solution to the email classification problem: the integration of temporal information with the traditional content-based classification approaches. We discover temporal relations in an email sequence in the form of temporal sequential patterns and embed the discovered information into content-based learning methods. The new heterogeneous classification system shows a good performance reducing the classification error by up to 22%.

## 1 Introduction

As estimated by Ferris Research [5], spam accounts for 15% to 20% of inbound email at U.S.-based corporate organizations. Half of users receive 10 or more spam messages per day while some of them can get up to several hundreds unsolicited emails. Spam is not only distractive for end users, it also costs corporations a great deal of money. However, spam is not the only problem with emails. The overwhelming amount of electronic correspondence has become very difficult to manage for many users. One could benefit from a personal email managing system that filters unsolicited messages, forwards, deletes, or saves messages into relevant folders, prioritizes incoming messages and forwards the urgent ones to mobile devices. At the company level, an automatic assistant that could forward messages to appropriate staff members or even automatically respond to general inquiries would be a considerable advantage. At the same time, the automatic systems have to make decisions very accurately without active interference from the user.

In this work we propose a novel solution to the email classification problem: incorporating temporal information in the traditional classification approaches. A lot of research has been recently carried out on email classification proposing general and specific solutions to some of the problems mentioned above. Generally, those approaches concentrate on content interpretation and work only with content-based features. At the same time, email data have a temporal character: every email message has a timestamp. Therefore, we have a sequence of items each of which has a timestamp and content. This temporal information is always present, but usually ignored by researchers. We would like to explore the potential relevance of temporal information to common tasks in the email domain. The basic idea is to extract useful features from the temporal information present in the email data and combine these new features with the conventional content-based classification approaches. Our belief is that since we work with a much richer information space, we could expect a significant improvement in the classification accuracy.

This paper presents a comprehensive study of employing temporal information in email classification. Various approaches have been implemented and tested on several email and newsgroups corpora. The results suggest that the proposed approach can improve the existing content-based techniques to a moderate degree.

The reminder of the paper is organized as follows. In the next section we give an overview of related work. Then, we start with some experiments on adding in time in the form of simple features. Next, we introduce our idea of representing temporal information in the form of temporal patterns. We give the definition of a temporal pattern, present an algorithm for discovering temporal patterns in a sequence, and describe several approaches to integrate temporal patterns into content-based classification. After presenting and discussing the results of our experiments, we conclude the paper with possible directions for future work.

## 2 Related Work

In this work we integrate methods from two different areas of research: email classification and temporal data mining.

Numerous studies on email classification appeared in the machine learning literature in the last few years (for an overview of the major ones see [9]). Most of the approaches put in use only word-based features, completely ignoring the temporal aspect of the email domain. At the same time, Sahami et al. [15] showed that bringing in other kinds of features (spam-specific features in their study) could improve the classification results. To filter spam messages, they join together two sets of features, traditional textual and non-textual such as over-emphasized punctuation, the domain type of the sender, time the message was received, etc. Our work explores the possibility of the usage of more complex time-related features in a general email classification context.

In the data mining field there have been a lot of research on mining sequential patterns in large databases of customer transactions: from the first Apriori-like algorithms AprioriSome and AprioriAll [2] to their numerous modifications and extensions [13,16,17]. Similar approaches were proposed for discovering sequential patterns, also called frequent episodes, in sequences [12]. Recently, more complex temporal sequential patterns were investigated. Works such as [7,8,11] consider interval-based sequences where events last over a period of time (as opposed to point-based events). In this study we work with point-based events but concentrate on time elapsed between the events. We propose a new algorithm MINTS that finds temporal sequential patterns consisting of not only event types, but also the time intervals between the events. Therefore, our method predicts not only the expected event in a sequence, but also when the event is likely to happen.

Work by Kleinberg [10] applies a temporal analysis in the context of email. His assumption is that the appearance of a new topic in a document stream is signaled by a "burst of activity". He presents a formal framework for identifying such "bursts" and shows that the document structure imposed by this analysis has a natural meaning in terms of the content.

However, research on temporal data mining focuses only on the temporal aspect of data and does not take into account any content-based features. The main contribution of this paper is an attempt to integrate methods from the two areas in one powerful heterogeneous system.

