INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Classificação Automática de Páginas Web numa Hierarquia de Tópicos

Vitor Hugo Fernandes Sequeira
(Licenciado)

Dissertação para obtenção do grau:

Mestre em Engenharia Informática e de Computadores

Comité de Avaliação

Presidente:	Prof. Doutor Antonio Manuel Ferreira Rito da Silva
Orientador:	Prof. Doutor Bruno Emanuel da Graca Martins
Co-orientador:	Prof. Doutor Pável Pereira Calado
Vogáis:	Prof. Doutor David Martins de Matos

Agosto 2011

Resumo

O continuo crescimento de informação em formato digital tem tornado impossível manter sustentável um processo de catalogação tradicional. Os sistemas de classificação automática têm sido alvo de muito trabalho de investigação, e têm sido desenvolvidas novas técnicas para dar resposta a este crescimento de informação. Uma forma de fazer a classificação de documentos envolve o uso de uma hierarquia de tópicos. Desta forma, é possível organizar conteúdos desde classes mais genéricas até classes especificas.

Vários autores dedicaram-se ao estudo de técnicas para realizar tarefas de classificação hierárquica automática de forma eficiente, mas têm-se deparado na generalidade com os mesmos problemas. As classes em níveis mais baixos são frequentemente pobres em exemplos positivos, o que dificulta a tarefa de caracterizar documentos pertencentes a essas classes. Além disso, também é difícil tornar um sistema de classificação hierárquica escalável, não só considerando o aumento de documentos para treino como também o um aumento de níveis e classes na hierarquia a considerar.

Nesta tese de Mestrado são propostas técnicas para a realização de tarefas de classificação hierárquica, as quais tentam reduzir os tempos de treino reduzindo o número de documentos. Essa redução foi feita comparando dois métodos para selecção de documentos, o primeiro simples e ingénuo em que é desprezada a forma como os documentos possam estar organizados na hierarquia, o segundo método de selecção tem em consideração a forma como os documentos estão organizados na hierarquia. Além de reduzir o tamanho da amostra de treino, foi proposta uma extensão da tradicional estratégia de classificação *top-down* em que além da classe mais provável é também considerada a segunda classe mais provável em cada nível.

Os resultados validaram a extensão proposta à estratégia *top-down* conseguindo melhorias no desempenho dos classificadores em duas colecções de documentos distintas. A redução do número de documentos para treino não revelou ter o impacto desejado, os resultados são baixos e não é garantido que o tempo gasto para treinar os classificadores reduza.

Abstract

With the continuous growth of information in digital form, it has become impossible to maintain a sustainable process of traditional cataloging said. Since this automatic classification systems has been investigated and new classification techniques developed to meet this growth of information. One way to make the classification of documents make it respecting a hierarchy of topics. This way you can organize content from a more general class to a specific class, the class from which the document can be an example.

Several authors have devoted themselves to the study of techniques for performing an automatic hierarchical classification task efficiently, but have faced often with the same problems. The classes at lower levels are not traditionally rich in positive examples, which makes it difficult to characterize a class. Furthermore, it is difficult to make a scalable hierarchical classification system, not only considering the increase of the documents for training as well as the increased levels and classes in the hierarchy to consider.

In this master thesis we develop approaches capable of performing hierarchical classification tasks, which attempt to reduce training time by reducing the number of documents. This reduction was done by comparing two methods for selecting training documents, the first simple and naive as it ignores how the documents can be organized in the hierarchy, and the second selection method taking into account how the documents are organized in the hierarchy. Besides reducing the size of training sample, this MSc thesis proposes one extension to the traditional *top-down* classification approach, where in addition to the most likely class the system also considers the second most probable class at each level of the hierarchy of classifiers.

The proposed extension to the traditional *top-down* approach was validated. The classifiers achieve better results with two distinct document collections. The document selection performed to reduce the time spent to train the classifiers do not show the desired impact, the qualiity of the results decreases and it is not guarantted that the time to train the classifiers will reduce.

Acknowledgements

The writing of this document represents the end of something special in my life. Only with the support of several people was this ending possible, it is to that people that I want to show my gratitude.

In first place, I want to thank my parents and my sister for the unconditional support throughout my life, particulary on my academic path.

I also want to thank to all members of DMIR group for all the support during my MSc thesis. I want to give a special thanks to my thesis advisor, professor Bruno Martins, and my thesis co-advisor, professor PĂˇvel Calado, for allowing and believind in my qualities to solve the proposed problem in this MSc thesis. I wont't forget all the support and avaiable time to discuss my ideias and help to follow the right path.

To my freshmen friends Tiago Alves, Rita Simões, Ricardo Gamboa, Filipe Luís, Luís Franqueira, Nélson Franqueira, Eduardo Ferreira and André Silva, without you my academic path would not taste the same way.

Last, but not the least, I want to thank to the great woman behind this little man, PatrĂcia, for all the friendship, love, and encouragement.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Automatic document classification is the general task of assigning one or more categories to an electronic document (Sebastiani, 2002). This task is usually addressed through machine learning techniques, with basis on a compact representation of the document contents. Machine learning algorithms such as Support Vector Machines (SVM) or Naive Bayes are usually used. These algorithms are based on a vectorial representation of the documents. Recently, several authors have explored automatic classification in the context of Web documents (Ceci & Malerba, 2007; Dumais & Chen, 2000; Xhemali *et al.*, 2009).

There are different types of document classification tasks. The simplest one is binary classification, where documents are assigned to either a positive or negative class. Another task is multi-class classification, where documents are assigned to one of several possible classes, and which can be solved through a combination of binary classifiers (e.g., one for each class).

Classification tasks can also be flat or hierarchical. In the hierarchical case, the lower classes represent a specialization of the upper classes. Hierarchical classification tasks can be solved through a number of different approaches. We can, for instance, have a flat classifier for each level of the hierarchy, or we can flatten the hierarchy, this way transforming a hierarchical classification problem into one of flat classification.

In text classification problems, documents are most often characterized by the terms that they contain. Consequently, the representation of textual data is of a very high dimensionality. An important aspect of text classification, particularly in the case of hierarchical classification, is thus feature selection. However, to the best of our knowledge, there are few empirical demonstrations of the effect of feature selection in hierarchical classification tasks.

Common information retrieval measures are normally used to evaluate the performance of document classification systems. Previous works have proposed extensions of traditional metrics, such as *Precision* or *Recall*, to the cases of multi-class and hierarchical document classification.

1.1 Thesis Proposal and Validation Methodology

In my MSc thesis, I propose an extension of the traditional top-down hierarchical classification approach, allowing classifiers to test an alternative path in the hierarchy. This extension considers the two most probable classes in each level of the hierarchy. Another objective of my MSc project is to find a method to reduce the time spent to train the classifiers. Two methods were proposed to perform document selection in order to reduce the size of the training dataset. The first method, is relatively *naive* because it does not consider the distribution of the documents over the hierarchy. The second document selection method is more *refined* because, when the document selection is performed, the distribution of the documents over the hierarchy is considered. Finally, a third objective of my MSc thesis is to compare different feature vector approaches namely, the bag of words with all the text tokens of the documents, and simple feature selection techniques such as stopword removal and lemmatization. To support the training and validation of the classification models, two different document collections were used, namely the *4Universities* dataset and the dataset that was made available to the participants of the first edition of the Pascal Challenge[1]

1.2 Contributions

The main contributions of my MSc thesis are as follows:

- Evaluating the impact of using different feature selection strategies when building the feature vectors, for hierarchical classification;
- Evaluating the impact of reducing the number of training documents;
- Evaluating an extension of the traditional top-down classification approach, which considers more than one path over the tree of classification decisions.

[1] `http://lshtc.iit.demokritos.gr/`

1.3 Results

The results achieved with the experiments performed, are not conclusive about the feature selection methods used to estimate the feature vectors, The two hierarchical classification approaches achieved their best results when performed different feature selection methods. The document selection methods brought a significant reduction over the time spent to train the classifiers, but the quality of the results also decreased significantly.

Experimental results also show that the proposed extension to the traditional top-down classification approach is valid. System achieved an accuracy of 59,77% and f-measure achieved a score of 33,47%, against the 54,39% accuracy and the 24,44% f-measure achieved with the traditional top-down classification approach.

1.4 Document Outline

The rest of this dissertation is organized in four chapters.

Chapter 2 presents the main concepts and related work in the field of document classification. We describe concepts such as document representation and classification and common measures used to evaluate automatic classifiers. We also survey the current state-of-the-art in hierarchical document classification.

Chapter 3 details the chosen approaches to solve the hierarchical classification problem.

Chapter 4 describes the validation plan. We have a description of the datasets used to test the hierarchical classifier, the preprocessing operations performed to turn data readable for the classifier, and the results of the experiments performed to validate the hypothesis.

In Chapter 5 we have a general analysis of the developed work. The chapter discuss the main problems that emerged along the course of the work, and presents directions for future work.

Chapter 2

Concepts and Related Work

In this chapter we will introduce and discuss some of the main concepts related to automatic text classification, document representation, types of classification tasks and measures to evaluate the results of classification systems. We will also present two well known classification algorithms, namely Naïve Bayes and Support Vector Machines. Finally, we describe related work in the field of hierarchical classification that, at the time of my research, represented the state-of-the-art in hierarchical classification.

2.1 Concepts

In this section we will introduce and discuss some of the main concepts related with text representation, text categorization and evaluation of text categorization systems. These concepts will be described not only for simple classification tasks, such as binary classification, but also in terms of how they can be extended to support in hierarchical classification tasks.

2.1.1 Text Representation Schemes

In order to make a document understandable to a classifier, the document first needs to be preprocessed, i.e., transformed into a feature-vector (Salton & Buckley, 1988). A feature-vector is a vector where each dimension corresponds to a separate feature, usually a token from the text. A text token can be a single word or an n-gram (i.e., a subsequence of n words or characters from the text). A simple implementation of feature-vectors is to set the value of each dimension as one or zero, according to the presence or absence of a specific token in the document. Another

simple approach consists of using the occurrence frequency of the feature (i.e., a token) in the document to weight each dimension in the vectorial representation.

