On the Utility of Incremental Feature Selection for the Classification of Textual Data Streams

Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas

Department of Informatics,
Aristotle University of Thessaloniki,
54124 Thessaloniki, Greece
{katak,greg,vlahavas}@csd.auth.gr

Abstract. In this paper we argue that incrementally updating the features that a text classification algorithm considers is very important for real-world textual data streams, because in most applications the distribution of data and the description of the classification concept changes over time. We propose the coupling of an incremental feature ranking method and an incremental learning algorithm that can consider different subsets of the feature vector during prediction (what we call a feature based classifier), in order to deal with the above problem. Experimental results with a longitudinal database of real spam and legitimate emails shows that our approach can adapt to the changing nature of streaming data and works much better than classical incremental learning algorithms.

1 Introduction

The World Wide Web is a dynamic environment that offers many sources of continuous textual data, such as web pages, news-feeds, e-mails, chat rooms, discussion forums, usenet groups, instant messages and blogs. There are many interesting applications involving classification of such textual data streams. The most prevalent one is spam filtering [1]. Other applications include filtering of pornographic web pages for safer child surfing [2, 3] and delivering personalized news feeds [4]. Another recent application involves the filtering of *web spam* [5], a name for web pages whose sole purpose is to mislead search engines into including an irrelevant commercial web site in the results of a query.

The dynamic nature of the above data streams requires continuous or at least periodic updates of the current knowledge in order to ensure that it always includes the information content of the latest batch of data. This is important in domains where the concept of each class and/or the data distribution changes over time. For example in spam filtering, parts of the spam concept change over time as different unsolicited commercials come into vogue [6]. Similarly, in a personalized news feeder, the interests of a user are changing over time. This phenomenon is also known as *concept drift* [7].

It is obvious that computationally efficient mining of data streams requires incremental algorithms that can update the current knowledge without reprocessing past data, that are either unavailable or too costly to be retrieved. The huge

amount of data that is constantly arriving from a stream, does not allow to permanently store all data, rather a small part of it (the latest batch) or a summary of all the data that have been seen.

Textual data streams are also high-dimensional. Documents are usually represented as a bag-of-words and the feature vector for a collection of documents in a typical application comprises several thousands of words. However, not all features are necessary. Therefore, feature selection must be performed in order to reduce the dimensionality of the problem and allow learning algorithms to obtain higher quality of knowledge with less computational cost. For this reason, a lot of studies on feature selection methods for text classification have been performed in the past [8, 9].

However, to the best of our knowledge, no work exists on the issue of incrementally/periodically updating the features that a text classification algorithm considers. We argue that this is very important for real-world textual data streams, because the predictive power of features changes over time: words that in the past have been important, become redundant with the passing of time and new high-predictive words arise that were not considered before.

In order to deal with this issue, this paper proposes an approach for classification of textual data streams that is incremental both with respect of the examples that arrive and with respect to the subset of features that it considers over time. To the best of our knowledge, such an approach has not been considered before, not only for textual data, but also for any other type of high-dimensional data.

Our approach requires two components: a) an incremental feature ranking method, and b) an incremental learning algorithm that can consider a subset of the features during prediction. To verify the utility of incremental feature selection, we experimentally evaluate our approach on a chronologically ordered version of a spam and legitimate email collection. The results showed that incremental feature selection offers higher accuracy compared to classical incremental learning.

The rest of this paper is organized as follows: Section 2, presents background knowledge on text classification. In Section 3, we describe the proposed approach. In Section 4 we give details about the experimental setup, including the preprocessing of the data set and the specific algorithms that were used. In Section 5 we present and discuss the results and finally in Section 6 we conclude and propose some future work on this topic.

2 Automated Text Classification

Automated text classification has gained scientific interest in the last 20 years. Applications like document organization [10], text filtering [11] and author identification [12] are some representative examples of the research outcome in the field of text classification. The impressive growth of the world wide web in the last decade resulted in many new interesting applications like web page classification [13], e-mail organization [14] and spam filtering [1] and raised new research issues [15].