## 3 Email Classification with Temporal Features

### 3.1 Simple Temporal Features

The simplest way of incorporating temporal information in email classification is to extract temporal features such as the day of the week and the time of the day the message was received from the message timestamps. We add those new features to content-based features[1] and run one of the standard classification algorithms. Unfortunately, our preliminary experiments showed that the overall effect of adding simple temporal features is not very positive for any learning algorithm that we tried. These results suggest one of the two possibilities: either temporal information is irrelevant to email classification or temporal information is associated with classes in more complex ways, and a simple classification algorithm cannot discover those associations from the timestamps by itself. It needs our help to represent those associations more explicitly, so that it can make use of them. To explore the second possibility we introduce a notion of temporal sequential patterns. From a temporal email sequence we discover temporal relations between the classes of messages in the form of temporal sequential patterns. Then, we transform these temporal relations into new features, combine them with content-based features, and feed them into one of the standard classification algorithms.

### 3.2 Temporal Sequential Patterns

**Definition.** A set of emails can be viewed as an *event sequence* $(c_1, t_1) \rightarrow (c_2, t_2) \rightarrow \ldots \rightarrow (c_n, t_n)$, where each event corresponds to an email and is represented as a pair $(c_i, t_i)$ with $c_i \in C$ being the category of the email

[1] As content-based features we use words composing the subjects and the bodies of emails. This is the most general representation of the email content. We do not include any other header information since it usually requires some type of preprocessing.

(the event type) and $t_i$ being the timestamp of the email. The events in the sequence are ordered by their timestamps: $t_1 \leq t_2 \leq \ldots \leq t_n$.

A *temporal sequential pattern* is an ordered sequence of event types $c_1 \rightarrow c_2 \rightarrow \ldots \rightarrow c_k$ along with an ordered sequence of time intervals $d_1 \rightarrow d_2 \rightarrow \ldots \rightarrow d_{k-1}$ denoted as $\{c_1 \rightarrow [d_1] \rightarrow c_2 \rightarrow [d_2] \rightarrow \ldots \rightarrow c_{k-1} \rightarrow [d_{k-1}] \rightarrow c_k\}$. The interpretation of pattern $\{c_1 \rightarrow [d_1] \rightarrow c_2 \rightarrow [d_2] \rightarrow \ldots \rightarrow c_{k-1} \rightarrow [d_{k-1}] \rightarrow c_k\}$ is the following: in a given event sequence there is a subsequence[2] $(c_1, t_1) \rightarrow (c_2, t_2) \rightarrow \ldots \rightarrow (c_k, t_k)$, where $t_1 \leq t_2 \leq \ldots \leq t_k$ and $t_2 - t_1 = d_1, t_3 - t_2 = d_2, \ldots, t_k - t_{k-1} = d_{k-1}$.

**Mining Temporal Sequential Patterns.** Our algorithm for mining temporal sequential patterns is based on the ideas first presented in the classical Apriori algorithm [1]. Apriori was initially designed to mine frequent[3] intra-transaction patterns in a database of customer transactions. It works iteratively producing an $(n+1)$-item pattern from a pair of $n$-item patterns that have the same prefix[4]. The observation that for any pattern to be frequent all its sub-patterns have to be frequent too makes the algorithm more efficient. So, before scanning the database to count the number of occurrences for a new $(n+1)$-item pattern we could check if all of its $n$-item sub-patterns are frequent.

To accommodate the temporal aspect of an email sequence we designed a new algorithm MINTS (MINing Temporal Sequential patterns) to mine sequential patterns along with the exact time intervals between the events in the patterns. Although there exist many extensions of the classical Apriori algorithm to deal with sequential data, to the best of our knowledge none of them deals with temporal sequences taking into account the time elapsed between the events. To add time intervals, two key modifications have to be made to the Apriori-like algorithm. The first one is when we combine pairs of 1-event patterns to make 2-event patterns, we have to consider every possible time interval between the events. The second modification is when we combine a pair of $n$-event patterns to produce an $(n+1)$-event pattern the $n$-event patterns have to have the same prefix, which includes not only the same event types, but also the same time intervals between the events. These modifications, especially the first one, significantly increase the computational complexity of the algorithm.

We also have to revise the criteria by which we choose interesting patterns. Traditionally, a pattern is considered interesting if its support and confidence exceed user-defined thresholds. Generally, support is defined as the number of occurrences of the pattern in the data, and confidence is defined as the ratio

[2] The elements of a subsequence are not necessarily consecutive elements of the event sequence.

[3] A pattern is called *frequent* if we can find at least $minS$ its occurrences in the data. $minS$ is a user-defined threshold called *minimum support*.