Each individual dimension of feature vector can also be weighted according to more sophisticated schemes than the simple use of occurrence frequency. The term frequency-inverse document frequency heuristic (*tf-idf*) is the combination of two heuristic measures, namely term frequency *tf* and inverse document frequency *idf*. The first one is the ratio between the number of occurrences of a feature in a document and the total number of features in the same document. The second is obtained through the $\log$ of the ratio between the total number of documents in the collection and the number of documents where the feature appears. The intuition is that terms that occur frequently in a document should be more important, but terms that occur in most documents are less important in discriminating between them. For a feature i, the individual *tf* and *idf* scores are given by Equation 2.1 and Equation 2.2, respectively.

$$tf_i = \frac{n_i}{\sum_k n_k} \tag{2.1}$$

$$idf_i = \log \frac{|D|}{1 + |\{ d : f_i \in d \}|} \tag{2.2}$$

The *tf-idf* is finally obtained multiplying *tf* with *idf* (Equation 2.3).

$$\textit{tf-idf}_i = \frac{n_i}{\sum_k n_k} \times \log \frac{|D|}{1 + |\{ d : f_i \in d \}|} \tag{2.3}$$

In long documents we can have many different features, resulting in vectors with *tf-idf* values different from zero for many different features. The longer the documents, the more likely they are to contain a specific feature. To circumvent this effect, the feature's weights can be normalized. A possible normalization procedure is the *cosine normalization* that explained in Singhal *et al.* (1995), where the individual feature weights are divided by the Euclidean length of the *tf-idf* weighted document vector.

$$\textit{normalized tf-idf}_i = \frac{\textit{tf-idf}_i}{\sqrt{\sum_i \textit{tf-idf}_i}} \tag{2.4}$$

When the feature-vector is computed, we can consider all the features present in the document or we can perform some filtering operations to discard features that do not add any semantic value to the representation of the document. Those features normally correspond to stop-words (common words like adverbs or prepositions), which are not related with the theme of the document, but instead with the sentence structure (Ceci & Malerba (2007); Dumais & Chen (2000)). Stopword

removal is therefore a common preprocessing step of feature selection.

Other common feature selection approaches are based on thresholding the values of individual features. Document Frequency (DF) and Information Gain (IG) are examples of metrics that are commonly used to perform automatic feature selection. Document Frequency is the number of documents where a feature occurs. Features that have Document Frequency values lower than a first threshold and higher than second threshold are not considered. This assumes that both rare and very common features are not informative enough to perform a good classification. Information Gain represents an estimation of the number of bits of information obtained for class prediction by knowing about the presence or absence of a feature in a document. One possible way to estimate Information Gain is the one used by Yang & Pedersen (1997). Let $C = (C_1, C_2, ..., C_m)$ be the set of *m* possible classes to witch we can assign each document, and let P the probability of occurrences for a class C_i. We then have that the information gain of a feature *t* estimated by the following formula:

$$\begin{aligned} IG_{(t)} = &- \sum_{i=1}^{m} P(c_i)\ \log P(c_i) \\ &+ P(t)\ \sum_{i=1}^{m} P(c_i \mid t)\ \log P(c_i \mid t) \\ &+ P(\bar{t})\ \sum_{i=1}^{m} P(c_i \mid \bar{t})\ \log P(c_i \mid \bar{t}) \end{aligned} \tag{2.5}$$

Another technique used for preprocessing text is reducing words to their stems (Ceci & Malerba (2007); Porter (1997)). The stem of a word is a representation of the concept that is related to that word. Applying stemming algorithms to a document results in having document represented in terms of concepts, avoiding the presence of a word in many different morphological variants (e.g., a verb written in different tenses). With stemming, we can also reduce the dimensionality of the feature-vector.

2.1.2 Text Classification

There are different types of document classification tasks. Figure 2.1 represents a taxonomy that illustrates the relationship between different types of classification problems.

First, a classification task can be binary or multi-class. In binary classification, the classifier verifies if a document belongs or not to a certain class. In the multi-class case the classifier assigns the document into one of several classes.

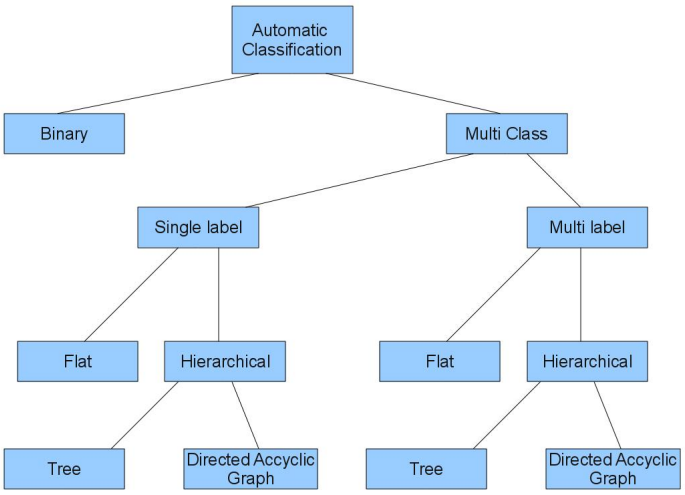


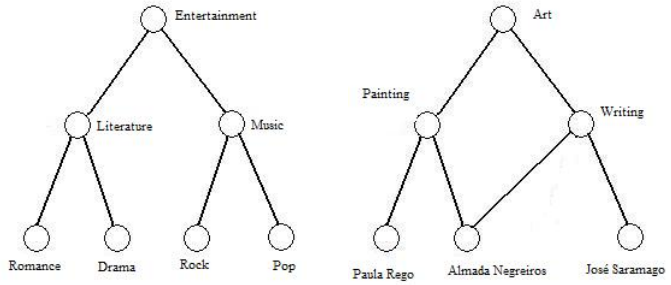
Figure 2.1: Illustration of how document classification is organized.

To perform the multi-class classification one of two strategies is usually followed, namely: *one-against-all* or *one-against-one*. In *one-against-all* the set of classes is divided in two sets namely, class c_j and its complement $\bar{c_j}$. This way, the classifier performs a simple binary classification task and estimates if the document belongs or not to class c_j. In the *one-against-one* strategy several comparisons are performed for each class, i.e. each class is tested against all the other classes. Only after performing all the possible comparisons does it become possible to decide which class is assigned to a document.

The previous classification tasks assign one class to each document, corresponding to what is called single-label classification tasks. Another possibility is a multi-label classification, where several classes can be assigned to a document. Normally, assigning a single class per document is done by assigning the class that gets the highest probability. In multi-class classification, the assignment criteria can be one of two, namely (i) classes that obtained a probability higher than a minimum threshold, or (ii) the n-classes with the highest probability values.

Multi-class classification tasks can also be flat or hierarchical. In the hierarchical scenario, classes are grouped by topics and the lower classes represent a specialization of upper classes (e.g. music and literature could be descendant of a general class like entertainment). The hierarchy of classes can be represented in different ways. We can have a tree, where each class has only one parent node, like in Fig. 2.2(a). In this case, documents can either be only in the leaf nodes (virtual category tree) or documents can be at any level of the hierarchy (category tree). We can also have a Directed Acyclic Graph (DAG), where a node can have more than one parent

node (Fig. 2.2(b)). The DAG structure can also have documents only in the leaf nodes (virtual Directed Acyclic Graph) or in any any node (Directed Acyclic Graph).



(a) (b)

Figure 2.2: Examples of hierarchical structures: (a) tree structure, (b) DAG structure.

There are several approaches to implement hierarchical classifiers. Common approaches are (i) flatten the hierarchy, (ii) use a multi-class classifier in each level of the hierarchy, (iii) use a top-down strategy or (iv) use the “big bang” strategy.

In the flat hierarchy strategy we transform a hierarchical classification task into a multi-class classification task. This transformation is obtained with a multi-class classifier with all the classes in the hierarchy that have any document, e.g., the leaf nodes in a virtual category tree scenario. In this case any possible relation between classes will not be considered.

We can also use multi-class classifiers for each level of the hierarchy. With this strategy, we also lose the relation between classes, although the number of classes for each classifier is potentially smaller.

Another way to implement hierarchical classification is with a top-down strategy. This consists in a tree of classifiers, with a classifier in each node. It allows the usage of different classification models in each node of the hierarchy, but if an error occurs in a higher level of the hierarchy, that error will be propagated to the lower levels.

A more complex strategy is the “big bang” approach. This strategy is similar to the flatten hierarchy approach, as it uses a multi-class classifier. However, in the “big bang” approach, each class represents a possible node in the hierarchy, whereas the flat hierarchy approach is only appropriate for problems where the classes correspond to leaf nodes.

Table 2.1: Confusion matrix for binary classification tasks.

		Predicted	
		true	false
Expected	positive	tp	fp
	negative	fn	tn

2.1.3 Evaluation Metrics

The performance and behavior of a classifier can be assessed in terms of the quality of the produced classifications. One way to visualize the behavior of a classifier is with a confusion matrix, showing in the rows the expected classification results and in the columns how the documents where actually classified. For instance, if we have a binary classification task, we obtain a confusion matrix with dimensionality 2×2. The matrix separates documents that are true positives (i.e., documents classified as true that are positive), false positives (i.e., documents classified as false that are positive), false negatives (i.e., documents classified as true that are negative) and true negatives (i.e., documents classified as false that are negative).

Once the values of the elements in the confusion matrix are identified, we can compute measures like *accuracy*, *Precision* or *Recall*. *Accuracy* represents the percentage of documents that were correctly classified in the dataset (Equation 2.6). *Precision* is the percentage of positive documents that were classified as true (Equation 2.7). *Recall* is the percentage of documents classified as true that are really positive examples (Equation 2.8).

$$\text{Accuracy} = \frac{tp + tn}{tp + fp + fn + tn} \tag{2.6}$$

$$\text{Precision} = \frac{tp}{tp + fp} \tag{2.7}$$

$$\text{Recall} = \frac{tp}{tp + fn} \tag{2.8}$$

These measures are only defined for binary classification problems. To evaluate a multi-class classifier, they can be replaced by micro and macro averages. To calculate the micro-averages we assume that C_i is the number of documents in class *i*, C'_i is the number of documents that were classified as class *i* documents and n is the number of possible classes to assign each document. We then have that:

$$\text{Micro-average Precision} = \frac{\sum_{i=0}^{n} C_i \cap C'_i}{\sum_{i=0}^{n} C'_i} \tag{2.9}$$

$$\text{Micro-average Recall} = \frac{\sum_{i=0}^{n} C_i \cap C'_i}{\sum_{i=0}^{n} C_i} \tag{2.10}$$

Macro-averages are the mathematical averages of *Precision* and *Recall*. Let P_i and R_i be respectively *Precision* and *Recall* of class *i*. We then have that:

$$\text{Macro-average Precision} = \frac{1}{n}\sum_{i=0}^{n} P_i \tag{2.11}$$

$$\text{Macro-average Recall} = \frac{1}{n}\sum_{i=0}^{n} R_i \tag{2.12}$$

Usually, research results do not discuss *Precision* and *Recall* in isolation (Bennett & Nguyen (2009); Sebastiani *et al.* (2000); Xue *et al.* (2008)). Instead, both measures are combined into a single evaluation measure, the F_1-measure, which is the weighted harmonic mean between *Precision* and *Recall* (Equation 2.13).

$$F_1\text{-measure} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.13}$$

All the previous measures deal with different types of errors in the same way. However, when we analise a hierarchical classifier, the errors can occur due to different reasons and they can have either severe or very light impacts on the quality of the results.

