

exploiting structural information, namely *tagging*, *splitting* and *stacking*. Tagging is a simple preprocessing scheme which ensures that words appearing in different parts of the document are treated as different features. In the splitting approach, a separate model is built for each distinct document part. The models are combined in a way that is natural to the underlying classification algorithm. Finally, we consider stacking, in which a meta-classifier makes the final prediction based on the results of different models obtained by splitting.

We show how these three general approaches can be combined with a number of different text classifiers. More importantly, we show that the relative success of these approaches is fairly stable across different base classifiers as well as different categorization problems. In particular, we find that stacking consistently performs best and often outperforms the flat text model by a wide margin. We also point out some of the drawbacks of modeling document structure, such as data sparseness and out-of-vocabulary words. In order to tackle these difficulties, we propose to include the flat text of each document in the final prediction, a modification with which we consistently achieve our best results. All methods were evaluated on four standard datasets containing different types of semi-structured documents, such as web pages, e-mail messages and news stories. To our knowledge, this work is also the most comprehensive and conclusive evaluation of *different* approaches to exploiting structure in semi-structured document categorization to date.

The remainder of the paper is structured as follows. We review previous work on semi-structured document classification in the following section. Section 3 gives a brief description of the Naive Bayes (NB), Support Vector Machine (SVM) and Fuzzy Set Membership (FSM) text classifiers. We introduce semi-structured documents and discuss some issues regarding their representation in Section 4. Section 5 presents the different approaches to structured document classification that were considered in the study. We review our experimental setup and report on the results of our evaluation in Section 6. The final section summarizes our findings and points out some promising directions for future research.

2 Related Work

Although most text categorization research is limited to flat text representations, there exists a limited body of literature dealing with semi-structured documents.

Fürnkranz (1999) shows an improvement in accuracy when classifying hypertext documents by using the textual context of links to a web page as features for classification, rather than text of the web page itself. A similar approach is adopted by Glover et al. (2002), who also consider combining such a model with a model trained on the local text. Chakrabarti et al. (1998) also exploit the hyperlink structure of web pages and combine this with the prediction based on the local text. These approaches typically use document structure to determine the linked documents and to extract the relevant portions of text from linked documents. However, they are all limited to HTML pages or other types of linked documents. In this work, we are more concerned with the use of internal document structure.

A study by Yang et al. (2002) investigates a number of approaches to classification of hypertext documents. Although they also use information gained from the HTML hyperlink structure, they do consider internal document structure as well. They show

that using HTML meta tags and title words alone sometimes outperforms the flat text model. A previous study by the same authors (Ghani et al., 2001) shows that combining the output of these models with the output of the flat text model by simple averaging usually outperforms either representation alone.

Yi and Sundaresan (2000) propose a tree structure for modeling HTML documents where nodes correspond to document elements, such as headings, links, etc. They consider tagging words with the path to the node in which they occur and using a Naive Bayes classifier on the resulting feature vector. They also propose splitting the feature vector into separate vectors for different node paths and training a separate Naive Bayes model for each distinct path. This approach is augmented with transition probabilities for the nodes in order to incorporate a model of the structure itself into the prediction. Experiments on two HTML datasets indicate significant performance gains with tagging and further gains with their splitting approach.

Denoyer and Gallinari (2004) use a very similar splitting approach in combination with the Naive Bayes text classifier. This method is further refined by using an SVM classifier with a Fisher kernel on top of the original generative model. They report encouraging results on a number of datasets with different types of structured documents.

In email categorization, Brutlag and Meek (2000) observe that using only the features of certain header fields, such as sender and subject headers, can be beneficial for categorization. Recently, Klimt and Yang (2004) investigated email categorization using ridge regression to combine SVM models trained on different message parts. This method substantially improves on the flat text model for their email corpus. Their approach is similar to the stacking approach considered here. It differs in the choice of intermediate representation and the use of a different meta-classifier. Moreover, their evaluation is limited to email categorization.

3 Classification Algorithms

We evaluated different approaches to structured document classification in combination with three text classifiers. We used two popular text categorization algorithms, namely Naive Bayes and the Support Vector Machine, and a variant of the Fuzzy Set Membership algorithm, which was recently proposed by Wingate and Seppi (2004). A brief description of each of these three algorithms is given in the following subsections.

3.1 Naive Bayes

The Naive Bayes classifier is commonly used in text categorization (Mitchell, 1997; Lewis, 1998) due to its relatively good performance, favorable size and speed complexity and its ability to learn incrementally. Many variants of Naive Bayes are found in the literature, depending on the underlying probability model that is used. The most common are Multivariate Bernoulli and Multinomial models, of which the Multinomial model usually performs best (McCallum and Nigam, 1998; Eyheramendy et al., 2003).

To classify a previously unobserved document d , the Multinomial Naive Bayes classifier selects the class c that is most probable with respect to the document text. An estimate for the conditional class probability $p(c|d)$ is obtained using Bayes rule:

$$p(c|d) = \frac{p(c)p(d|c)}{p(d)} \quad (1)$$

The prior probability $p(c)$ is estimated from the training data as the relative frequency of training documents belonging to class c . The conditional probability of document d , given the class label c , is calculated from the probabilities of individual word occurrences over all words found in d :

$$p(d|c) = \frac{|d|!}{\prod_{w \in d} f(w, d)!} \prod_{w \in d} p(w|c)^{f(w, d)} \quad (2)$$

In the above equation, $f(w, d)$ denotes the number of occurrences of word w in document d and $|d|$ is the sum of the frequencies of all words in the document, i.e. the length of the document. The multinomial coefficient $\frac{|d|!}{\prod_{w \in d} f(w, d)!}$ denotes the number of all possible orderings of the words. Equation (2) is the probability of the observed word distribution under a multinomial model. It is based on the assumption that each word occurrence is the result of an independent multinomial trial. Note that $p(d)$ in Equation (1) and the multinomial coefficient in Equation (2) are independent of class, and can therefore be omitted in the final class selection rule.