[4] A prefix of an $n$-item pattern is the first ($n$-1) items in the pattern.

of the support of the pattern and the support of its prefix. These conventional measures have several drawbacks, especially in the temporal context. First, support does not take into account the a priori probabilities of event types. Patterns formed by frequent event types would naturally occur more frequently than patterns formed by rare event types. Therefore, selecting patterns by their support can result in missing highly predictive patterns consisting of rare event types and finding a large number of useless patterns formed by frequent event types. Second, the confidence of a temporal pattern no longer represents the predictive power of the pattern. Consider the following example. In a given sequence we have 20 occurrences of event type $c_1$ and 10 occurrences of temporal pattern $c_1 \rightarrow [d] \rightarrow c_2$. Thus, the confidence of the pattern $c_1 \rightarrow [d] \rightarrow c_2$ is 50%. That means that after an event of type $c_1$ an event of type $c_2$ would happen in $d$ time units with 50% probability. However, if in the given sequence no other event types ever happen in $d$ time units after an event of type $c_1$, the probability of an event happened in $d$ time units after $c_1$ to be of type $c_2$ is 100%. Since we are interested in predicting the type of a new event knowing when it happened and, therefore, knowing the exact time distances between the new and previous events, we prefer a measure that would be related to the probability of an event encountered at a time point to be of a particular class and not to the probability to encounter an event of a particular class at the given time point.

To accommodate those drawbacks, we suggest an alternative measure of the predictive power of a pattern. We consider a pattern to be interesting if it occurs much more frequently than we would expect based on the frequency of the event types composing the pattern and the length of the intervals between the events. So, for each pair of event types we scan the sequence to find all time intervals that occur between events of those types and count the number of occurrences ($n$) for each time interval. Then, we would consider only those intervals for which $n$ is significantly greater than we would expect assuming the independence of events. In other words, if the probability to encounter the pattern $n$ times is very small, according to the Poisson distribution, then we consider it interesting and keep it; otherwise, we discard it.

What kinds of patterns can one expect to find? Among the patterns that we discovered, some patterns seem to occur by chance, but some are quite interesting. For example, we can find a pattern like $\{c_i \rightarrow [d_j] \rightarrow c_i\}$, which is interpreted as events of type $c_i$ occur regularly each $d_j$ time units (for example, once a week). These types of patterns are common for subscription letters. Another interesting pattern is $\{c_i \rightarrow [0\,hours] \rightarrow c_i\}$, which says that the events of type $c_i$ occur in bunches (or at least in pairs). We also can find patterns like $\{c_i \rightarrow [0 hours] \rightarrow c_i \rightarrow [0 hours] \rightarrow c_i\}$, $\{c_i \rightarrow [0 hours] \rightarrow c_i \rightarrow [0\,hours] \rightarrow c_i \rightarrow [0\,hours] \rightarrow c_i\}$, etc., that specify how many events of type $c_i$ usually occur within a few minutes from each other. These patterns are common for some public mailing lists like MLNet. Most of the patterns, like the ones described above, consist of one event type. Yet, in some datasets we can find patterns with several event types, which suggests that those event types are related. For example, messages from the QA department to software engineers regarding some problems with a software product could be followed by an email discussion of those problems among the engineers.

**Sequence Prediction.** Although temporal patterns give us insights into the nature of mechanisms generating the sequence, the main application of patterns is to predict the type of a new event in the sequence. To do so, we look for patterns that are present at the end of the sequence. If for pattern $\{c_1 \rightarrow [d_1] \rightarrow c_2 \rightarrow [d_2] \rightarrow \ldots \rightarrow c_{k-1} \rightarrow [d_{k-1}] \rightarrow c_k\}$ we find a subsequence corresponding to its prefix $\{c_1 \rightarrow [d_1] \rightarrow c_2 \rightarrow [d_2] \rightarrow \ldots \rightarrow c_{k-2} \rightarrow [d_{k-2}] \rightarrow c_{k-1}\}$, then the instance found in $d_{k-1}$ time units after the end of the subsequence is expected to have type $c_k$. We say that the pattern is applicable to that instance.

## 3.3 Integration of Temporal and Content-based Features

Temporal patterns alone will not provide us with good prediction results, but we can integrate them with content-based features, expecting that the temporal characteristics of data would enrich the information space and allow us to obtain better classification performance. In this work we propose six simple methods of combining temporal and content-based information.