Previous works, like those of Costa *et al.* (2007) and Sun & Lim (2001), presented metrics specifically designated to evaluate hierarchical classifiers. Four kinds of measures were identified, namely: (i) distance based measures, (ii) depth based measures, (iii) semantic based measures and (iv) hierarchy based measures

Distance based measures assume that a mismatch between close classes in the hierarchy is not as severe as a mismatch between far classes. The distance between classes is usually measured through the number of edges in the path from one class to another. These distance based measures do not make distinguish along hierarchy depth, and while we go deeper in the hierarchy the number of documents per class gets lower, becoming harder to do a correct classification. Thus, distance based measures can be extended to depth based measures. In the case of depth-based measures, the edges between classes have an associated weight that considers the depth in the hierarchy, so that the deeper in the class hierarchy the lower is the weight associated to the edges.

The previous measures based on distance assume that mismatches between similar classes are not severe. Instead of distance or depth considerations, we can also evaluate a mismatch

classification based on the semantic similarity between classes. Similarity between classes can be estimated according to the feature vectors that describe classes C_i and C_j. Sun & Lim (2001) used the cosine distance between the feature vectors to estimate the semantic similarity between classes (Equation 2.14).

$$C_i = \{w_1 t_1, w_2 t_3, ..., w_N t_N\}$$

$$C_j = \{v_1 t_1, v_2 t_3, ..., v_N t_N\}$$

$$ClassSimilarity(C_i, C_j) = \frac{\sum_{n=1}^{N}(w_n \times v_n)}{\sqrt{\sum_{n=1}^{N} w_n^2 \times \sum_{n=1}^{N} v_n^2}} \tag{2.14}$$

Finally we can also evaluate the performance of an hierarchical classifier with basis on the hierarchical level. These measures have two possible approaches, as we can (i) consider the descendants, or (ii) consider the ancestors of the predicted and the true classes. In the descendants approach we consider the subtrees rooted in the true class and in the predicted class. The calculation of hierarchical *Precision* and *Recall*, is based on the intersection between the previous subtrees. Consider Descendant(C_p) the subtree rooted in the predicted class and Descendant(C_t) the subtree rooted in the true class. *Precision* is obtained by dividing the number of common classes by the number of classes in the subtree rooted in the predicted class (Equation 2.15).

$$\text{hP} = \frac{\mid \text{Descendant}(C_p) \cap \text{Descendant}(C_t) \mid}{\mid \text{Descendant}(C_p) \mid} \tag{2.15}$$

Recall is obtained by dividing the number of common classes by the number of classes in the subtree rooted in the true class (Equation 2.16).

$$\text{hR} = \frac{\mid \text{Descendant}(C_p) \cap \text{Descendant}(C_t) \mid}{\mid \text{Descendant}(C_t) \mid} \tag{2.16}$$

When we consider the ancestor classes of the predicted class and true class, we calculate the number of common classes that are ancestors of the predicted class and the true class. We have that Ancestor(C_p) is the set of classes that are ancestors of the predicted class and Ancestor(C_t) is the set of classes that are ancestors of the true class. If we divide the number of common classes by the number of ancestors of the predicted class we have *Precision* (Equation 2.17).

$$\text{hP} = \frac{\mid \text{Ancestor}(C_p) \cap \text{Ancestor}(C_t) \mid}{\mid \text{Ancestor}(C_p) \mid} \tag{2.17}$$

If we divide the number of common classes by the number of ancestors of the true class we have

the value for *Recall* (Equation 2.18).

$$\text{hR} = \frac{|\ \text{Ancestor}(C_p) \cap \text{Ancestor}(C_t)\ |}{|\ \text{Ancestor}(C_t)\ |} \tag{2.18}$$

2.2 Text Classification Algorithms

Automatic text classification is usually seen as a supervised machine learning problem. Figure 2.3 illustrates the automatic classification process, where in a training stage we start by generating a classification model based in some previously classified examples, also known as the training set (steps 1 and 2). Once generated, the classification model can then be tested. A second set of documents is classified and measures like *Precision*, *Recall* and *Accuracy* are calculated to estimate the classifier performance (steps 3 and 4). Finally, the model can be used to classify a previously unseen document, returning the predicted class for that document (steps 5 and 6). We have different families of classification algorithms including boosting algorithms like Ada Boost, decision tree approaches like ADTree, probabilistic algorithms like Naïve Bayes, or statistical algorithms like Support Vector Machines. In this document we will discuss the last two.

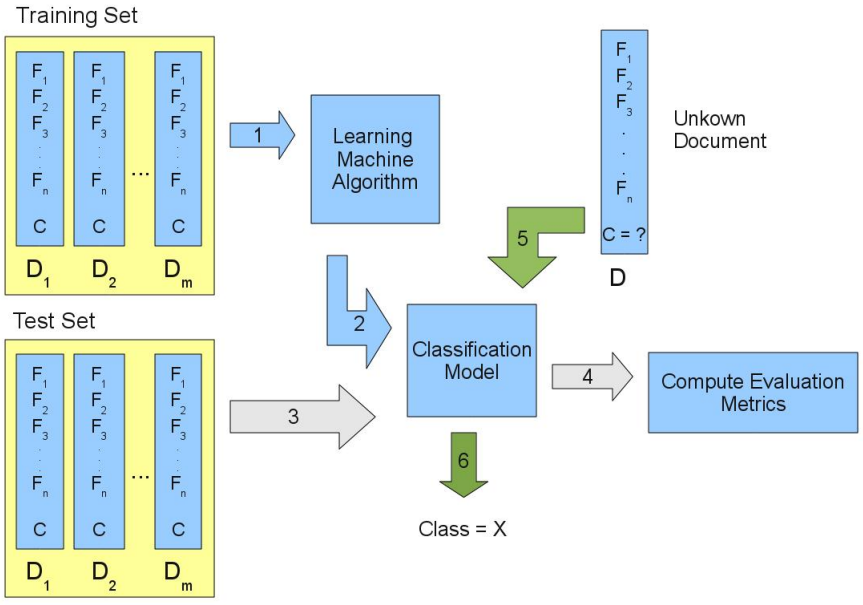


Figure 2.3: General approach for automatic document classification.

2.2.1 Naïve Bayes Classifiers

Naive Bayes is a probabilistic classification algorithm that builds a classification model based on the presence or absence of specific features. Using the Bayes theorem, Naive Bayes classifiers estimate the probability of class c_i given a document x, as shown in Equation 2.19. In the formula, c_i is the i-*th* class of the set of classes C and X is the variable that assumes the values of the feature vector $x = (x_1, x_2, ..., x_m)$ representing a document.

$$P(C = c_i \mid X = x) = P(C = c_i) \times \frac{P(X = x \mid C = c_i)}{P(X = x)} \tag{2.19}$$

These classifiers assume that features are independent, i.e., the presence or absence of a feature is unrelated to the presence or absence of another (Rish (2001)). This assumption allows us to calculate the probability of a document x given a class c by Equation 2.20, where f_j represents the *j-th* feature in the document collection.

$$P(X = x \mid C = c_i) = \prod_{j=0}^{m} P(f_j \mid C = c_i) \tag{2.20}$$

Once all the possible values for Equation 2.19 are computed, the Bayes classifier performs the class assignment following the decision function given by Equation 2.21, where the assigned class is the one with the highest probability.

$$\underset{i}{argmax}(\frac{P(X = x \mid C = c_i)}{P(X = x)} P(C = c_i)) \tag{2.21}$$

The decision rule in Equation 2.21 can also easily be extended to multi-label classification tasks, following one of the multi-label classification strategies presented in Section 2.2.

Naive Bayes classifiers are easy to understand and their implementation is simple and fast, due to the feature independence assumption (Rennie *et al.* (2003)). This fact makes Naive Bayes classifiers a common baseline for document classification (Lewis (1998)).

2.2.2 Support Vector Machines

Support Vector Machines (SVMs) are a binary classification method proposed by Vladimir Vapnik with basis on statistical learning theory (Vapnik (1982)).

SVM classifiers produce a separation between classes through an hyperplane estimated in the training stage. SVMs predict to each one of the two classes a new document belongs, based

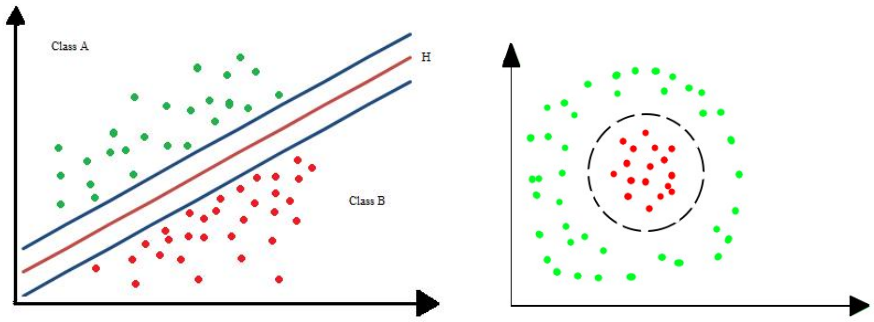


Figure 2.4: Examples of possible binary classification tasks: (a) classes that are linearly separable, (b) classes that are not linearly separable.

on which side of the hyperplane the given document is placed. The resulting hyperplane is also called the *optimal hyperplane*, because it is the one that separates classes with the maximum possible margin (Cristianini & Shawe-Taylor (2000)).

It is interesting to notice that not all classification problems are linearly separable through an hyperplane - see Figure 2.2.2. SVMs address this issue by mapping the objects from the input feature space to a high-dimensional space, where the objects are linearly separable. This is done through the use of a kernel function. Some of the well-known kernel functions are shown bellow, where x_i and x_j are two objects from the input space, d is the degree of the polynomial function and γ a parameter to control the shape of the separating hyperplane.

- Linear kernel: $K(x_i, x_j) = x_i^T x_j$.
- Polynomial kernel: $K(x_i, x_j) = (x_i \cdot x_j)^d$.
- Radial basis function kernel: $K(x_i, x_j) = \exp(-\gamma \left\| x_i - x_j \right\|^2)$.