The individual word probabilities are derived from the training data by simply counting the number of word occurrences (word frequencies)

$$p(w|c) = \frac{1 + f(w, c)}{|\mathcal{V}| + \sum_{w' \in \mathcal{V}} f(w', c)} \quad (3)$$

where $f(w, c)$ denotes the number of occurrences of word w in class c . Laplace smoothing is commonly used to avoid zero probabilities. The particular form of smoothing used here corresponds to adding 1 ‘virtual’ occurrence of each word in vocabulary \mathcal{V} to complement data in the training document collections for each class.

3.2 Support Vector Machine

The SVM (Vapnik, 1995) is a kernel-based classification method suited to binary classification problems. The SVM has favorable theoretical properties for text categorization and is also known to perform well on such tasks empirically (Joachims, 2002, 1998b; Yang and Liu, 1999). We outline the basic ideas and motivation behind the SVM here (for details, see for example Burges, 1998).

The SVM seeks to find a hyperplane that separates positive and negative training examples with the widest margin, as depicted in Figure 1. A dual optimization problem to the problem of finding the maximum margin hyperplane exists, which may be solved efficiently using quadratic programming. It turns out that the maximum margin hyperplane is a linear combination of the examples which lay closest to the decision boundary. These examples are called *support vectors*.

The SVM classifies a test example \vec{d} into the positive class if it occurs on the positive side of this hyperplane, and vice versa. In the linear case, this means that the positive class is selected if

$$\vec{a} \cdot \vec{d} + b > 0 \quad (4)$$

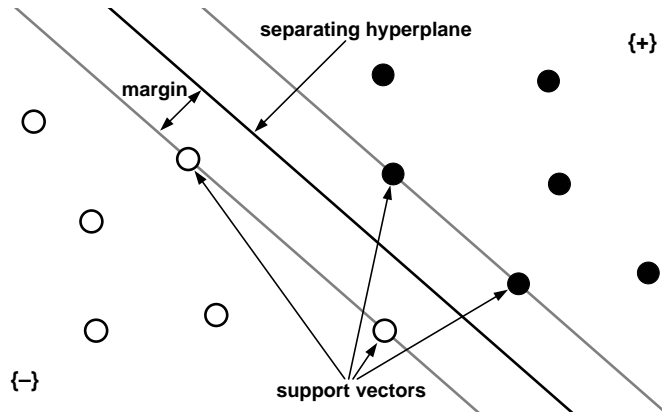


Figure 1: Support vectors and the maximum margin hyperplane

where \vec{a} and b are the model parameters learned by the SVM.

The SVM may be generalized to non-linear class boundaries by replacing the dot product with a non-linear kernel, however, linear kernels are usually sufficient for text classification problems (Yang and Liu, 1999).

Different methods exist for extending the SVM to multi-class problems. We use the most straightforward and most often used *one-vs-rest* approach which builds one binary classifier for each class. All training documents not in the class are used as negative examples. The class with the highest score is predicted in case of unlabeled classification. For multilabeled classification, all classes that satisfy the decision rule in Equation 4 are selected.

A number of different document representations have been studied for text categorization with SVMs (see Debole and Sebastiani, 2003). In our experiments, documents were represented using Term Frequency/Inverse Document Frequency (TFIDF) vectors, a representation which is commonly used in combination with the SVM. TFIDF weights for each word w in document d are computed by the following formula:

$$\text{TFIDF}(w, d) = f(w, d) \cdot \log \frac{|D|}{|\{d'; d' \in D \wedge f(w, d') > 0\}|} \quad (5)$$

where $f(w, d)$ is the number of occurrences of word w in document d and $|\{d'; d' \in D \wedge f(w, d') > 0\}|$ is the number of documents in the training data that contain w . These weights are used as components of the document vector \vec{d} .

3.3 Fuzzy Set Membership

The FSM algorithm (Wingate and Seppi, 2004) is a simple ad hoc linear classifier. It was designed as an alternative to Naive Bayes, primarily to overcome the need for smoothing and other parameter estimation problems of NB, without sacrificing its speed and its ability to learn incrementally.

The original FSM algorithm, as proposed by Wingate and Seppi (2004), selects the class $c(d)$ according to the following class selection rule:

$$c(d) = \arg \max_c \left[\sum_{w \in d} f(w, d) \frac{f(w, c)}{f(w, \bullet)} \right] \quad (6)$$

The sum in Equation 6 is over all words w that appear in document d . $f(w, d)$ denotes the number of occurrences of word w in document d , $f(w, c)$ is the number of occurrences of the word in all training documents belonging to class c and $f(w, \bullet)$ is the total number of occurrences of word w in the entire training dataset. The fraction $\frac{f(w, c)}{f(w, \bullet)}$ can be interpreted as the probability of class c , given that word w appears in the document. The overall score for class c is thus the *average* probability of class c over all word occurrences.

We found that the original FSM method suffers from a high bias in favor of larger classes. In order to improve performance on datasets with skewed training data, we modified the original class selection rule to use *relative* word frequencies:

$$c(d) = \arg \max_c \left[\sum_{w \in d} f(w, d) \frac{f^*(w, c)}{\sum_{c'} f^*(w, c')} \right] \quad (7)$$

where $f^*(w, c)$ denotes the relative frequency of word w in class c :

$$f^*(w, c) = \frac{f(w, c)}{f(\bullet, c)} \quad (8)$$

In the above equation, $f(\bullet, c)$ is the number of occurrences of all words in the training data for class c . This modified algorithm, which we labeled FSM*, is very similar to the Rocchio-style AIM classifier (Barret and Selker, 1995), which was applied to email categorization by Brutlag and Meek (2000).