The following three methods transform the temporal information into new features and add them to the set of content-based features:

- "Predicted Class"
  Based on the temporal patterns applicable to the instance in question, we can predict the class of that instance (see section 3.2). We add the predicted class to the feature set and train a classifier. This predicted class presents our expectations for the next message to be of a particular class. Certainly, our expectations cannot be very reliable; they are just guesses. We anticipate that the classification system would learn for which sets of feature values it could rely on those guesses and for which it could not.
- "Cascading"
  This method is similar to one presented in [6]. Instead of adding the class of an instance predicted by the temporal patterns, we add the probabilities of classes predicted by the temporal patterns. There can be several patterns applicable to the instance in question, and they can predict different classes. Based on pattern unexpectedness we calculate the probabilities for each class to be the class of the instance in question and add those probabilities to the feature set.
- "Patterns"
  Here we add all temporal patterns applicable to the instance in question to the feature set. We anticipate that the classification system would learn on which patterns and in what context it could rely.

The next two methods combine the predictions of two classifiers, content-based and time-based:

- "Simple Vote"
  This method implements the simple voting scheme. First, we train a classifier[5] on content-based features (a content-based classifier) and run it on the test set getting a set of class probabilities for each test instance. We also run our sequence prediction algorithm (a time-based classifier) on the same test set getting the second set of class probabilities. Then, we compare the probability sets and for each test instance pick the class with the greatest probability value.
- "Replacing Uncertainties"
  As in "Simple Vote" we have two classifiers (content-based and time-based) and two sets of class probabilities for each test instance. Since the content-based classifier is much more accurate than the time-based one, we pick the class predicted by the content-based classifier unless its probability is less than a certain threshold. If it is the case we pick the class predicted by the time-based classifier.

The last method is specific for the Naive Bayes classifier:

- "Replacing A Priori Probabilities"
  In the well-known Bayes' formula

$$P(Class|Doc) = \frac{P(Class) * P(Doc|Class)}{P(Doc)}$$

  the term P(Class) represents the a priori class probabilities. These probabilities show how frequent the classes are and how probable is for a class to be the class of an instance without any other knowledge about the instance. We can change the a priori probabilities with respect to our temporal expectations. For example, if the most frequent class is $c_1$ but the temporal patterns suggest the class $c_2$ to be the class of the next instance, then we put the a priori probability of $c_2$ greater than the a priori probability of $c_1$. We do it by simply replacing the a priori probabilities with the class probabilities predicted by the temporal patterns.

## 3.4 Results

We have tested the proposed approaches on 6 datasets described in Table 1.

Corpus Email I consists of about 1500 personal emails of one of the authors for a period of 1.5 years. Messages were divided into 15 categories by

[5] Any classification algorithm producing the class probabilities as the output can be used here. In these experiments we used only Naive Bayes since the implementations of Decision Trees and SVM that we have chosen cannot output the probabilities.

**Table 1.** Characteristics of the datasets used.

| Name | Description | # of classes | Size of the training set | Size of the test set | Vocabulary size |
|---|---|---|---|---|---|
| Emails I | Personal email | 15 | 1055 | 452 | 3119 |
| Emails II | Personal email | 8 | 2507 | 1075 | 3251 |
| Emails III | "Corporate" email | 5 | 1637 | 702 | 2148 |
| News I | comp.ai<br>comp.ai.alife<br>comp.ai.doc-analysis.misc<br>comp.ai.games<br>comp.ai.philosophy | 5 | 1425 | 610 | 4649 |
| News II | alt.folklore.music<br>alt.gothic.music<br>alt.music.bee-gees<br>alt.music.beethoven<br>alt.music-dire-straits | 5 | 597 | 149 | 1908 |
| News I+II | News I & News II joined | 10 | 1947 | 834 | 5569 |

their content and senders. Corpus Email II is a much bigger set of personal emails divided into 8 categories. These categories share much of the vocabulary, which makes them hard to distinguish for a conventional content-based classifier. Corpus Email III consists of messages sent to a company's general email address and models the situation where an automatic assistant has to redirect the incoming messages to the appropriate staff members.

We have also used the newsgroups messages, which are close in structure and nature to email messages. For News I corpus we have chosen 5 AI-related newsgroups and collected all the postings in these groups for about 3 months. This corpus is very hard to classify since the groups are very close in content and vocabulary. News II corpus is smaller and easier for classification; the categories are more distinct. Corpus News I+II is just two previous corpora merged together.