An informal definition of text classification would be "the categorization of previously unseen documents into predefined classes". In author identification for example, the classes are the authors. In document organization, the classes are a number of predefined topics and in spam identification there are only two classes: a mail can be spam or legitimate. This is a binary classification problem or else a text filtering problem.

The first problem we come across in any text learning task is that the data cannot be immediately processed by a classifier. We have to first follow a procedure to convert our text to a format that is acceptable by learning algorithms. The most common approach is the vector space model, where every text document is represented as a vector of feature weights $\overrightarrow{d_j} =< w_{1j}, \ldots, w_{|V|j} >$, where V is the set of words that occur at least once in a document and consists our problem's vocabulary. This is the so-called bag-of-words approach. Another option is to use phrases as features, although research has shown that this approach does not improve effectiveness [16]. The weight for each word of the vector is either tf-idf [17] (the term frequency in the collection of documents divided by the frequency in the current document), or more commonly a binary value denoting the existence or absence of the word in the document.

For text classification and especially for spam filtering applications, a widely used classifier, mainly for its simplicity and flexibility is the Naive Bayes Classifier [18] which showed decent performance in the identification of junk e-mails [1].

Feature selection has been studied extensively in the context of text classification [8, 9]. The reason is that text data are usually high-dimensional and feature selection is essential for a) reducing the computational complexity of machine learning algorithms, and b) improving the accuracy of classification.

Feature selection methods fall broadly into two categories: a) the filter approach, and b) the wrapper approach. In the wrapper approach feedback from the use of the learning algorithm with the selected features is used to evaluate these features. When the best set is found the algorithm is called to make new classifications based on this set of features only. On the other hand, the filter approach is totally independent of the classifier. Another useful categorization of feature selection methods is based on whether they evaluate individual features, or subsets of features.

Wrappers are usually more computationally intensive than filters due to extensive use of the learning algorithm. Methods that evaluate subsets of features are also more computationally intensive due to the larger number of evaluations that they have to consider. For this reason in text classification tasks, where the dimensionality of the data is typically very large, filters that evaluate features are usually considered. Features are ranked based on the result of the evaluation and the top N features are selected for further classification use.

3 Our Approach

Our approach uses two components in conjunction: a) an incremental feature ranking method, and b) an incremental learning algorithm that can consider a subset of the features during prediction.

In Section 2 we noted that feature selection methods that are commonly used for text classification are filters that evaluate the predictive power of all features and select the N best. Such methods evaluate each word based on cumulative statistics concerning the number of times that it appears in each different class of documents. This renders such methods inherently incremental: When a new labelled document arrives, the statistics are updated and the evaluation can be immediately calculated without the need of re-processing past data. These methods can also handle new words by including them in the vocabulary and initializing their statistics. Therefore the first component of our approach can be instantiated using a variety of such methods, including *information gain*, the χ^2 statistic and *mutual information* [8, 9].

The incremental re-evaluation and addition of words will inevitably result into certain words being promoted to/demoted from the top N words. This raises a problem that requires the second component of the proposed approach: a learning algorithm that is able to classify a new instance taking into account different features over time. This problem has not been considered before to the best of our knowledge. We call learning algorithms that can deal with it *feature based*, because learning is based on the new subset of features, in the same way that in *instance based* algorithms, learning is based on the new instance.

Two inherently feature based algorithms are *Naive Bayes* (NB) and *k Nearest Neighbors* (kNN). In both of these algorithms each feature makes an independent contribution towards the prediction of a class. Therefore, these algorithms can be easily expanded in order to instantiate the second component of our approach. Specifically, when these algorithms are used for the classification of a new instance, they should also be provided with an additional parameter denoting the subset of the selected features. NB will only consider the calculated probabilities of this subset, while kNN will measure the distance of the new instance with the stored examples based only on this subset.

It is worth noticing that the proposed approach could work without an initial training set. This is useful in personalized web-content (e-mail, news, etc.) filtering applications that we want to work based solely on our perception of the target class. However, very often an initial collection of labelled documents is available. Figure 1 presents algorithm InitialTraining for the initial training of our approach, based on such a collection of Documents that belong to one of several Classes.