More formally, consider a dataset D, with a set of n points, where c_i is either 1 or -1, representing the class that each point x_i belongs to. Each x_i is a *n*-dimensional vector. We can define any hyperplane as the set of points x that satisfies the Equation:

$$\langle w \cdot x \rangle + b = 0 \tag{2.22}$$

$$D = \{(x_i, c_i) \mid x_i \in \mathbb{R}^n, c_i \in \{-1, 1\}\}_{i=1}^n$$

In Equation 2.22, w is a normal vector perpendicular to the hyperplane, and b is the distance between the origin and the hyperplane. This equation divides the space in two regions: (i) one where $\langle w \cdot x \rangle + b < 0$, and (ii) one where $\langle w \cdot x \rangle + b > 0$. As a decision function $f(x)$ we can have

the signal function given by Equation 2.23.

$$f(x) = sign(\langle w \cdot x \rangle + b) \tag{2.23}$$

The process to estimate the *optimal hyperplane* is better explained by Cristianini & Shawe-Taylor (2000). Here we present only the final equation of the *optimal hyperplane* w. The *optimal hyperplane* can be obtained through Equation 2.24, where α_i is the i-*th* Lagrange multiplier.

$$w = \sum_{i=1}^{n} \alpha_i c_i x_i, \alpha_i \in \mathbb{R} \tag{2.24}$$

Thus, we can replace w in Equation 2.23, resulting in

$$f(x) = sign\left(\sum_{i=1}^{n} \alpha_i c_i \langle x \cdot x_i \rangle + b\right). \tag{2.25}$$

In order to address problems that are not linearly separable, we include a kernel function that given an input of two vectors returns the dot product of the images in the feature space.

$$\mathsf{k}(x, x_i) = \langle \phi(x) \cdot \phi(x_i) \rangle \tag{2.26}$$

Considering the kernel function together with the decision function, we finally get:

$$f(x) = sign\left(\sum_{i=1}^{n} \alpha_i c_i \mathsf{k}(x, x_i) + b\right). \tag{2.27}$$

2.3 Related Work

The information retrieval and machine learning literature is rich in approaches designed specifically to take advantage of class taxonomies in document classification. This section presents a survey on these previous works, which during the development of this work represented the state of the art in hierarchical Web documents classification.

Previous works (Cai & Hofmann, 2004; Labrou & Finin, 1999) to solve the hierarchical document classification problem tried to directly adapt known classification algorithms, although Liu *et al.* (2005) and Madani *et al.* (2007) have proven that this approach it is infeasible for a large-scale hierarchies.

Dumais & Chen (2000) tried to take advantage of an hierarchical structure of classes to classify documents in a large collection of very heterogeneous Web content. An hierarchy with two levels was considered, using the first two levels present in the LookSmart Web directory[1]. The authors used Support Vector Machines and two different classification approaches were tested. In the first, a classifier learns to distinguish second level categories between other second level categories with the same top level category (hierarchical case). In a second approach, the classifier learns to distinguish second level categories within all the 150 second level categories (flat non-hierarchical case). Results showed small advantages in the F_1 accuracy score for the hierarchical models, when compared with a baseline flat non-hierarchical model. This suggests that its easier to distinguish between a restrict number of classes related to the same top level class, than between all the classes in the same hierarchy level.

As previously noted, a common approach in hierarchical text classification involves associating independent classifiers with nodes in the category hierarchy and classifying text documents in a top-down manner. With this approach, documents can be wrongly rejected by the classifiers at higher-levels, this way never being passed to the classifiers at lower-levels. Sun & Lim (2001) proposed three methods to address the problem:

- **Threshold Reduction**, which is based on the principle that using lower thresholds for the upper subtree classifiers will allow more documents to be passed to the classifiers at lower-levels (i.e., it relaxes the decision functions of subtree classifiers along the hierarchy, allowing more documents to pass through). Nonetheless, higher classification accuracy will now be required for the classifiers at lower-levels to prevent misclassification.

- **Restricted Voting**, which gives low-level classifiers a chance to access documents before they are rejected by the subtree classifiers of their parent nodes. This method allows the local classifier of a node to receive documents from the subtree classifier of its grandparent node if they are accepted by a secondary classifier associated with the grandparent node.

- **Extended Multiplicative**, which is derived from the multiplicative method proposed by Dumais & Chen (2000). In this method, a document is accepted by the local classifier of a node if the combined probabilities of i) the local classifier, and ii) its hierarchical ancestor classifiers, is above a given threshold defined empirically.

Experiments with SVM classifiers on the Reuters-21578 collection have shown that all the three methods described above can improve the results in terms of accuracy, with the restricted voting method performing best (Sun & Lim (2001)).

[1] `http://www.looksmart.com`

Using the approach based on associating independent classifiers with nodes in the category hierarchy, it is also important to note that the different classifiers can (i) be based on the same learning method, although constructed using different sets of features, or (ii) be based on different learning methods with the same or different feature sets. Sun & Lim (2001) proposed an heuristic to select training instances for individual Bayes classifiers associated with each of the nodes of the category hierarchy. Other authors have experimented with the use of feature selection mechanisms for each of the independent classifiers, showing that, locally, using only a small number of discriminative features is sufficient to achieve reasonable classification accuracy (Dumais & Chen (2000); Koller & Sahami (1997); Mladeni'c & Grobelnik (1998)).

Xue *et al.* (2008) proposed a new approach to classify documents in a large scale hierarchy, similarly to Sun & Lim (2001). The motivation of the proposed approach was also reduce the error propagation from the top levels in the hierarchy to the lower levels. The authors proposed a two stage based approach. The first stage is called the search stage, and the second is called the classification stage. In the search stage, the hierarchy is organized into flat categories, where the algorithm performs a search process over the large-scale hierarchy and retrieves the related categories for a given document. This stage prunes the large scale hierarchy to a smaller subset of categories. The retrieved categories are ranked and the most related categories are seen as category candidates. In the classification stage, a classification model is trained on the category candidates. This classification model will be trained in a smaller subset of categories, reducing training time and computational cost.

Both stages, search stage and classification stage, were tested with distinct strategies. In terms of strategies for the search stage, the authors experimented with:

- **Document based**: This strategy compares the relevance between the given document and the documents in the training set. In the end, the top N most similar categories are retrieved.
- **Category based**: This strategy represents each category by a feature-vector with term frequencies. The feature vector is build with basis on the positive examples of each class in the training set. A given document is compared with each category and again the N most similar categories are retrieved and selected has candidate categories.

In both cases, the authors used the cosine distance to measure similarities. The classification stage was also experimented with two distinct strategies:

- **Flat Strategy**: In this strategy all the category candidate were considered as a flat structure, without considering the category information of their ancestors.

- **Pruned Top-down Strategy**: In this strategy the authors considered hierarchy information following two observations. First, the training data from the category candidate can be insufficient, especially in a deep category. Second, although the training data from its higher up ancestors may be too general to detect characteristics of a deeper category, we can borrow data from the ancestors. This should not be done for ancestors that are too high up, and common to more than one of the retrieved candidate category.

To access the results of the proposed methods, three algorithms were compared over a dataset collected from the Open Directory Project. The three tested algorithms were (i) hierarchical SVM, to represent a top-down classification algorithm, (ii) search based strategy, as described previously and similarly to a nearest neighbor approach, and finally (iii) the deep classification method proposed by Xue *et al.* (2008).

The time complexity was estimated like in Yang *et al.* (2003) and results shown that the proposed algorithm is scalable and can achieve 77.7% improvement, over the top-down SVM classification algorithm, in the 5th level on the large scale hierarchies. Results also shown that a low number of candidate categories brings better results in upper categories, but poor results in lower categories. Increasing the number of candidate categories reduces the performance in upper categories, but the gain in lower categories is higher than the loss in upper categories.

Bennett & Nguyen (2009) also developed an approach to solve the error propagation issue, similar to that of Xue *et al.* (2008). Not only the error propagation was the motivation for this work but also the increasingly complex decision surfaces, also called nonlinearity. This nonlinearity is a question related with the upper classes in the hierarchy that often contain general concepts, e.g., Kids & Teens, that are difficult to discriminate because they cover very diffuse topics. To solve the problem related with the error propagation, the authors proposed a strategy called *Refinement*, which consists in modifying the training set. Instead of the standard training set, used in hierarchical document classification, a cross-validation over the training was performed and the predicted labels are used to filter the training data to a node. This filtering would remove some false negatives from the training data and classification results would probably suffer some degradation. Once the quality of the classification results depend more from the positive examples than the number of examples. The training data was composed by a union of the standard training data with the predicted errors.

To deal with the nonlinearity issue Bennett & Nguyen (2009) adapted the methods of meta-classification and combination of classifiers, used by Bennett *et al.* (2005). Predictions from the lower levels in the hierarchy were introduced as meta-features to the higher levels. This was done by training linear classifiers at the leaf nodes, using cross-validation over the training data,

and then using the predictions over the training data gathered during cross-validation as meta-features available at the next higher level. The amount of information for each node was restricted to predictions from a node's children and their cousins.

The previous approach together with *Refinement* suggests a bottom-up training pass, that helps drive documents down the right branch, followed by a top-down training pass that prevents the error propagation. This method was called *Refined Experts* because the bottom-up training pass is seen as a passing up guesses from specialized classifiers.

The two classification methods, proposed by Bennett & Nguyen (2009), were compared against a baseline classifier, namely hierarchical SVM. Results showed that *Refinement* outperforms the baseline classifier and *Refinement Experts* outperforms both with macro-F_1 values about 36% and micro-F_1 values about 46%.

Some documents are not easy to classify, e.g., when we use a SVM classifier the document ends placed very close of the estimated hyperplane in the training stage. When such thing happens probability of a false positive is higher. To improve classification results we can make use of auxiliary classifiers (ying Jia *et al.* (2009)). ying Jia *et al.* (2009) proposed a technique that consists in having a main classifier that returns a predicted class with some confidence interval. If the confidence interval is lower than a desired threshold an auxiliary classifier is used to see if the initial prediction is correct or not. The authors tested a setup with two classifiers (i) a centroid based method as the main classifier and a K Nearest Neighbor and a SVM as auxiliary classifiers and (ii) a centroid based as the main classifier and SVM as auxiliary classifier. Experiments results showed that this classification approach achieves better performance than single classifiers, not only in flat classification but also in hierarchical classification. The increase in time complexity is small.

The methods discussed above are all based on transforming the hierarchical classification problem into a set of multi-class (or binary) classification problems, where an independent training process is used for each of the simpler classification tasks. However, several authors have noticed that this may lead to suboptimal discrimination, since the classifiers are finally operating in a specific architecture that combines their output based on heuristics. Previous works have, for instance, proposed kernel based methods (e.g., generalizations of Support Vector Machines exploring the hierarchical structure of the classes) that directly address the hierarchical classification problem (Rousu *et al.* (2006), Cai & Hofmann (2004) and Liu *et al.* (2005)). However, these methods are often associated with an increased complexity, and there are few readily available software packages that implement them. Moreover, the quality of the results obtained with these methods is also far from optimal, and data sparsity remains an important issue (Liu *et al.* (2005)).

2.4 Summary

Chapter 2 surveyed the literature concerning the development of classification systems able to classify documents accordingly to a topic hierarchy.

One important task in the field of the automatic classification is the representation of the documents, and typically documents are represented by feature vectors. The dimensions of those vectors represent the individual features from the document collection (*e.g.*, the words or n-grams), and thus they are often of high dimensionality. The dimensions of those vectors can be weighted according to many approaches, such as *tf*idf* or just counting the number of occurrences in each document.

The literature concerning the hierarchical classification of documents describes three general possible approaches, each one of it with their advantages and issues. The approach traditionally used by the research community is the *top-down* approach. This approach creates a hierarchy of classifiers where, typically, each classifier represents a node in the hierarchy. One advantage of this approach is that we reduce a complex task, that is the hierarchical classification of a document, into several multi-class tasks.