4 Modeling Semi-structured Documents

In this article, we focus on classification of *semi-structured* documents. Such documents may be broken down into *elements* or *fields*, each of which contains either structured (numeric or symbolic) data or non-structured textual data. In general, semi-structured documents are represented with a hierarchical *tree structure*, so that structural elements may include other (sub)elements. Two examples are shown in Figure 2.

We model documents as collections of *components*, defined by some mapping from the original tree structure. Each component contains the textual content of the corresponding structural elements. In the simplest scenario, a straightforward one-to-one mapping from structural elements to components can be used. When dealing with tree-structured documents, components can be defined by the most specific node containing the actual text (Denoyer and Gallinari, 2004), or the full path to the node in the tree structure, thus preserving the structural context of the element (Yi and Sundaresan, 2000). In general, the mapping depends on the particular semantics of document structure and its perceived relevance to classification, much like choosing appropriate attributes to describe examples in any problem domain. For example, a `<description>` element in an XML document may be used in different contexts to describe different entities. Therefore, `<description>` elements could be used in combination with some parent element to define different components for descriptions of different entities. On the other hand, using the full path of

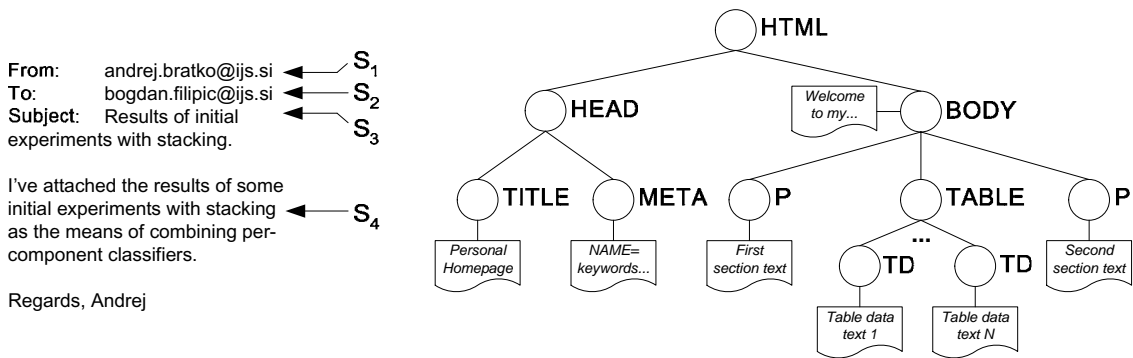


Figure 2: Two typical examples of semi-structured documents. The first is an email message with a simple flat structure consisting of four fields S_1 - S_4 . The second example is a tree-structured HTML document.

structural elements may be impractical, since many elements may be considered irrelevant for classification (such as, for example, various formatting tags in HTML).

5 Exploiting Document Structure

In this section, we present three different methods for exploiting structural information for document categorization. In particular, we would like to decouple the strategy used for exploiting document structure from the underlying text classification algorithm.

5.1 Tagging

A common approach is to treat word occurrences in different document components as different features. We refer to this as *tagging*, since the most straight-forward implementation is to tag words with the name of the component in which they appear, thus producing different features for the same word in different contexts. This method has the advantage of being completely independent of the classification algorithm that is used and thus requires no modification of the underlying classifier.

It should be noted that tagging is often used by authors in comparative evaluations of text categorization algorithms, such as the one by Dumais et al. (1998), but no comparison to the flat text model is usually provided. Experiments by Yi on two HTML datasets indicate that tagging can significantly boost accuracy (Yi and Sundaresan, 2000).

5.2 Splitting

Another strategy for incorporating document structure is to train as many instances of the base classifier as there are structural components. A separate model is trained for each document part. We call this approach *splitting*, since the flat text *bag-of-words* vector is split into a set of vectors, one for each component. Each document is represented by one such set of vectors.

5.2.1 Combining the Individual Component Models

The splitting approach assumes that words are distributed differently in different document components and attempts to build a better classifier by modeling these distributions separately. Word occurrences remain the primary features on which classification is based. The predictions of the individual component models are combined so that all word occurrences are considered equally important for classification, irrespective of their structural context. The contribution of document components is therefore proportional to the amount of words that they contain. The way in which this is achieved generally depends on the classification algorithm at hand. Details for the three classifiers that were considered in this paper are given below.

NB: Staying with the independence assumption, components are considered independent, so that the total probability of the observed document is merely a product of the individual component probabilities:

$$p(d|c) = \prod_{s \in d} p_s(d_s|c) \tag{9}$$

$$p_s(d_s|c) = \prod_{w \in s} p_s(w|c)^{f_s(w,d)} \tag{10}$$

The first product is over all structural components s that are present in the document d where $p_s(d_s|c)$ is the probability of each document component. A separate model p_s is trained for each component and word frequencies $f_s(w, d)$ are maintained on a per-component basis. This method of combining per-component Naive Bayes models has also been used by other authors (Yi and Sundaresan, 2000; Denoyer and Gallinari, 2004).

SVM: The output of the SVM classifier is the distance from the test example to the separating hyperplane. The most straight-forward method of combining these predictions is a sum over all per-component models. However, this approach overvalues words occurring in shorter components. The reason for this is that the SVM is relatively stable w.r.t. the dimensionality of the input vector, thus we expect all components to contribute a similar amount to the overall sum *on average*. In other words, the information about how many features contributed to the prediction of a particular component model is lost.

Platt reports empirical evidence that the output of a linear SVM classifier is often proportional to the log odds of the positive class (Platt, 1999). In view of this, a weighted sum of the scores produced by the individual component models seems like a good way of combining the predictions. One plausible approach is to multiply the component scores with the number of features that contributed to the score. The binary decision rule for the SVM thus becomes:

$$\sum_{s \in d} f_s(\bullet, d)(\vec{a}_s \cdot \vec{d}_s + b_s) > 0 \tag{11}$$

The sum is over all structural components s , $f_s(\bullet, d)$ is the total number of occurrences of all words in the component and \vec{d}_s is the corresponding feature vector of the document. A separate model $\langle \vec{a}_s, b_s \rangle$ is learned by the SVM for each component. If we assume

that the output of the SVM is independent of the dimension of the input vector, then this approach ensures that all word occurrences are treated as equally important.