The first step is to build the Vocabulary of distinct words that appear in all documents of the collection using the BuildVocabulary function. We also initialize WordStats, which is a construct that will hold the number of appearances of each Word in the Vocabulary for each different class of documents, for the purpose of feature ranking. Next, for each Document in the collection of training Documents we update the WordStats. Based on the calculated statistics, we sub-

```
input  : Documents, Classes
output: Classifier, Vocabulary, WordStats, FeatureList
begin
    Vocabulary ← BuildVocabulary(Documents)
    foreach Word ∈ Vocabulary do
        foreach Class ∈ Classes do
            WordStats [Word][1][Class] ← 0
            WordStats [Word][0][Class] ← 0
    foreach <Document, DocClass> ∈ Documents do
        foreach Word ∈ Vocabulary do
            if Word ∈ Document then
                WordStats [Word][1][DocClass] ← WordStats [Word][1][DocClass] + 1
            else
                WordStats [Word][0][DocClass] ← WordStats [Word][0][DocClass] + 1
    FeatureList ← ∅
    foreach Word ∈ Vocabulary do
        Evaluation ← EvaluateFeature(Word, WordStats)
        InsertSort(<Word, Evaluation>, FeatureList)
    Classifier ← BuildClassifier(Documents, Vocabulary)
end
```

Fig. 1. Algorithm InitialTraining

sequently evaluate each Word in the Vocabulary using the metric of preference and insert the Word and its Evaluation in the list FeatureList, which is sorted according to the evaluation metric. Finally we train a feature based classifier using all Documents and the complete Vocabulary. Note that the training of the Naive Bayes classifier does not demand any other statistics than those already collected in WordStats.

Figure 2 presents algorithm Update for the incremental update of our approach. When a new Document arrives as an example of a DocumentClass, the first thing to happen is to check if it contains any new words. If a new Word is present then it is added to the vocabulary (AddWord) and the WordStats of this Word are initialized to zero. Then for each Word in the Vocabulary we update the counts based on the new document, re-calculate the evaluation metric and sort the list according to the new evaluations, as before. Finally, the classifier must also be vertically updated based on the new example and also take into account any new words. Note that for the Naive Bayes classifier updating the WordStats is enough for this purpose.

Finally, when a new unlabelled Document arrives for classification, the feature based classifier of our approach considers just the top NumToSelect ranked words from the sorted FeatureList. This process is performed by algorithm ClassifyDocument shown in Figure 3.

```
input  : Document, DocClass, Classes, Vocabulary
output: Classifier, Vocabulary, WordStats, FeatureList
begin
    foreach Word ∈ Document do
        if Word ∉ Vocabulary then
            AddWord(Word, Vocabulary)
            foreach Class ∈ Classes do
                WordStats [Word][1][Class] ← 0
                WordStats [Word][0][Class] ← 0
    foreach Word ∈ Vocabulary do
        if Word ∈ Document then
            WordStats [Word][1][DocClass] ← WordStats [Word][1][DocClass] + 1
        else
            WordStats [Word][0][DocClass] ← WordStats [Word][0][DocClass] + 1
    FeatureList ← ∅
    foreach Word ∈ Vocabulary do
        Evaluation ← EvaluateFeature(Word, WordStats)
        InsertSort(<Word, Evaluation>, FeatureList)
    Classifier ← UpdateClassifier(Document, DocClass)
end
```

Fig. 2. Algorithm Update

```
input  : Classifier, Document, FeatureList, NumToSelect
output: Class
begin
    FeatureSubset ← SelectFeatures(NumToSelect, FeatureList)
    Class ← UseClassifier(Classifier, Document, FeatureSubset)
end
```

Fig. 3. Algorithm ClassifyDocument

4 Experimental Setup

In this section we present the data set, feature selection method and learning algorithm that were used in the experiments.

4.1 Data Set

In order to evaluate the utility of the proposed approach for classification of textual data streams, it is important to use real-world data. For the domain of spam filtering this means that we need real-world spam and legitimate emails chronologically ordered according to their date and time of arrival. In this way we can approximate the time-evolving nature of this problem and consequently we can evaluate properly the different approaches.