We saw that some authors tried to directly adapt flat classification techniques to hierarchical classification problems (Cai & Hofmann, 2004; Labrou & Finin, 1999), but this approach is not feasible in hierarchical classification over large-scale hierarchies. In the work related to the hierarchical classification of documents, we have works like that of Dumais & Chen (2000) that compared flat strategies with hierarchical strategies and the results showed that the hierarchical strategies can achieve better results.

As previously noted, a common approach in hierarchical text classification involves associating independent classifiers with nodes in the category hierarchy and classifying text documents in a top-down manner. With this approach, documents can be wrongly rejected by the classifiers at higher-levels, this way never being passed to the classifiers at lowerlevels. Authors like Sun & Lim (2001) and Xue *et al.* (2008) proposed classification methods to address that issue. In the first (Sun & Lim, 2001), were proposed methods that (i) based on the principle that using lower thresholds for the upper subtree classifiers will allow more documents to be passed to the classifiers at lowerlevels, and (ii) gives low-level classifiers a chance to access documents before they are rejected by the subtree classifiers of their parent nodes. Xue *et al.* (2008) divided the classification problem in a two stage based approach, A search stage the classification stage, where the hierarchy is organized into flat categories, and the algorithm performs a search process over the largescale hierarchy and retrieves the related categories for a given document. And the second stage, the classification stage, where the classifier assigns to a document one of the

categories that were retrieved in the search stage.

Bennett & Nguyen (2009) also tried to solve the nonlinearity problem, that is related with the upper classes of the hierarchy, where we have classes that often contain general concepts, e.g., Kids & Teens, that are difficult to discriminate because they cover very diffuse topics. To address this issue Bennett & Nguyen (2009) proposed an approach that adapted the methods of meta-classification and combination of classifiers, used by Bennett *et al.* (2005). Predictions from the lower levels in the hierarchy were introduced as meta-features to the higher levels. This was done by training linear classifiers at the leaf nodes, using cross-validation over the training data, and then using the predictions over the training data gathered during cross-validation as meta-features available at the next higher level. The amount of information for each node was restricted to predictions from a node's children and their cousins.

Like in the work of the previous authors, in this MSc thesis we will have also a top-down approach to solve a hierarchical classification problem, having an independent classifiers in each node of the hierarchy. We will have classifiers that are able to assign more than one class to a document. The classifiers will follow the multi-label classification assigning the two most probable classes to a document, at each level of the hierarchy. Then we will expand each node and finally when the system achieves only leaf classes it decides for the most probable class.

Chapter 3

Hierarchical Classification of Web Documents

The classification of text documents has been focus of the research of many authors (Bennett *et al.*, 2005; Sebastiani, 1999), but there are differences when we deal with hierarchical Web document classification tasks. For start, the traditional text classification is typically performed with structured documents, in the sense that they contain to certain styles (*e.g.*, news articles) and are grammatically correct, while Web documents typically do not have such property.

In the context of this MSc thesis, we proposed three approaches to perform hierarchical classification tasks. In the first approach we wanted to see if the classification results improve when we perform simple feature selection while the system estimates the feature vectors. One motivation for this MSc thesis was to reduce the time spent to train the classifiers, an issue already noticed by other authors. To respond to this issue, we tested two methods that reduce the number of training documents. Another issue related to hierarchical document classification is the error propagation, caused by the misclassification in higher levels. To address that issue we proposed an extension to the traditional top-down approach that considers the two most probable classes in each level of the hierarchy.

In this chapter we have a description of the methods that were proposed to solve hierarchical classification problems. Section 3.1 presents the classification approaches that we want to test with this work, the traditional top-down approach and the proposed extension. Section 3.2 announces the feature selection methods that were followed at the time to estimate the feature vectors. Finally, Section 3.3 presents the two methods proposed to reduce the training data.

3.1 Classification Approaches

To classify the documents accordingly to the hierarchies represented at each dataset, we have some similarities. First, we start the classification over the higher levels of the hierarchy until we reach a leaf node. Second, the classification algorithm, used at each node of the hierarchy, is the same in both approaches. We used the Support Vector Machines implemented by Chang & Lin (2011). As kernel function we use the *radial basis function kernel* and the parameters C and γ had the default values of 0 and $1/number\_of\_features$, respectively.

3.1.1 Traditional Top-Down Approach

When we want to solve a hierarchical classification problem, this is usually the starting point. This approach simplifies the complex task, that is the hierarchical classification of documents, reducing it to several multi-class classification tasks where we have a reasonable among of classifiers already studied. This approach is also the baseline classification approach of this work. This

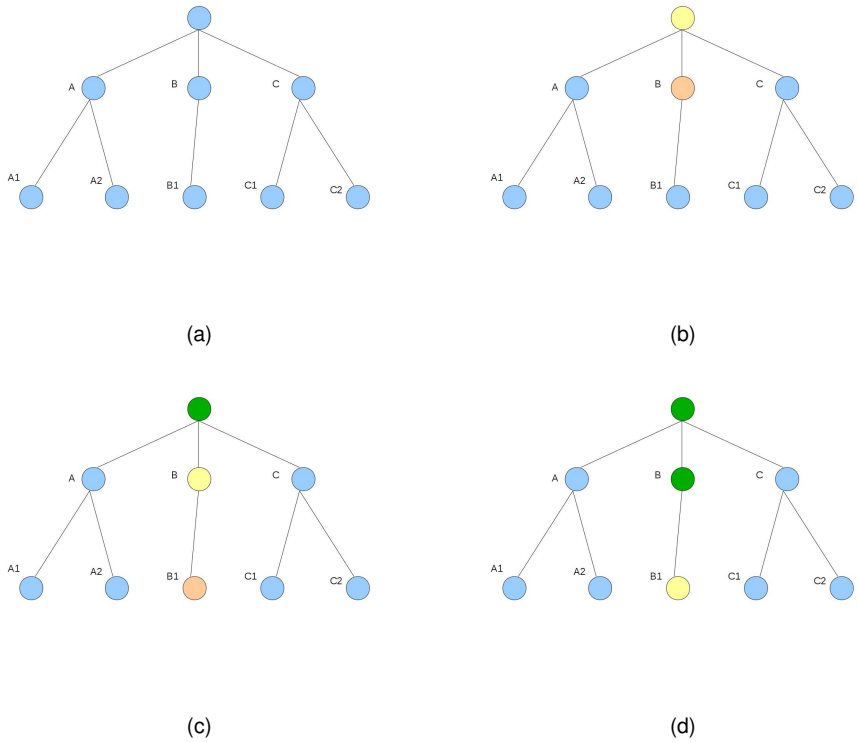


Figure 3.5: Illustration of the traditional top-down hierarchical classification approach: (a) initial hierarchy tree; (b) result of the classification over the first level; (c) result of the classification over the second level; (d) final classification path.

classification approach starts by classifying a document D at the root node to decide each one is the first level class. After that, the chosen node is expanded to decide between the second level classes. This process is repeated until the system achieves a leaf node of the hierarchy.

Figure 3.5 illustrates how the traditional top-down classification is processed. In Sub-Figure 3.5(a) we have the initial hierarchy where the system will expand the root node. Sub-Figure 3.5(b) shows the root node expanded, in yellow, and the decision for class B, in light orange. After this, the system finished the process over the root node, in the Sub-Figure 3.5(c) marked with green, expands the node B of the hierarchy, in yellow, and performs the classification over the second level of the hierarchy. Finally, we have processed the root and the node B of the hierarchy and the system will expand the node $B1$, the light orange circle in Sub-Figure 3.5(c). Once the node $B1$ is a leaf node the system will assign the class $B1$ to document D.

3.1.2 Alternative Top-Down Approach

A common issue in hierarchical classification tasks is the error propagation. This issue is caused by misclassifications in higher levels. When we use the traditional top-down approach we do not anticipate this issue and it is possible that the classification results stay far from desired. To address this issue, I propose a classification method that always considers the two most likely classes for each non-leaf level of the hierarchy of classes, *i.e.*, instead of the continuous expansion of one node in each level of the hierarchy, the classifier will expand the two most probable nodes in the hierarchy. In order to classify a document D, the classifier will find the two classes in the first level that are most likely to correspond to document D. After finding these two classes, the nodes in the hierarchy to which they correspond will be jointly considered in the next classification decision, which will be made with basis on the two child classed from both parents that are more likely to correspond to document D. This process is repeated for each level in the hierarchy until the classifier reaches leaf classes. At that moment, the most likely class will be the one signed to document D.

Figure 3.6 illustrates how the nodes in a hierarchy are expanded during the classification process of this hierarchical classification approach. To classify a document D over a hierarchy like the one present in Sub-Figure 3.6(a), the system will expand the root node returning the two classes, in the first level, that most fit to the given document, Sub-Figure 3.6(b) yellow and light orange nodes respectively. After this, the system considers the root node as a processed node, green node in Sub-Figure 3.6(c), and expands the two previously returned classes to classify the document D at the second level of the hierarchy, yellow nodes at Sub-Figure 3.6(c). The system will again consider the two classes that most fit, now at the second level, light orange node in Sub-Figure

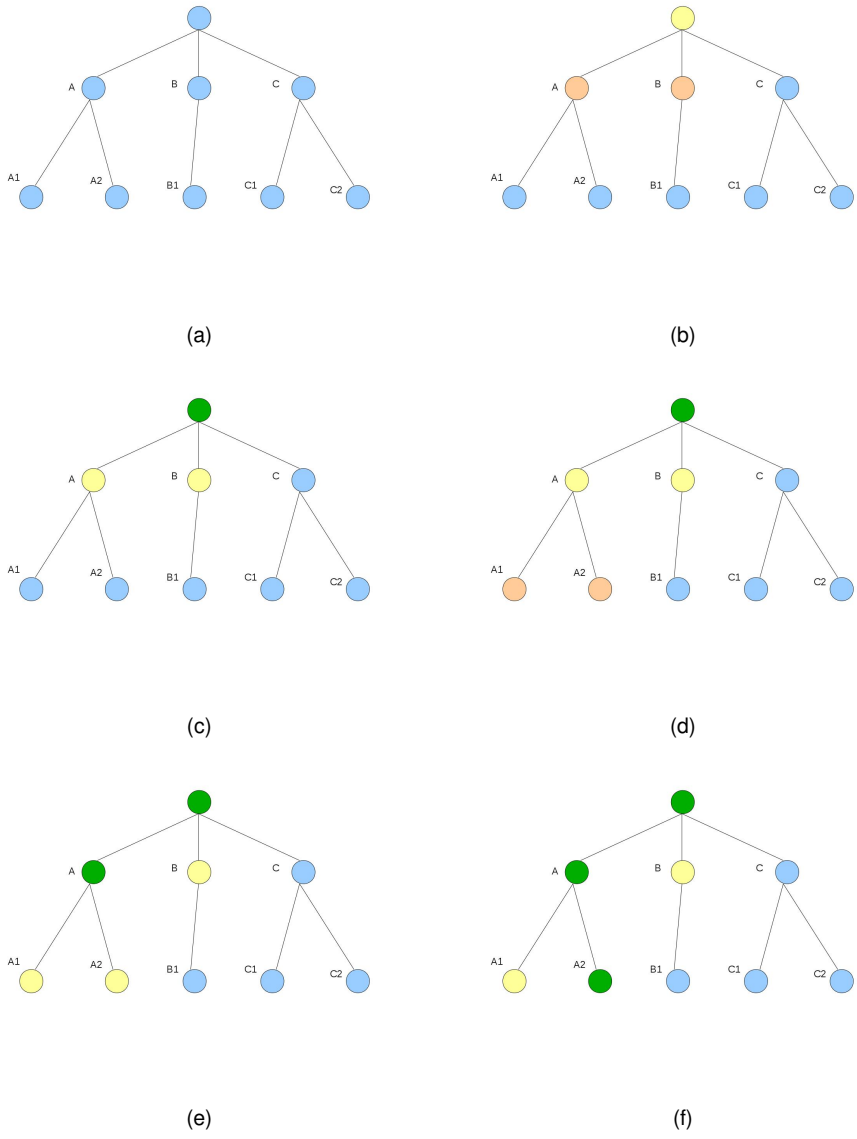


Figure 3.6: Illustration of the alternative top-down hierarchical approach: (a) initial hierarchy tree; (b) classification over the first level; (c) nodes to be expanded in the first level; (d) classification over the second level of the hierarchy; (e) nodes to be expanded in the second level of the hierarchy; (f) final classification path.