For each dataset we have trained and tested the standard ("Words only") and three modified ("Predicted Class", "Cascading" and "Patterns") versions of three learning algorithms, namely Decision Trees (C4.5) [14], Naive Bayes [3] and Support Vector Machines (SVM) [4]. We have also run three other modified versions ("Simple Vote", "Replacing Uncertainties" and "Replacing A Priori Probabilities") of the Naive Bayes algorithm. Table 2 summarizes the results of these experiments[6]. Figure 1 shows the performance of the three learning algorithms and their three modifications with varying vocabulary size (the number of content-based features used).

The first observation of the results is that we have been able to reduce the classification error for each dataset tested: the reduction in error for emails is up to 10%, for newsgroups it is up to 22%. The second observation is that no algorithm is a clear winner. From Figure 1 we can see that "Patterns" modification of SVM and Naive Bayes was the best algorithm for most of the vo-

[6] In these experiments the words were stemmed, and the stems appearing in less than 6 documents along with the stop-words were removed.

**Table 2.** Classification accuracies.

| Dateset | Naive Bayes | | | | | | | Decision Trees (C4.5) | | | | Support Vector Machines (SVM) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | words only | pred. class | casca-ding | pat-terns | simple vote | repl. uncert. | repl. a priori prob. | words only | pred. class | casca-ding | pat-terns | words only | pred. class | casca-ding | pat-terns |
| emails I | 82.52 | 82.74 | 78.98 | **83.19** | 82.30 | 82.74 | 82.96 | **81.6** | 80.3 | 81.2 | 81.0 | 78.54 | 73.23 | 78.54 | **80.31** |
| emails II | 49.30 | 49.30 | 51.91 | **54.42** | 48.93 | 49.77 | 53.95 | 51.2 | **52.0** | 51.4 | 51.2 | 56.00 | 49.95 | **60.56** | 59.44 |
| emails III | 65.81 | 65.67 | 65.24 | 65.67 | 65.81 | **66.38** | 65.67 | **42.2** | 42.2 | 42.2 | 42.2 | 64.81 | 64.96 | 64.96 | **65.10** |
| news I | 66.72 | 66.89 | 67.21 | 66.89 | 66.72 | **74.1** | 66.89 | 64.8 | 64.4 | **66.6** | **66.6** | **71.64** | 59.51 | 69.18 | 62.95 |
| news II | 82.55 | 83.22 | 83.89 | 82.55 | 82.55 | **85.91** | 81.88 | **91.3** | 91.3 | 91.3 | 90.6 | 87.92 | 54.36 | 89.26 | **89.93** |
| news I+II | 67.51 | 67.51 | 67.27 | 67.51 | 67.51 | **71.70** | 67.63 | 70.5 | **73.0** | 72.1 | 71.9 | **73.26** | 35.73 | 71.70 | 66.31 |

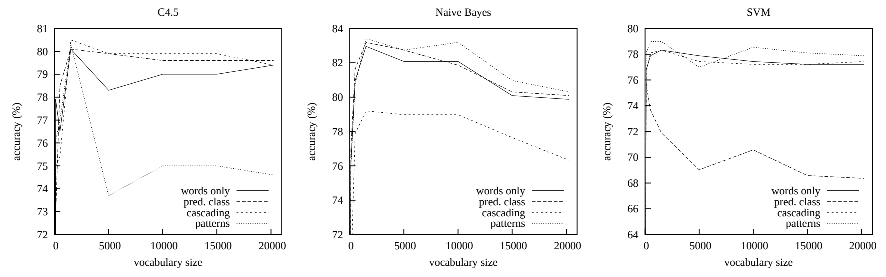

**Fig. 1.** Performance of the three algorithms and their three modifications with varying vocabulary size.

cabulary sizes. At the same time, this modification of Decision Trees worked very poorly for larger vocabulary. "Predicted class" performed pretty well with Decision Trees and Naive Bayes but extremely poorly with SVM. Surprisingly, the simple "Replacing Uncertainties" modification of Naive Bayes worked very well for all newsgroup datasets reducing the error by 13-22%.

We noticed that Naive Bayes and SVM performed better with the addition of temporal information on more datasets than Decision Trees. We attribute this to the fact that Decision Trees are univariate, which means that they can test only one attribute at a time. Temporal characteristics can be in quite complex dependencies with content-based features. When tested alone they are not much of help. They seem to do better when combined with content in more sophisticated ways such as in Naive Bayes and SVM.