There are various collections of spam messages available on the Web, including the repository of SpamArchive[1], the public corpus of the SpamAssasin project[2] and the Ling-Spam corpus[3]. Our choice was the public corpus of SpamAssassin for two main reasons: a) Every mail of the collection is available with the headers, so we are able to extract the exact date and time that the mail was sent or received, and b) It contains both spam and ham messages with a decent spam ratio (about 20 percent).

The Spam Assassin collection comes in four parts (folders): spam, spam2, ham, and easy ham which is a collection of more easily recognized legitimate messages. In order to convert this collection into a longitudinal data set we extracted the date and time that the mail was sent. Then we converted the time into GMT time. Date was also changed where needed. We stamped each mail with its date and time by renaming it in the format yyyy_MM_dd_hh_mm_ss (yyyy: year, MM: month, dd:day, hh: hours, mm: minutes, ss: seconds). If a mail was more than once in the corpus (sometimes a user may get the same mail more than once) we kept all copies. All attachments were removed. The boolean bag-of-words approach was used for representing the mails.

4.2 Feature Selection Method and Learning Algorithm

The feature ranking method that we selected for the experiments is the χ^2 statistic, for its simplicity and effectiveness [8]. As we mentioned in the precious section, there are many other similarly simple metrics that could be used for instantiating our framework [8, 9]. Here, we are not focusing on the effectiveness of different feature selection methods, rather on whether the proposed incremental feature selection approach is useful in textual data stream classification.

The algorithm that we selected is Naive Bayes. The k-NN algorithm is inefficient for data-streams, because it needs to store all data. Naive Bayes on the other hand store only the necessary statistics, and is therefore our choice for incremental learning from textual data streams.

[1] http://www.spamarchive.org
[2] http://spamassassin.apache.org/
[3] http://www.iit.demokritos.gr/skel/i-config/downloads/

The Naive Bayes (NB) classifier is a simplistic but practical Bayesian learning algorithm that performs extremely well in many text classification tasks. The decision of the algorithm is determined by the following equation:

$$c_{NB} = argmax P(c_j) \prod P(w_i|c_j) \tag{1}$$

Where c_j is the $i-th$ class of our classification problem. In our case, we have only two classes. A mail can belong to class c_L (legitimate mail) or c_S (spam mail). w_i is the $i-th$ word of a vocabulary built from our corpus by collecting all distinct words. $P(c_j)$ expresses the probability a random document to belong in the $j-th$ class. It can be approximated from the training set by dividing the number of documents of class j with the total number of training documents available.

$$P(c_j) = \frac{|docs_j|}{|Examples|} \tag{2}$$

The $P(w_i|c_j)$ probability represents the possibility the word w_i to be present in a document of class c_j. Hence, it expresses how possible it is for our unlabelled document to belong in class c_j if it contains the word w_j. This possibility can be typically approximated by:

$$P(w_i|c_j) = \frac{NumberOfTimesWordw_iOccursinallc_jDocuments}{TotalNumberofc_jdocuments} \tag{3}$$

The simplicity of the NB classifier is obvious from all the above equations. To classify a new document we only need to have the knowledge of the above probabilities. We have expanded the Naive Bayes implementation of the Weka library of machine learning algorithms [19] in order to be able to perform classification using a subset of all features that were used for training.

The χ^2 statistic of a word w and a class c can be calculated by the following equation.

$$\chi^2(w,c) = \frac{N \times (AD - CB)^2}{(A+C) \times (B+D) \times (A+B) \times (C+D)} \tag{4}$$

where A is the number of times w and c co-occur, B is the number of times w occurs without c, C is the number of times c occurs without w, D is the number of times neither c nor w occur, and N is the total number of documents.

We have also expanded the implementation of the χ^2 feature ranking method of Weka [19] in order to perform incremental updates.

5 Results and Discussion

In the experiments, we compare the predictive performance of our approach (NB3) with a classical incremental Naive Bayes classifier (NB2) and a non-

incremental Naive Bayes classifier (NB1). NB1 is only trained once on a percentage of the data, and uses the features that are selected based on this initial training set. NB2 also uses a static feature space, but apart from the initial training, each time a new document arrives, it updates the probabilities for each feature. Our approach (NB3) also updates the probabilities not only for the selected features but for *all* of the features and in addition it recalculates the χ^2 rank of these features based on the updated probabilities. It therefore uses a dynamic feature space, that may change over time. In addition our approach can utilize new features that appear in new documents.