3.6(d). After the system founds the two classes, at the second level, that most fit to the document the system tries to expand those nodes, yellow nodes in Sub-Figure 3.6(e), because the two nodes are leaf nodes the system will assign, between the two classes, the class that most fits to the document D.

3.2 Feature Selection

We experimented with three approaches to build the feature vectors. These approaches were (i) the bag-of-words approach, where all the tokens in the text were considered, (ii) the stopword removal approach, where a list of words that do not add any semantic value, to the representation of the document, were not considered when feature vectors were calculated (see Appendix A), and (iii) the lemmatization approach, where besides stopword removal we have that all the remaining tokens were reduced to their stem accordingly to the algorithm proposed by Porter (1997).

The feature selection approaches were tested only in one of the two used datasets, the *4Universities* dataset. Only in that dataset we had the source code of the *html* pages. After removed the *html* tags, feature vectors were calculated with the API provided by the Weka Software (Hall *et al.*, 2009). To calculate the feature vectors we used the *StringToWordVector* filter. In the filter configuration terms were weight following the *tf-idf* heuristic, all the data was normalized and the text was tokenized by any punctuation or white space. Weka's API was used in a command line shell, were besides the input and output files also the configuration of each filter was defined.

The second dataset that was used in the experiments, the *Pascal Challenge* dataset, had already estimated the feature vectors. Each dimension of these vectors was also weighted accordingly to *tf-idf* heuristic.

3.3 Document Selection

The time spent in training a classifier is not only related with the number of classes and the dimensionality of the feature vectors, but also with the number of training documents. Some of the experiments performed in the context of this MSc thesis aimed at measuring the impact over the results if we reduce the number of documents in the training set. This reduction was implemented with two different methods. The first method, *naive*, ignores if in the selected documents we have examples from all the child classes of the considered node. The second method, *refined*, ensures that for each child class the training set will have at least one document.

Both document selection methods follow a top-down approach, *i.e.,* there were created sub datasets that represent the sub trees from the hierarchy, those trees will rooted in each node of the hierarchy that is not a leaf node.

3.3.1 Naive Document Selection

This was the first method proposed to perform the document selection. This method sets a limit to the number of documents for each child of the considered sub tree of the hierarchy, i.e., when we train the classifier for a non-leaf node N_i, we impose that the maximum number of samples taken from each node N_j that is a child node of N_i will be x. This document selection is performed over all the levels in the hierarchy for all the classes that are not leaf classes. The method is called naive because during this operation we do not care if we have documents from all the children of N_i or not.

To build the new training document collections the system considers each node in the hierarchy that is not a leaf node. Then, the system counts the number of child classes and starts one counter for each one of the children, each counter will control the number of documents per each class. After collected the initial information the system reads the training file and will select any document that belongs to one of the children classes of the considered node. A document is selected only if the counter of the document class did not reach the limit of documents.

To implement the document selection method, proposed in this sub-section, and described on the two previous paragraphs, the system makes use of a Java class that identifies the non-leaf nodes of the hierarchy, counts the number of children for each class, starts the classes counters and reads the input file sequentially. The input file is the text file made available to train the classifiers, and follows the *libSVM* format, each line of the file is a feature vector that represents a document.

3.3.2 Refined Document Selection

The second method proposed for document selection, refereed to as the *refined* method, not only tries to have documents from all the nodes N_j that are descendants of node N_i, but we also tries to have an equal number of documents in each child node. Imagine that we are training the first level classifier over the hierarchy present in Figure 3.7. First, we present the classifier with a number of examples equal to the suggested training sample size, e.g., fifty examples. In the example of Figure 3.7, we have two child classes, namely *literature* and *music*, and so the number of documents per each class will be the result of the division between the suggested number, fifty,

and the number of children of the node, two. That will give us twenty-five documents per each class at level two. For each child class of *entertainment*, which is not a leaf node in the hierarchy of classes, we will do the same operation but this time the suggested number will be twenty-five. The number of children of class *literature* is two, so we will divide twenty-five by two and we will have the number of documents for classes *romance* and *drama*. This operation is repeated until we reach only leaf classes. When result of the division is zero (i.e., the number of child classes is bigger than the number of suggested documents) the system will set the number of documents to one. Similarly to the first document selection method, this method is also performed for all the levels in the hierarchy, for each class that is not a leaf node. Thus, after training the first level classifier, the classifiers at the second level will repeat this operations with the initial suggested training sample size.

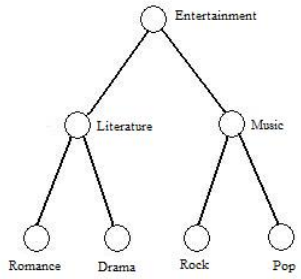


Figure 3.7: Example of a hierarchy of classes.

To implement the method described previously the system makes use of a Java class that has the same behavior of the class used to implement the naive document selection, but instead of a counter for each child node, we have a counter for each descendant node that is a leaf node. After started the counters, the system reads the input file sequentially and selects all the documents that are descendants of the considered node. This selection is performed while the counters did not reach the limit of documents for it class.

3.4 Summary

In Chapter 3, we describe a set of approaches to solve a hierarchical classification problem and the motivations for that proposals.

We start by announcing the baseline classifier and its configuration, a *top-down* approach with independent Support Vector Machines classifiers at each node of the hierarchy. The kernel function is the Radial Basis Function Kernel where the default values were assigned to parameters γ

and C.

This proposals were motivated by problem of the error propagation and the time spent to train the hierarchical classifiers, two issues reported by other authors. For the first issue we proposed an extension to the traditional *top-down* approach that consider the two more probable classes from each level of the hierarchy and in the end decides between the more probable leaf class. To reduce the time spent to train the classifiers we proposed methods to reduce the training sample and evaluate the impact of it, not only about the time spent to train the classifiers and classify a document but also in the quality of the experiments results.

In Chapter 4 we have the description of all of the steps taken to validate the approaches proposed in this chapter. That description includes a description of the used document collections, the considered evaluation measures and finally the results achieved by each one of the experiments.

Chapter 4

Validation Experiments

In this chapter we have a description of the experiments performed in order to validate the proposed approaches. We describe the datasets used during the experiments and present also the evaluation measures that were considered during the experiments. Finally, we present and discuss the results of each experiment.

4.1 Evaluation Datasets

To validate the approaches proposed in this MSc thesis, we used two datasets. These datasets are distinct in four dimensions, namely the number of documents, the dimensionality of the feature vectors, the number of classes, and finally the maximum depth in the hierarchy. The two datasets have also some similarities. The first similarity is the structure of the hierarchy, which is a tree, *i.e.* each class of the hierarchy has only one parent class. Another similarity is the fact that documents are only assigned to the leaf nodes. Table 4.2 details the differences between the two datasets.

Table 4.2: Differences between the datasets.

Dataset	Number of documents	Number of classes	Number of target classes	Max path size
4Universities	8282	42	35	2
Pascal Challenge	4463	2387	1139	5

The first dataset is the *4Universities* dataset[1], which has been used in many previous document classification experiments (Craven *et al.* (1998), Craven *et al.* (2000)). The second is the dataset from the first edition of the Pascal Challenge on Document Classification[2], a joint evaluation effort related to hierarchical classification of Web documents according to large scale hierarchies of classes.

In both datasets, that we used to validate the proposed classification approaches, we have that the documents are ordered by class. Given this, when we perform the document selection in the higher levels is very likely that we do not have documents from all of the leaf classes. This would reduce the sample to a short number of classes and can result in misclassifications in higher nodes. To avoid that, and before we start the train, we randomly choose the documents. With this operation we expect to have represented all, or almost all, of the leaf classes when the system trains the classifier.

4.1.1 The 4Universities Dataset

The *4Universities* dataset is a dataset with a total of 8282 *html* documents collected from computer science departments of various universities in United States at January 1997. The documents are organized in a two level hierarchy with 42 categories (35 of them leaf nodes). After processing the documents, the dataset presented a vocabulary with 140000 features, including all the words present in the text and the title fields of the *html* pages.

To build the feature vectors of the *4Universities* dataset some preprocessing steps were taken. First we need to extract the text and titles from the *html* documents. The extraction was performed with a Java class. After extracting the desired information, text and titles from the *html* documents were stored in a database table with the previously labeled class. After concluded this preprocessing step, we build the feature vectors with the *StringToWordVector* filter available in the Weka's API. This data filtering was made accordingly to the proposed methods in Section 3.2. Three approaches to build the feature vectors, namely the bag-of-words approach, the stopwords removal approach and the lemmatization approach were applied. To weight the features, the *tf-idf* heuristic was the selected method. During construction of the feature vectors all the text tokens were converted to lower case and the feature weights were normalized.

The *4Universities* dataset has no distinguish between train documents and test documents. In order to evaluate the performed experiments with the *4Universities* dataset, we split the dataset in 90% of the data to train the classifiers and 10% of the data to test the classifiers. The split is

[1] http://www.cs.cmu.edu/afs/cs/project/theo-20/www/data/
[2] http://lshtc.iit.demokritos.gr/

done with a Java class that reads the input file line by line and counts the number of documents per each class. After the counting we read again the input file line by line and select documents to the test set while the class limit is not reached. Once the class limit is reached the remaining documents are selected to be part of the training set.