FSM*: The FSM* algorithm assigns a score to each class which is a sum over all features in the document vector (Equation 7). After splitting the vector by document components, this sum is over all features of all the resulting vectors. Equivalently, the class score is computed as a sum of the scores produced by the individual component models:

$$c(d) = \arg \max_c \left[\sum_{s \in d} \sum_{w \in s} f_s(w, d) \frac{f_s^*(w, c)}{\sum_c f_s^*(w, c)} \right] \quad (12)$$

Again, the first sum is over all structural components s that are present in document d . Relative word frequencies $f_s^*(w, c)$ are maintained for each document component separately.

5.3 Stacking

The splitting approach discussed in the previous section has the advantage of modeling word distributions in each document component separately. When using this approach, longer components will usually dominate the classification. Although this can be justified by the fact that such components contain more information on which the classification is based, in reality, we may expect that some document components are more discriminative than others. A reasonable solution is to use *stacked generalization* or *stacking* (Wolpert, 1992) to combine the predictions of individual per-component models. In the stacking approach, the predictions of a number of *level-0* classifiers are used as input for a *level-1* classifier, or *meta-classifier*, which in turn makes the final prediction. To achieve this, a number of design decisions must be made.

5.3.1 Meta Features

Firstly, we have to choose a suitable representation for the output of the *level-0* models. The features of this intermediate representation are also called *meta-features*, since they serve as input to the meta-classifier. Ting and Witten (1997) suggest that per-class probability estimates predicted by each of the underlying models are suitable meta-features. When classifying an example document d , the input for the meta-classifier is a fixed-length vector \vec{e}_d of the following form:

$$\vec{e}_d = \langle p_s^c; s \in d, c \in C \rangle \quad (13)$$

Here, p_s^c is the probability of class c , as predicted by the model corresponding to the structural component s .

We implemented stacking using per-component models of NB, SVM and FSM* as level-0 classifiers. The meta-features p_s^c are computed differently for each base classifier, as indicated below. All equations for p_s^c contain a normalization constant Z , which is chosen so that $\sum_c p_s^c = 1$.

NB: Since Naive Bayes is a probabilistic classifier, its predictions are directly applicable as meta-features after normalization:

$$p_s^c = \frac{1}{Z} \cdot p(c) \prod_{w \in s} p_s(w|c)^{f_s(w,d)} \quad (14)$$

SVM: In order to obtain probability estimates, we transform the output of the level-0 SVM classifier for component s with the sigmoid function and re-normalize¹:

$$p_s^c = \frac{1}{Z} \cdot \frac{1}{1 + e^{a_{sc}^{\vec{}} \cdot \vec{d}_s + b_{sc}}} \quad (15)$$

In the above equation, $\langle a_{sc}^{\vec{}}, b_{sc} \rangle$ are the parameters of the maximum margin hyper-plane for the binary classification problem for class c and \vec{d}_s denotes the feature vector of document d pertaining to component s .

FSM*: The FSM* classifier is a probabilistic classifier. To obtain meta-features, we simply re-normalize the output of the per-component FSM* models:

$$p_s^c = \frac{1}{Z} \sum_{w \in s} f_s(w, d) \frac{f_s^*(w, c)}{\sum_{c'} f_s^*(w, c')} \quad (16)$$

5.3.2 The Meta Classifier

We expect that a suitable meta-classifier will be able to learn which document components are more discriminative and weight the predictions of the corresponding models accordingly. The work of Ting and Witten (1997) suggests that a linear combination of the predictions produced by the level-0 classifiers may work well in practice. We chose the SVM with a linear kernel as our meta-classifier. Besides being a linear model geared directly toward classification, we expect it to work well with the chosen intermediate representation \vec{e}_d with all meta-features in $[0..1]$.

5.3.3 Training the Stacked Classifier

Finally, a strategy must be devised for training the meta-classifier. In particular, unbiased training examples containing predictions made by the level-0 classifiers should be provided. To this end, we create a k -fold split of the training data. For each fold, $1/k$ of the training data is held out and the level-0 classifiers are trained on the remaining portion of the data. The predictions of these models on the held-out portion of the data are used to create training examples for the meta-classifier. After training the meta-classifier, the level-0 classifiers are re-trained on the entire training set. At classification time, an example document is first classified by all level-0 (per-component) classifiers. The output of these models is combined by the meta-classifier to form the final prediction.

¹A sigmoid is often used to transform the output of an SVM into probability estimates (see for example Platt, 1999). It should be noted that our approach is very simplistic in this respect. A more sophisticated approach would be to introduce additional parameters in order to calibrate the sigmoid. However, we expect that any gains would be negligible, since the output of the level-0 SVM classifiers is fairly regular and the meta-classifier may adapt to the uncalibrated probability estimates easily.

5.4 Drawbacks of Structured Models

Word frequencies in natural language are modeled well by the heavy-tailed Zipfian (power-law) distribution. This means that most words in a typical corpus are extremely infrequent, making inference of predictive models very difficult. Using a structured representation implies that a different model, or a different set of parameters must be learned for each structural component, greatly increasing the problem of data sparseness.

A second problem is an increase in the number of words that are novel in a certain context, so called *out-of-vocabulary* words. By splitting the training data, we will often discard meaningful words in a test document because they are novel in the context of a particular document component, even if the word is otherwise a good predictor of document class. It is intuitive to expect that a word which is common in one structural component is likely to occur in other components. Moreover, words generally have the same *meaning* irrespective of the structural context.