## 4 Conclusion and Future Work

In this paper we present a novel approach of combining temporal and content-based features into a heterogeneous classification system. We apply this approach to the email domain. We also present a new algorithm MINTS to discover temporal relations in the form of temporal sequential patterns. These temporal relations are then embedded into the conventional content-based learning methods. The performance of the heterogeneous classification system has been tested on several email and newsgroup datasets. The system has been able to reduce the classification error by up to 22%.

An email system is a dynamic environment. Typically, the variety of email categories and their meaning are altered frequently: some categories become unused, new ones emerge. Temporal patterns change over time. Currently, we do not have any mechanisms to accommodate those changes. In the future, we have to consider a possibility of the evolutionary adaptation of the system to handle the changing context.

The proposed algorithm is fairly general and can be applied to other domains that are characterized by time-related and content-related features. In fact, email is not extremely time dependent, and, yet, we were able to achieve reasonable improvements there. With domains more heavily dependent on time we would expect even greater improvements. As our future work we plan to investigate the possibilities of applying our approach to other domains.

We can view each email message as an event generated by some external event such as a product release, a conference announcement, approaching holidays, etc. An external event "causes" a temporal pattern to emerge, and then the messages of a particular class are generated according to the pattern. In this work we analyzed only the temporal patterns themselves without looking at the external events. Because the nature of external events varies from national events to minor personal events, it is not clear how to retrieve them automatically. However, including these events into our analysis could result in greater improvements.

## Acknowledgements

This work was supported by Natural Sciences and Engineering Research Council of Canada, Communications and Information Technology Ontario, and AmikaNow! Corporation (www.amikanow.com).

## References

1. Agrawal, R., Srikant, R. (1994). Fast Algorithms for Mining Association Rules. Proc. of the 20th Int. Conf. on Very Large Data Bases (VLDB), 487–499. Morgan Kaufmann.
2. Agrawal, R., Srikant, R. (1995). Mining Sequential Patterns. Proc. of the 11th Int. Conf. on Data Engineering (ICDE), 3–14. IEEE Computer Society Press.
3. Borgelt, C. (2002). Bayes, version 2.7. http://fuzzy.cs.uni-magdeburg.de/~borgelt/software.html#bayes.
4. Chang, C.-C., Lin, C.-J. (2001). LIBSVM : a library for support vector machines. http://www.csie.ntu.edu.tw/~cjlin/libsvm/.
5. Ferris Research (2003). Spam Control: Problems & Opportunities. http:://www.ferris.com.
6. Gama, J. (1998). Combining classifiers by constructive induction. Proc. of the 10th European Conf. on Machine Learning (ECML), 178–189. Springer.

7. Höppner, F., Klawonn, F. (2002). Finding Informative Rules in Interval Sequences. Intelligent Data Analysis, **6**, 237–255.
8. Kam, P., Fu, A. W. (2000). Discovering Temporal Patterns for Interval-based Events. Proc. of the 2nd Int. Conf. on Data Warehousing and Knowledge Discovery (DaWaK), 317–326. Springer.
9. Kay, J., McCreath, E. (2001). Automatic Induction of Rules for E-Mail Classification. UM2001: 8th Int. Conf. on User Modeling, Workshop on User Modeling, Machine Learning and Information Retrieval.
10. Kleinberg, J. (2002). Bursty and Hierarchical Structure in Streams. Proc. of the 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD), 91–101.
11. Laxman, S., Unnikrishnan, K.P., Sastry, P.S. (2002). Generalized Frequent Episodes in Event Sequences. 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, Workshop on Temporal Data Mining.
12. Mannila, H., Toivonen, H., Verkamo, A. I. (1995). Discovering Frequent Episodes in Sequences. Proc. of the 1st Int. Conf. on Knowledge Discovery and Data Mining (KDD), 210-215. AAAI Press.
13. Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.-C. (2001). PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. Proc. of the 17th Int. Conf. on Data Engineering (ICDE), 251–225. IEEE Computer Society Press.
14. Quinlan, J.R. (1992). C4.5: Programs for Machine Learning. Morgan Kaufmann.
15. Sahami, M., Dumais, S., Heckerman, D., Horvitz, E. (1998). A Bayesian Approach to Filtering Junk E-Mail. Proc. of the AAAI Workshop on Learning for Text Categorization.
16. Srikant, R. Agrawal, R. (1996). Mining Sequential Patterns: Generalizations and Performance Improvements. Proc. of the 5th Int. Conf. on Extending Database Technology (EDBT), 3–17. Springer.
17. Zaki, M. (2001). SPADE: An Efficient Algorithm for Mining Frequent Sequences. Machine Learning, **42(1-2)**, 31–60.