4.1.2 The Pascal Challenge Dataset

The Pascal Challenge dataset was made available to the participants of the first edition of the Pascal Challenge, which required participants to solve a hierarchical classification problem. The feature vectors were already computed by the organizers of the Pascal Challenge. As in the first dataset, the documents were associated only to leaf classes in a hierarchy with a maximum of five levels. In the Pascal Challenge dataset, the documents are already split in train and test documents. To train the classifiers we have a total of 4463 documents distributed by a total 2387 classes where 1139 of them are leaf classes. The test dataset is composed by 1860 documents. Table 4.8 presents more statistical information about the data available to train the classifiers in the Pascal Challenge dataset.

Table 4.3: Number of classes per level in the Pascal dataset.

Level	Number of classes	Number of leaf classes	Number of documents	Average documents per class
1	10	0	0	0
2	158	0	0	0
3	522	41	128	3.12
4	846	247	632	2.56
5	851	851	3703	4.35

4.2 Evaluation Metrics

In order to evaluate the proposed approaches to solve an hierarchical classification task we performed a set of experiments using the datasets described in Section 4.1. Those experiments were evaluated according to a set of evaluation measures. In this section we describe which were those evaluation measures.

To compare the proposed classification approaches we use the standard measure *Accuracy*. However, like reported by other authors on their research (Bennett & Nguyen, 2009; Sebastiani *et al.*, 2000; Xue *et al.*, 2008), we discuss our results not only about the *Accuracy* score, but

also about the weighted harmonic mean of *Precision* and *Recall*, the *f-measure* (Equation 4.30). Like reported in Section 2.1.3 those measures need to be extended for multi-class classification problems. They can be extended in micro and macro-averages. In this work, we used the macro averages that are defined as follows in Equations 4.28 and 4.29. The definition is based on the number of true positives (TP), false positives (FP) and false negatives (FN) for each class c_i.

$$\text{Macro-average-Precision} = \frac{1}{n}\sum_{i=1}^{n}\frac{tp_i}{tp_i + fp_i} \tag{4.28}$$

$$\text{Macro-average-Recall} = \frac{1}{n}\sum_{i=1}^{n}\frac{tp_i}{tp_i + fn_i} \tag{4.29}$$

In the equations, used as remainders of the formulas already presented in Section 2.1.3, n represents the total number of categories (i.e., classes). With the above definitions of precision and recall, the F1 metric can be computed as shown in the formula bellow:

$$F_1\text{-measure} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4.30}$$

The validation of the approaches proposed in this MSc thesis is obtained not only with an analysis with the presented information retrieval measures but also measuring the time spent. This last evaluation measure is obtained with the command line utility *time*.

All the experiments were performed in a personal laptop with the following configuration:

- CPU - Intel Core I5-520m Dual-Core;
- Ram - 4Gb DDR3 1333;
- Unix Kernel - 2.6 64bits;
- Hard Disk - 500Gb SATA (7200rpm);
- Java Machine - jre version - 1.6.0-24

4.3 Results

In this section we report the results achieved by the classifiers in the experiments performed to validate the proposed classification approaches. Moreover, we discuss the results and try to interpret their meaning. We also compare our results with previous works and analise the differences between our work and the previous works.

4.3.1 4Universities Dataset

Tables 4.4 and 4.5 show the results obtained over the *4Universities* dataset, respectively with the traditional top-down classification approach and with the extended top-down classification approach. The tables have the results grouped by each considered feature selection method, and then the results are grouped by each document selection method. The columns *t-time* and *t-test* represent the time spent for training and testing the classifiers respectively. The column *DocSelection* identifies the method used to make the document selection, having the value of of *none* if we did not performed any document selection, *naive* for the first document selection method and of *refined* for the second method. The column *NumDocs* presents the number of documents to be selected in each method.

Table 4.4: Results over the *4 Universities* dataset with the traditional top-down approach.

Experiments without feature selection

DocSelection	NumDocs	Accuracy	Precision	Recall	F1-Measure	t-time	t-test
none	full	0.1155	0.0038	0.0333	0.0069	5m14s	26m41s
naive	100	**0.2493**	**0.1598**	**0.1302**	**0.1435**	**1m36s**	**2m27s**
naive	750	0.4177	0.0784	0.1244	0.0962	2m30s	10m55s
refined	1000	0.0835	0.0028	0.0333	0.0051	2m39s	1m04s
refined	4000	0.2482	0.0314	0.0707	0.0435	6m51s	10m18s

Experiments with stopwords removal

DocSelection	NumDocs	Accuracy	Precision	Recall	F1-Measure	t-time	t-test
none	full	0.1155	0.0038	0.0333	0.0069	4m45s	22m49s
naive	100	**0.2359**	**0.1598**	**0.1212**	**0.1378**	**1m22s**	**2m05s**
naive	750	0.4128	0.0818	0.1230	0.0983	2m22s	10m25s
refined	1000	0.0835	0.0028	0.0333	0.0051	0m17s	2m21s
refined	4000	0.2297	0.0241	0.0662	0.0353	1m23s	10m19s

Experiments with stems

DocSelection	NumDocs	Accuracy	Precision	Recall	F1-Measure	t-time	t-test
none	full	0.1155	0.0038	0.0333	0.0069	5m04s	21m38s
naive	100	**0.2150**	**0.1647**	**0.1208**	**0.1393**	**1m32s**	**2m02s**
naive	750	0.3821	0.0753	0.1143	0.0908	2m32s	9m48s
refined	1000	0.0835	0.0028	0.0333	0.0051	0m19s	2m25s
refined	4000	0.2396	0.0385	0.0683	0.0493	1m29s	10m12s

In a quick analysis over the results presented in Tables 4.4 and 4.5 we can see that the time taken to train and test the classifier is directly related with the number of documents used to train the classifiers, which corroborates the initial supposition. About the achieved *f-measures* it was

Table 4.5: Results over the *4 Universities* dataset with the alternative top-down approach.

Experiments without feature selection

DocSelection	NumDocs	Accuracy	Precision	Recall	F1-Measure	t-time	t-test
none	full	**0.8010**	**0.6429**	**0.6613**	**0.6520**	**25m30s**	**24m13s**
naive	100	0.0159	0.0040	0.1333	0.0077	7m17s	2m04s
naive	750	0.5971	0.5123	0.6536	0.5744	11m44s	30m47s
refined	1000	0.3931	0.3149	0.5865	0.4097	1m04s	3m19s
refined	4000	0.6609	0.5251	0.6996	0.5999	6m51s	10m18s

Experiments with stopwords removal

DocSelection	NumDocs	Accuracy	Precision	Recall	F1-Measure	t-time	t-test
none	full	**0.8206**	**0.6694**	**0.6725**	**0.6709**	**22m47s**	**21m23s**
naive	100	0.0160	0.0046	0.1333	0.0090	6m11s	1m49s
naive	750	0.5221	0.4563	0.5705	0.5071	10m54s	11m08s
refined	1000	0.4189	0.3652	0.5915	0.4515	0m53s	3m15s
refined	4000	0.6671	0.5183	0.6817	0.5889	6m02s	8m30s

Experiments with stems

DocSelection	NumDocs	Accuracy	Precision	Recall	F1-Measure	t-time	t-test
none	full	**0.7998**	**0.6466**	**0.6552**	**0.6509**	**25m08s**	**23m44s**
naive	100	0.0897	0.1182	0.2962	0.1690	6m37s	2m07s
naive	750	0.4865	0.4384	0.6117	0.5107	11m43s	11m41s
refined	1000	0.3955	0.3398	0.5809	0.4288	0m59s	3m36s
refined	4000	0.6290	0.5021	0.6890	0.5809	7m12s	9m33s

expected that the scores get worst because of the reduction of training data, but when performed the naive document selection results show improvements in the traditional top-down approach. That can be related with the features included in the selected documents that could be highly related to classes and that makes possible to better characterize each class.

When we compare the time spent to train the classifier between the feature selection methods we do not observe significant differences. Which give us the idea that the feature selection method does not implies a strong impact in the time spent and only in the behavior of the classifier.

The results achieved by the classifiers with the alternative top-down classification approach, proposed in this MSc thesis, validate the proposed method, still the time spent to train the classifiers increases in some cases five times, the *f-measure* achieves the score 0.6709 against the 0.1435 achieved in the traditional top-down approach.

Another observation that can be done with the results of Tables 4.4 and 4.5 is that feature selection do not imply results improvement. Feature selection can allow classifiers to achieve better

results depending on the classification approach and probably depending on the dataset.

4.3.2 Pascal Challenge Dataset

Before the presentation of the results obtained in the experiments performed in the context of this MSc thesis, is important to known what were the results announced by the organization of the the first edition of the Pascal Challenge. The results achieved in terms of *Accuracy* and *f-measure* are reported in Table 4.6 for the top five systems.

Table 4.6: Results announced by the organization of the Pascal Challenge, with participants systems ordered by *Accuracy*.

Participant	Accuracy	F1-Measure
alpaca	0.4676	0.3412
jhuang	0.4632	0.3549
arthur_general	0.4433	0.3197
XipengQiu	0.4431	0.3367
Turing	0.4317	0.3232

In Table 4.7 we have the results achieved by the proposed classification approaches in the experiments performed with the Pascal Challenge Dataset.

The results reported in Table 4.7 validate the proposed approach to extend the traditional top-down classification. The results are better when compared with the traditional top-down approach, since this method reaches better *f-measure* scores.

Results also show some relation of the time spent to train the classifiers with the number of documents, but there is one exception, which can be related with the CPU availability at the time we perform the experiment, the difference is smaller than thirty seconds in a total time about the thirty six minutes. There is also some increasing in the time spent to train the classifiers, when we compare the traditional top-down approach with the proposed alternative top-down approach, but there is also one exception, like the previous exception this can be related with the CPU available at the time.

Table 4.8 reports the Accuracy achieved by each proposed approach level by level, in this case the environments where were achieved the best f-measure results. We can see that the misclassifications at the higher levels are smaller with the alternative top-down approach. Resulting in a more accurate classification of the documents.

Finally we are able to compare the results of our experiments with the results announced by the organization of the first edition of the Pascal Challenge. The proposed approach achieves significantly better results in *Accuracy*, the best score announced is 0.4676, and in our experiments we

Table 4.7: Results with the Pascal Challenge dataset.