As an initial step toward tackling these issues, we experiment with including the flat text of each document as if it were yet another component. In the tagging approach, this allows the classifier to decide whether a word occurrence is relevant for classification in the context of a certain document component, or whether the word is predictive on its own. Intuitively, the less expressive flat text model has a higher bias and a lower variance. It thus acts as a “regularization component” or prior in the splitting approach. In the stacking approach, the meta-classifier may use the flat text model as a back-off in the event that per-component models are weak, presumably due to insufficient training data.

6 Experiments and Results

We conducted an extensive experimental evaluation of the different approaches to semi-structured document categorization that were discussed in previous sections. This section contains a description of our experimental setup and a summary of the results of these experiments.

6.1 Experimental Setup

In all experiments, texts were first split into word tokens consisting of one or more consecutive alphabetical characters delimited by whitespace or punctuation. All characters were converted to lower case. All words in the resulting vocabulary were used for classification. No stemming, stop-word removal or feature selection was used.

In experiments with the SVM, the TFIDF weights were computed separately for each train/test split, so that the IDF component reflected the document frequency of a word in the training set alone. All TFIDF document vectors were normalized with their Euclidean length. Words that appeared in less than two training documents were excluded prior to training of the SVM. We used the SVM^{light} package of Joachims (1998a) with a linear kernel for the SVM classifier. All parameter settings were left at their default values, in particular, the default value for C was used (the trade-off between training error and margin), resulting in $C = 1$ due to normalization. We used our own implementations of Naive Bayes and FSM*.

Unless otherwise noted, testing was done using 4-fold cross validation. To this end, we split the data randomly into four equal parts. On each cross validation fold, 25% of

Table 1: Basic statistics for all four document collections used for evaluation.

Dataset	<i>Number of categories</i>	<i>Number of documents</i>	<i>Documents in smallest category</i>	<i>Documents in largest category</i>	<i>Number of unique words</i>
WebKb	4	4199	504	1641	54980
20News-18828	20	18828	628	999	113726
Ling-Spam	2	2893	481	2412	54722
Reuters-21578	135	12902	0	3964	29294

the data was used for testing and the remaining 75% of the data was used for training, so that each document appeared in the test set exactly once. We evaluated all methods using the same split in all experiments.

6.2 Datasets

Experiments were conducted on four publicly available datasets containing different types of semi-structured documents. We give some basic statistics for all four corpuses in Table 1.

The WebKb² dataset consists of HTML web pages from 4 different universities which are classified into 7 different categories. Following McCallum and Nigam (1998), we only used the 4 largest topical categories (**project**, **faculty**, **student** and **course**).

The 20 Newsgroups³ dataset consists of 18828 newsgroup posts, collected from 20 different public newsgroups. The messages are distributed roughly evenly across all 20 classes.

The Ling-Spam⁴ dataset contains spam and legitimate email messages posted to a linguistics newsgroup. The distribution of examples is approximately five to one in favor of legitimate messages. We used the “bare” version of this dataset and the predefined 10-fold cross validation split.

The Reuters-21578⁵ dataset contains newswire articles from the Reuters news agency. Reuters-21578 is a *multilabeled* dataset, so each article may be assigned to zero or more categories. We perform multilabeled categorization by training a binary *one-vs-rest* classifier for each category. Experiments were performed on the standard “ModApte” train/test split, which uses 12902 documents of the original collection, of which 9603 are used for training. As is customary for this dataset, we report classification results for the 10 largest categories.

6.3 Document Modeling

Semi-structured documents were modeled as follows. Email messages were represented with **sender**, **subject** and **body** components. Note that the Ling-Spam corpus does not

²<http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/>

³<http://people.csail.mit.edu/~jrennie/20Newsgroups/>

⁴<http://www.aueb.gr/users/ion/data/>

⁵<http://www.daviddlewis.com/resources/testcollections/reuters21578/>

include `sender` headers, so we omit this component in experiments with the Ling-Spam dataset. Web pages were split into `<TITLE>`, `<H*>` (all headings) and `<BODY>` components, as well as `link` URLs and `image source` URLs. We used `title` and `body` fields of news articles in the Reuters-21578 dataset.

We represent each document with the textual value of these components. We ignore any hierarchical relations between the components (e.g. we ignore the fact that `<H*>` components are nested within a `<BODY>` component) and the cardinality of components (e.g. we concatenate the contents of all `<H*>` components found in a document into one textual feature).

6.4 Performance Measures

We measured classifier *accuracy* on unlabeled datasets, i.e. the proportion of test documents that are classified correctly by the classifier. For the multilabeled Reuters dataset, we report on *micro-averaged break-even-point* (BEP) which is defined in terms of the standard measures of *precision* and *recall*. Precision p is the proportion of true document-category assignments among all assignments predicted by the classifier. Recall r is the proportion of true document-category assignments that were also predicted by the classifier. Formally, the BEP statistic finds the point where precision and recall are equal. Since this is hard to achieve in practice, a common approach is to use the arithmetic mean of recall and precision as an approximation, i.e. $BEP = (p + r)/2$. The *micro-averaged* BEP is computed by first summing the elements of all binary contingency tables (one for each category). Precision, recall and BEP are then computed from these accumulated statistics.

To determine statistical significance of the results, we compare methods using a one-sided micro sign test (Yang and Liu, 1999), which we apply to the classification outcomes over all test documents, i.e. whether the classification is correct or incorrect. For the multilabeled Reuters dataset, the test was performed on the outcomes of all binary document-category decisions, as suggested by Yang and Liu (1999). We compared all pairs of methods for each dataset/classifier combination. This yields a total of 252 tests, to which we applied the Benjamini-Hochberg correction (Benjamini and Hochberg, 1995) to control the false discovery rate due to multiple comparisons.