We varied the percentage of documents that were used for initial training from 0.1 to 0.5 with a step of 0.1. Using a 0.1 percentage of initial training data allows us to study the behavior of the filter for a longer time (0.9 of the whole longitudinal data set), while an initial training of 0.5 simulates a short period of online filtering. The number of features to select was set to 250 and 500.

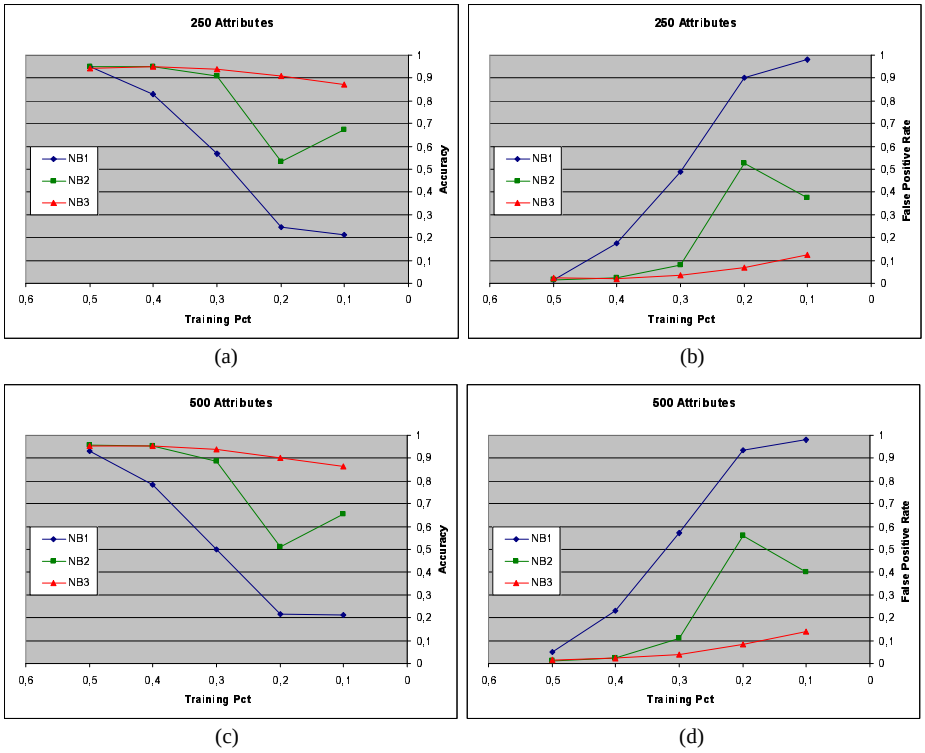

Fig. 4. (a) accuracy and (b) false positive rate when we select the top 250 features and (c) accuracy and (d) false positive rate when we select the top 500 features

Figure 4 shows (a) the accuracy and (b) the false positive rate when we select the top 250 features and (c) the accuracy and (d) the false positive rate

when we select the top 500 features. We first notice that the behavior of the three approaches follows the same pattern independent of the number of features used.

The non-incremental Naive Bayes classifier (NB1) performs very badly when run for a long time (0.1 to 0.4) and approximates the other two incremental classifiers when half of the total data that it subsequently classifies are used for training. This is something expected as incremental learning is important for updating the current model with the latest data.

The incremental Naive Bayes classifier (NB2) also has problems when run for a long time (0.1 to 0.3) and approximates NB3 when 0.4 or more of the total data is used for its training. This actually shows that vertically incremental learning alone cannot catch up with the changing nature of real-world data streams.

The feature based incremental Naive Bayes classifier together with the incremental χ^2 feature ranking method (NB3) shows much better behavior than the other two classifiers even when little data is used for initial training (i.e. even when run for a long time). This verifies our initial argument that incremental feature selection is very important for classification of textual data streams.