6.5 Evaluation of Individual Component Models

To evaluate the usefulness of individual document components for our particular text categorization tasks, we conducted experiments using only the textual contents of individual components. The results of these tests are depicted in Figure 3. The flat text representation is superior to any individual structural component with the single exception of the `<BODY>` component in the WebKb dataset in combination with the SVM classifier. As a rule of thumb, components that contain a larger amount of text are more informative for classification. The graphs suggest that all classifiers make good use of the abundance of features provided by such components. One notable exception is the `title` component of Reuters news stories, which is more useful than the `body` component. As expected, the SVM clearly dominates in the flat text representation which is consistent with studies by other authors (Yang and Liu, 1999; Joachims, 1998b). However, its performance is inferior to some of the simpler models for certain components with fewer features. This is

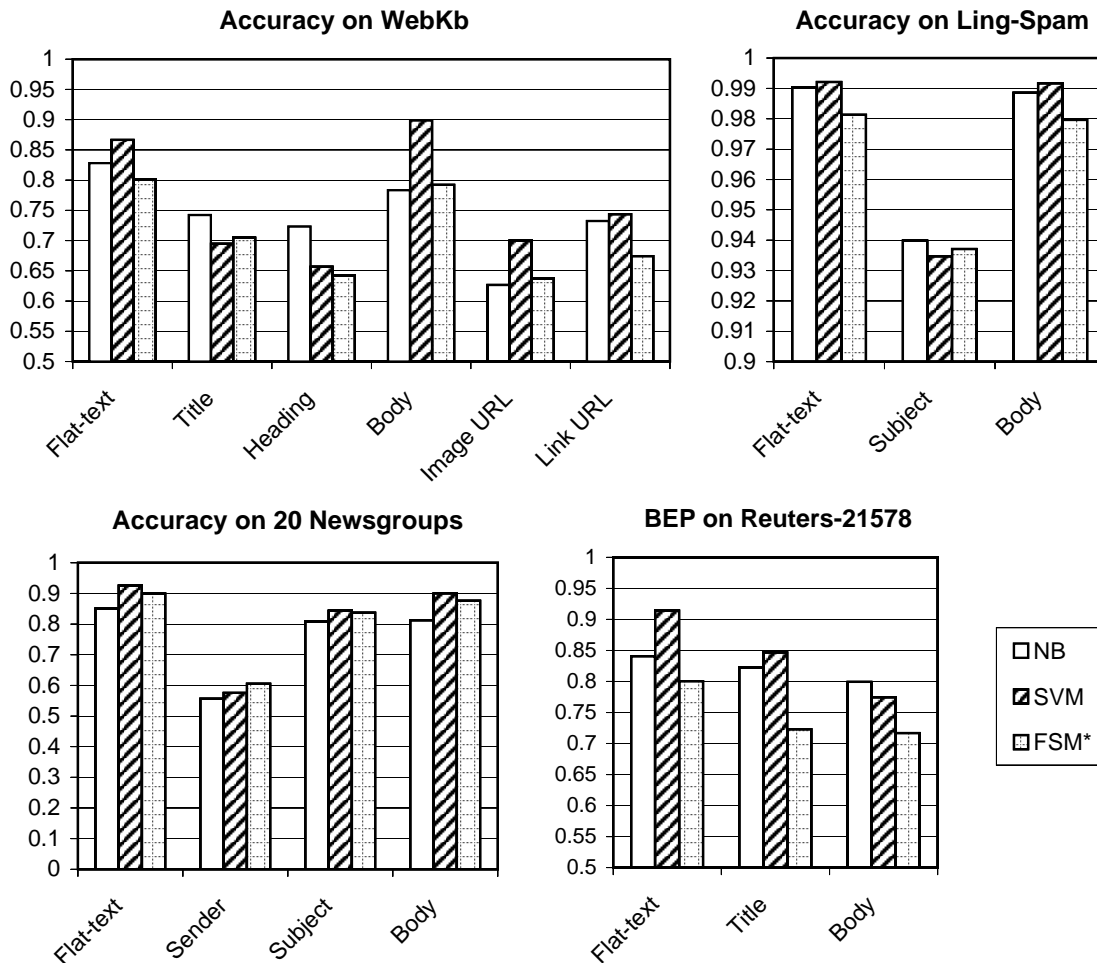


Figure 3: Results of categorization using the flat text representation and individual document components separately. Note the different scales of the graphs.

consistent with the findings of Forman and Cohen (2004), who show that simpler models, such as Naive Bayes, may be superior to SVMs when little training data is available.

6.6 Main Results

The main results of our experiments are given in Tables 2 and 3. The first table contains categorization results for all unlabeled datasets. The second table contains results on the multilabeled Reuters-21578 dataset. The tag^+ , split^+ and stack^+ captions denote tagging, splitting and stacking approaches in which the flat text of the document was also used as if it were a separate component.

As with flat text categorization, the SVM proves most effective in all structured approaches as well. Of the different approaches to structured document categorization, the stacking approach is the clear winner. Its performance is superior to other models on all datasets, sometimes strikingly so. The results achieved with the stacking approach

Table 2: Unlabeled categorization results. Average accuracy (%) over all cross validation runs is reported, followed by the standard deviation. The best result for each dataset and base classifier is in boldface. Results that are followed by a superscript differ significantly from *all* other results on the same row. The † and ‡ superscripts denote, respectively, 0.05 and 0.01 confidence levels with a false discovery rate equal to the confidence level.

	Flat	Tag	Tag⁺	Split	Split⁺	Stack	Stack⁺
<i>WebKb</i>							
NB	82.8±0.6	83.7±0.7	83.2±0.8	85.5±0.6	84.9±0.7	90.3±0.9	90.8±0.8[‡]
SVM	86.7±0.7	89.0±1.0	87.2±0.8	91.2±0.6	90.5±0.7	94.6±0.8 [‡]	94.7±0.9[‡]
FSM*	80.1±1.1	82.3±0.8	80.9±0.9	83.4±0.8	82.3±1.0	88.9±0.7 [‡]	89.1±0.8[‡]
<i>20News-18828</i>							
NB	85.1±0.5	84.7±0.5	86.0±0.5	85.0±0.5	85.3±0.5	89.9±0.5	90.7±0.6[‡]
SVM	92.6±0.2	92.9±0.4	92.8±0.2	91.9±0.4	92.4±0.4	93.3±0.2	93.7±0.4[‡]
FSM*	90.0±0.5	91.0±0.6	90.5±0.5	91.3±0.6	90.6±0.5	91.2±0.5	91.8±0.5[‡]
<i>Ling-Spam</i>							
NB	99.0±0.8	98.9±0.8	99.0±0.8	99.0±0.8	99.0±0.8	99.4±0.5[†]	99.4±0.5[†]
SVM	99.2±0.4	99.2±0.3	99.2±0.4	99.2±0.4	99.2±0.4	99.3±0.2	99.4±0.3
FSM*	98.1±1.2	98.2±1.1	98.2±1.1	98.2±1.1	98.2±1.1	99.0±0.8 [‡]	99.1±0.8[‡]

Table 3: Micro-BEP (%) on the 10 largest categories of Reuters-21578. The best result for each base classifier is in boldface. The results achieved by the **stack⁺** method differ significantly from the results of *all* other methods to the $p < 0.01$ level, using a one-tailed micro sign test and multiple test correction with a false discovery rate of 0.01.

	Flat	Tag	Tag⁺	Split	Split⁺	Stack	Stack⁺
NB	84.1	84.5	82.7	85.0	84.1	85.1	88.4[‡]
SVM	91.4	91.0	91.4	91.3	91.5	91.6	92.3[‡]
FSM*	80.0	81.4	80.8	81.6	80.8	84.4	85.7[‡]

Table 4: Relative performance of `flat`, `tag+`, `split+` and `stack+` methods for each dataset and classifier pair. Methods are ordered by increasing accuracy/BEP. Methods appearing after a `<` or `<<` sign achieve significantly higher results compared to *all* methods before the sign, with a significance level of 0.05 or 0.01, respectively. The false discovery rate is again equal to the significance level. Results of methods delimited by `~` do not differ significantly.

<i>WebKb</i>	
NB	flat << tag ⁺ << split ⁺ << stack ⁺
SVM	flat << tag ⁺ << split ⁺ << stack ⁺
FSM*	flat << tag ⁺ << split ⁺ << stack ⁺
<i>20News-18828</i>	
NB	flat << split ⁺ << tag ⁺ << stack ⁺
SVM	split ⁺ << flat << tag ⁺ << stack ⁺
FSM*	flat << tag ⁺ ~ split ⁺ << stack ⁺
<i>Ling-Spam</i>	
NB	flat ~ tag ⁺ ~ split ⁺ < stack ⁺
SVM	flat ~ tag ⁺ ~ split ⁺ ~ stack ⁺
FSM*	flat ~ tag ⁺ ~ split ⁺ << stack ⁺
<i>Reuters-21578</i>	
NB	tag ⁺ << flat ~ split ⁺ << stack ⁺
SVM	flat ~ tag ⁺ ~ split ⁺ << stack ⁺
FSM*	flat << tag ⁺ ~ split ⁺ << stack ⁺

are always improved by including the flat text representation. This gain is statistically significant in most cases.

The benefits of tagging and splitting approaches are less clear and vary among different combinations of base classifiers and test datasets. Including the flat text of a document in the prediction typically stabilizes this behavior. An interesting pattern emerges if we observe the relative performance of different methods, all of which also include the flat text model in their prediction, as depicted in Table 4. In 10 out of 12 trials, the flat text model performs worst. It performs second to last in the remaining two experiments. The stacking approach is consistently best. Although splitting usually outperforms tagging, this is not always the case and the differences are rarely significant.

Generally, exploiting document structure seems most beneficial for the WebKb dataset. Here, tagging, splitting and stacking consistently improve performance over the flat text representation. The combination of Naive Bayes and splitting improves accuracy by 2.7%, which is comparable to the improvement of 2% reported by Denoyer and Gallinari (2004), who use a similar approach on a different subset of WebKb, although they also consider transitions between structural elements in their model. The 82.8% accuracy of the flat text NB model is comparable to their NB baseline of 81%. Using the `stack+` method, classification error on the WebKb dataset drops by 8–9% compared to the flat text model. It is almost halved in all cases.

The `stack+` method also performs well on the 20 Newsgroups and Reuters datasets. Although the absolute gains are not as pronounced as for WebKb, they are consistently

significant. This is particularly relevant since these two text collections are favorites in text categorization research, although we are not aware of any previous study which would consider exploiting document structure for categorization on these datasets. Our Reuters results are comparable to those obtained by Dumais et al. (1998). They report 92.0% micro-BEP using the SVM with a different document representation and mutual information feature selection. They use tagging in their experiments, i.e. they distinguish words that appear in the title from words in the body of a document. Our flat text 20 Newsgroups results with Naive Bayes are practically identical to the 85% accuracy reported by McCallum and Nigam (1998). Our best stacking result with the SVM is considerably higher than the 91.3% accuracy reported by Bekkerman et al. (2003), which was achieved with a combination of SVM and distributional word clusters⁶. To the best of our knowledge, our result of 93.7% is the best 20 Newsgroups result reported to date.

Stacking also performs best on the Ling-Spam dataset, although the difference is not always significant. Even the baseline algorithms achieve very good results on this dataset and thus leave little room for improvement.

7 Conclusion

We have presented three different strategies for exploiting internal document structure for text categorization and evaluated their performance on a number of datasets containing different types of semi-structured documents. Of the methods that were considered, stacking is the clear winner in terms of classification performance. The potential benefits of tagging and splitting are less clear and vary among datasets. We believe that stacking is the most viable approach when accuracy is the most important factor in the design of a text categorization system. A minor limitation of the stacking method is its computational overhead. We have also shown that incorporating the flat text model usually improves performance. This improvement is particularly consistent with stacking, yielding the winning combination in all trials.