6 Conclusions and Future Work

This paper argued that incremental feature selection is very important for the classification of textual data streams due to the changes in data distribution and class concept that occur over time. We presented an approach that combines an incremental feature selection methods with what we called a feature based learning algorithm in order to deal with the above problem. The experimental results show that the proposed approach offers better predictive accuracy compared to classical incremental learning, and are encouraging for further work.

In the future we intend to experiment with maintaining statistics for a fixed number of features over time instead of the whole vocabulary. This would increase the efficiency of the proposed approach, due to the reduced storage and processing requirements. We would like to see whether this would also increase the effectiveness of the proposed approach in dealing with fast changing data/concepts, as it focuses more aggressively on the latest data.

References

1. Sahami, M., Dumais, S., Heckerman, D., Horvitz, E.: A bayesian approach to filtering junk E-mail. In: Learning for Text Categorization: Papers from the 1998 Workshop, Madison, Wisconsin, AAAI Technical Report WS-98-05 (1998)
2. Chandrinos, K.V., Androutsopoulos, I., Paliouras, G., Spyropoulos, C.D.: Automatic web rating: Filtering obscene content on the web. In Borbinha, J.L., Baker, T., eds.: Proceedings of ECDL00, 4th European Conference on Research and Advanced Technology for Digital Libraries, Lisbon, Portugal, Springer Verlag (2000) 403–406
3. Lee, P.Y., Hui, S.C., Fong, A.C.M.: Neural networks for web content filtering. IEEE Intelligent Systems **17** (2002) 48–57

4. Lang, K.: Newsweeder: Learning to filter netnews. In: Proceedings of the 12th International Conference on Machine Learning, Tahoe City, California (1995) 331–339
5. Fetterly, D., Manasse, M., Najork, M.: Spam, damn spam, and statistics: using statistical analysis to locate spam web pages. In: WebDB '04: Proceedings of the 7th International Workshop on the Web and Databases, New York, NY, USA, ACM Press (2004) 1–6
6. Fawcett, T.: "in vivo" spam filtering: A challenge problem for data mining. KDD Explorations **5** (2003)
7. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine Learning **23** (1996) 69–101
8. Yang, Y., Pedersen, J.O.: A comparative study on feature selection in text categorization. In Fisher, D.H., ed.: Proceedings of ICML-97, 14th International Conference on Machine Learning, Nashville, US, Morgan Kaufmann Publishers, San Francisco, US (1997) 412–420
9. Forman, G.: An extensive empirical study of feature selection metrics for text classification. J. Mach. Learn. Res. **3** (2003) 1289–1305
10. Larkey, L.S.: A patent search and classification system. In Fox, E.A., Rowe, N., eds.: Proceedings of DL-99, 4th ACM Conference on Digital Libraries, Berkeley, US, ACM Press, New York, US (1999) 179–187
11. Schapire, R.E., Singer, Y., Singhal, A.: Boosting and Rocchio applied to text filtering. In Croft, W.B., Moffat, A., van Rijsbergen, C.J., Wilkinson, R., Zobel, J., eds.: Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval, Melbourne, AU, ACM Press, New York, US (1998) 215–223
12. Fung, G.: The disputed federalist papers: Svm feature selection via concave minimization. In: TAPIA '03: Proceedings of the 2003 conference on Diversity in computing, New York, NY, USA, ACM Press (2003) 42–46
13. Peng, X., Choi, B.: Automatic web page classification in a dynamic and hierarchical way. In: IEEE International Conference on Data Mining. (2002) 386–393
14. Clark, J., Koprinska, I., Poon, J.: A neural network based approach to automated e-mail classification. In: IEEE/WIC International Conference on Web Intelligence (WI'03). (2003) 702–705
15. Sebastiani, F.: Machine learning in automated text categorization. ACM Computing Surveys **34** (2002) 1–47
16. Lewis, D.D.: An evaluation of phrasal and clustered representations on a text categorization task. In: SIGIR '92: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM Press (1992) 37–50
17. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Inf. Process. Manage. **24** (1988) 513–523
18. John, G.H., Langley, P.: Estimating continuous distributions in Bayesian classifiers. In: Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann (1995) 338–345
19. Witten, I., Frank, E.: Data Mining: Practical machine learning tools with Java implementations. Morgan Kaufmann (1999)