There are many ways in which the stacking approach could be improved, and we intend to pursue these options in future research. In particular, the meta-classifier lacks the information on how confident the predictions of the level-0 classifiers are. This additional information could reflect the number of words that contribute to the prediction of each level-0 model or more sophisticated *reliability indicators* could be used, as proposed by Bennett et al. (2005). The use of meta-classifiers with richer hypothesis spaces should be investigated in order to take advantage of such additional information.

Throughout this work, we assume the representation of documents is prescribed in advance. However, choosing a suitable representation may prove difficult when dealing with complex, highly structured documents, especially if the semantics of the document structure are not known. Automated methods for finding a good mapping from the original tree structure may therefore present an interesting avenue for future research.

Of the three text categorization algorithms used, the FSM algorithm is a newcomer in the field of text categorization. A modified version of this algorithm, FSM*, exhibits a combination of strong classification performance and favorable computational properties.

⁶Note that their result was achieved on a slightly different version of the 20 newsgroups dataset, which also contains duplicate documents.

Although not the primary goal of this paper, our results do warrant further research into the FSM method.

Acknowledgment

The work presented in this paper was supported by the Ministry of Higher Education, Science and Technology of the Republic of Slovenia.

References

- Barret, R. and T. Selker (1995). Aim: A new approach for meeting information needs. Technical Report IBM RC 9983, IBM Research.
- Bekkerman, R., R. El-Yaniv, N. Tishby, and Y. Winter (2003). Distributional word clusters vs. words for text categorization. *Journal of Machine Learning Research* 3, 1183–1208.
- Benjamini, Y. and Y. Hochberg (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B, Methodological* 57, 289–300.
- Bennett, P. N., S. T. Dumais, and E. Horvitz (2005). The combination of text classifiers using reliability indicators. *Information Retrieval* 8(1), 67–100.
- Brutlag, J. D. and C. Meek (2000). Challenges of the email domain for text classification. In *Proceedings of ICML '00, 17th International Conference on Machine Learning*, pp. 103–110. Morgan Kaufmann Publishers Inc.
- Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery* 2(2), 121–167.
- Chakrabarti, S., B. E. Dom, and P. Indyk (1998). Enhanced hypertext categorization using hyperlinks. In *Proceedings of SIGMOD-98, ACM International Conference on Management of Data*, pp. 307–318. ACM Press, New York, US.
- Debole, F. and F. Sebastiani (2003). Supervised term weighting for automated text categorization. In *Proceedings of the 2003 ACM Symposium on Applied Computing*, pp. 784–788. ACM Press.
- Denoyer, L. and P. Gallinari (2004). Bayesian network model for semi-structured document classification. *Information Processing and Management* 40(5), 807–827.
- Dumais, S., J. Platt, D. Heckerman, and M. Sahami (1998). Inductive learning algorithms and representations for text categorization. In *Proceedings of 7th International Conference on Information and Knowledge Management*, pp. 148–155. ACM Press.
- Eyheramendy, S., D. D. Lewis, and D. Madigan (2003). On the naive Bayes model for text categorization. In *Proceedings of AISTATS 2003, 9th International Workshop on Artificial Intelligence and Statistics*.

- Forman, G. and I. Cohen (2004). Learning from little: Comparison of classifiers given little training. In *Proceedings of PKDD-04, 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pp. 161–172.
- Fürnkranz, J. (1999). Exploiting structural information for text classification on the WWW. In *Proceedings of AIDA-99, 3rd International Symposium on Advances in Intelligent Data Analysis*, pp. 487–498.
- Ghani, R., S. Slattery, and Y. Yang (2001). Hypertext categorization using hyperlink patterns and meta data. In *Proceedings of ICML-01, 18th International Conference on Machine Learning*, pp. 178–185. Morgan Kaufmann Publishers, San Francisco, US.
- Glover, E., K. Tsioutsoulis, S. Lawrence, D. Pennock, and G. Flake (2002, May 7–11). Using web structure for classifying and describing web pages. In *Proceedings of WWW2002, 11th International World Wide Web Conference*, Honolulu, Hawaii, pp. 562–569.
- Joachims, T. (1998a). Making large-scale support vector machine learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola (Eds.), *Advances in Kernel Methods: Support Vector Machines*, pp. 169–184. MIT Press, Cambridge, MA.
- Joachims, T. (1998b). Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pp. 137–142. Springer Verlag.
- Joachims, T. (2002). *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms*. Kluwer Academic Publishers.
- Klimt, B. and Y. Yang (2004). The Enron corpus: A new dataset for email classification research. In *Proceedings of ECML-04, 15th European Conference on Machine Learning*, pp. 217–226.
- Lewis, D. (1998). Naive Bayes at forty: The independence assumption in information retrieval. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pp. 4–15.
- McCallum, A. and K. Nigam (1998). A comparison of event models for naive Bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, pp. 137–142.
- Mitchell, T. (1997). *Machine Learning*. McGraw-Hill.
- Platt, J. (1999). Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans (Eds.), *Advances in Large Margin Classifiers*, pp. 61–74. MIT Press.
- Ting, K. M. and I. H. Witten (1997). Stacked generalization: When does it work? In *Proceedings of IJCAI-97, 15th International Joint Conference on Artificial Intelligence*, pp. 866–873.
- Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc.

- Wingate, D. and K. Seppi (2004). Linking naive Bayes, vector quantization, and fuzzy set membership for text classification. Unpublished manuscript, Brigham Young University, Provo, UT.
- Wolpert, D. H. (1992). Stacked generalization. *Neural Networks* 5, 241–259.
- Yang, Y. and X. Liu (1999). A re-examination of text categorization methods. In *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval*, pp. 42–49. ACM Press, New York, US.
- Yang, Y., S. Slattery, and R. Ghani (2002). A study of approaches to hypertext categorization. *Journal of Intelligent Information Systems* 18(2-3), 219–241.
- Yi, J. and N. Sundaresan (2000). A classifier for semi-structured documents. In *Proceedings 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 340–344. ACM Press.