Experiments performed with traditional top-down classification

DocSelection	NumDocs	Accuracy	Precision	Recall	F1-Measure	t-time	t-test
none	full	**0.5439**	**0.1955**	**0.3257**	**0.2444**	**32m07s**	**7m17s**
naive	50	0.1459	0.0567	0.0921	0.0702	36m29s	1m58s
naive	100	0.2224	0.0766	0.1268	0.0955	36m56s	2m26s
naive	200	0.2919	0.1039	0.1688	0.1286	37m07s	3m23s
naive	300	0.3716	0.1365	0.2197	0.1684	37m33s	4m43s
naive	400	0.4642	0.1653	0.2676	0.2044	38m01s	5m21s
refined	100	0.2515	0.0454	0.0740	0.0563	37m00s	4m18s
refined	500	0.2698	0.0478	0.0759	0.0587	36m18s	4m21s
refined	1000	0.2709	0.0481	0.0758	0.0589	36m26s	4m26s
refined	2000	0.2876	0.0490	0.0762	0.0597	39m27s	4m36s
refined	3000	0.2967	0.0490	0.0765	0.0597	39m43s	5m58s

Experiments performed with alternative top-down classification

DocSelection	NumDocs	Accuracy	Precision	Recall	F1-Measure	t-time	t-test
none	full	**0.5977**	**0.3969**	**0.3258**	**0.3347**	**34m55s**	**8m37s**
naive	50	0.1836	0.0890	0.1248	0.1039	36m23s	2m59s
naive	100	0.2445	0.1139	0.1518	0.1302	37m12s	3m28s
naive	200	0.3183	0.1487	0.1990	0.1702	37m33s	4m15s
naive	300	0.4012	0.1979	0.2605	0.2249	38m39s	5m44s
naive	400	0.5024	0.2324	0.3163	0.2679	39m08s	6m32s
refined	100	0.2337	0.0505	0.0749	0.0603	37m03s	4m53s
refined	500	0.2693	0.0569	0.0764	0.0652	36m54s	5m11s
refined	1000	0.2784	0.0616	0.0814	0.0701	36m32s	5m23s
refined	2000	0.2924	0.0619	0.0812	0.0702	39m50s	5m57s
refined	3000	0.2983	0.0629	0.0812	0.0709	39m43s	5m58s

reach 0.5977. But the *f-measure* achieves lower scores, we have a maximum of 0.3347 and the organization announces 0.3549. This makes us believe that classes with few training documents are not well characterized by the classifier, to improve the classification results we would need to tune the parameters of our classifier, or make use of another classifier that can achieve better results with smaller training samples. Still the achieved *f-measure* is not better that the announced by the organization, the score 0.3347 would place our approach in the top four of the *f-measure* scores.

The short papers written by the participants of the challenge, that are available at the Web site, does not have the evaluation of the classifiers level by level, it would be interesting to see were the classification approaches start to produce distinct results, if it is at the top levels of the hierarchy, or if it is at the lower nodes.

Table 4.8: *Accuracy* over the Pascal Challenge dataset, for each hierarchy level.

Traditional Top-Down Approach

DocSelection	NumDocs	Level 1	Level 2	Level 3	Level 4	Level 5
none	full	0.9995	0.9575	0.8250	0.6809	0.6143
naive	400	0.8056	0.7728	0.6699	0.5766	0.5306
refined	3000	0.9892	0.9483	0.7891	0.6887	0.6181

Alternative Top-Down Approach

DocSelection	NumDocs	Level 1	Level 2	Level 3	Level 4	Level 5
none	full	1.0000	0.9731	0.8595	0.7208	0.6606
naive	400	0.8045	0.7851	0.6903	0.5981	0.5723
refined	3000	0.9914	0.9515	0.8066	0.7305	0.6767

From the short papers available at the Web site of the Pascal Challenge we have the short paper of *jhuang*, one of the participants that achieved better results than we. In his paper we have a description of the approaches developed to solve the hierarchical classification task, proposed in the Pascal Challenge. The approaches estimates a sparse matrix, called weight matrix. According to the author this matrix measures the relation if each feature with few classes. Authors report that the estimation of the weight matrix requires several passes over the training data, making us believe that is an approach that requires more resources than the approach proposed and evaluated in this thesis.

4.4 Summary

This chapter presented and discussed the results of each one of the experiments performed to validate the three proposed approaches to solve hierarchical classification problems. The main findings were as follows:

- The methods proposed to estimate the feature vectors do not have a meaningful impact in the classifiers behavior, the two classification approaches did not achieved the best results with the same feature vectors (vectors that were estimated following the same feature selection method).
- The document selection methods, proposed to reduce the time spent to train the classifiers, have a significant impact over that results dimension but the classification results can stay

far from the results achieved when we use the entire training dataset.

- Results show that the proposed extension to the traditional top-down classification approach is indeed effective in hierarchical classification problems.

Chapter 5

Conclusions

Automatic document classification is the general task of assigning one or more categories to an electronic document. This task is usually addressed through machine learning techniques, with basis on a compact representation of the document's contents. Recently, several authors have explored automatic classification in the context of Web documents.

In this MSc thesis the research focus was the automatic classification of Web documents according to a topic hierarchy. The common issues related to hierarchical classification tasks are: (i) time spent to train the classifiers, and (ii) the error propagation. To manage with these two issues were proposed two approaches. To manage with the first issue we proposed two methods to reduce the size of the training sample. To avoid the misclassification in the higher levels we proposed an extension to the traditional top-down approach.

To reduce the size of the training sample the first method performs a naive document selection, the system does not try to collect documents from all the leaf classes under the considered node. The system just reads the input file line by line and selects the documents to each class while the limit of documents per class is not reached. The second method to select the training documents tries to get at least one document per each class that is under the considered node in the hierarchy.

To avoid misclassifications in the higher levels we proposed an extension of the traditional top-down classification approach that expands the two most probable nodes in each level of the hierarchy, instead only the most probable like in the traditional top-down approach.

To evaluate the proposed approaches we considered the time spent during the experiments and some information retrieval measures traditionally used in multi-class classification environments,

namely *Accuracy*, *Recall*, *Precision* and *f-measure*. The three last by default do not support multi-class classification environments, so we used the macro-averages extensions of those measures.

The results validate the proposed extension to the traditional top-down approach, when compared with the results of the traditional top-down approach we see improvements in the experiments performed with two different datasets.

5.1 Contributions

This MSc thesis proposed some techniques to solve hierarchical classification problems. Each technique was individually and evaluated, producing the following main contributions:

- The results obtained in the experiments with both datasets validated, for the best of our knowledge, a simple modification to a usually followed approach. The error propagation, provoked by the misclassification in the higher levels of the hierarchy, one issue already identified by other authors, is attenuated with the proposed modification improving the classification results in all the experiment environments.
- Since one of the issues related to hierarchical classification is the time spent to train the classifiers with large scale hierarchies, to reduce the training time we proposed methods to reduce the training set and evaluated the results. Results show that still the reduction in the time spent. To the document selection were proposed two methods that in fact reduce the training time spent but only for small hierarchies, for large scale hierarchies the classifier needs even more time to be trained. The results do not validate this document selection proposal, not only because it is not guaranteed that we reduce the time spent to train the classifiers but also because the classification results suffer some degradation.
- One of the remaining challenges in automatic text classification is how to build the feature vectors. The experiments with the *4Universities* dataset show that different classification approaches achieve better results with different feature vectors.

5.2 Future Work

The conclusions about the proposed classification approaches were achieved with small relatively datasets, and it would be interesting to evaluate the developed system with bigger datasets, like the dataset made available for the second edition of the Pascal Challenge on Large Scale Hierarchical text classification.

Currently the system makes use of a Support Vectors Machine has classification algorithm, the classifier that is used by many authors in this research field, compare the results of the system with this setup with the same classification approach but using another classifier, such as Naïve Bayes, Decision Trees or AdaBoost, will ensure if the proposed classification approach in fact improves the classification results or not.

Bibliography

BENNETT, P.N. & NGUYEN, N. (2009). Refined experts: improving classification in large taxonomies. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, 11–18, ACM, New York, NY, USA.

BENNETT, P.N., DUMAIS, S.T. & HORVITZ, E. (2005). The combination of text classifiers using reliability indicators. *Information Retrieval*, **8**, 67–100, 10.1023/B:INRT.0000048491.59134.94.

CAI, L. & HOFMANN, T. (2004). Hierarchical document categorization with support vector machines. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, CIKM '04, 78–87, ACM, New York, NY, USA.

CECI, M. & MALERBA, D. (2007). Classifying web documents in a hierarchy of categories: a comprehensive study. *Journal of Intelligent Information Systems*, **28**, 37–78, 10.1007/s10844-006-0003-2.

CHANG, C.C. & LIN, C.J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, **2**, 27:1–27:27, software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

COSTA, E., LORENA, A., CARVALHO, A. & FREITAS, A. (2007). A review of performance evaluation measures for hierarchical classifiers. In *Evaluation Methods for Machine Learning II: papers from the AAAI-2007 Workshop, AAAI Technical Report WS-07-05*, 1–6, AAAI Press.

CRAVEN, M., DIPASQUO, D., FREITAG, D., MCCALLUM, A., MITCHELL, T., NIGAM, K. & SLATTERY, S. (1998). Learning to extract symbolic knowledge from the world wide web.

CRAVEN, M., DIPASQUO, D., FREITAG, D., MCCALLUM, A., MITCHELL, T., NIGAM, K. & SLATTERY, S. (2000). Learning to construct knowledge bases from the world wide web. *Artificial Intelligence*, **118**, 69 – 113.

CRISTIANINI, N. & SHAWE-TAYLOR, J. (2000). *An introduction to support Vector Machines: and other kernel-based learning methods*. Cambridge University Press.

DUMAIS, S. & CHEN, H. (2000). Hierarchical classification of web content. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '00, 256–263, ACM, New York, NY, USA.

HALL, M., FRANK, E., HOLMES, G., PFAHRINGER, B., REUTEMANN, P. & WITTEN, I.H. (2009). The weka data mining software: an update. *SIGKDD Explor. Newsl.*, **11**, 10–18.

KOLLER, D. & SAHAMI, M. (1997). Hierarchically classifying documents using very few words. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, 170–178, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

LABROU, Y. & FININ, T. (1999). Yahoo! as an ontology: using yahoo! categories to describe documents. In *Proceedings of the eighth international conference on Information and knowledge management*, CIKM '99, 180–187, ACM, New York, NY, USA.

LEWIS, D. (1998). Naive (bayes) at forty: The independence assumption in information retrieval. **1398**, 4–15, 10.1007/BFb0026666.

LIU, T.Y., YANG, Y., WAN, H., ZENG, H.J., CHEN, Z. & MA, W.Y. (2005). Support vector machines classification with a very large-scale taxonomy. *SIGKDD Explor. Newsl.*, **7**, 36–43.

MADANI, O., GREINER, W., KEMPE, D. & SALAVATIPOUR, M.R. (2007). Recall systems: Efficient learning and use of category indices. In *In International Conference on Artificial Intelligence and Statistics (AISTATS)*.

MLADENI'C, D. & GROBELNIK, M. (1998). Feature selection for classification based on text hierarchy. In *International Conference on Machine Learning*.

PORTER, M.F. (1997). *An algorithm for suffix stripping*, 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

RENNIE, J., SHIH, L., TEEVAN, J. & KARGER, D. (2003). Tackling the poor assumptions of naive bayes text classifiers. In *International Conference on Machine Learning*.

RISH, I. (2001). An empirical study of the naive bayes classifier. In *International Joint Conference on Artificial Intelligence*.

ROUSU, J., SAUNDERS, C., SZEDMAK, S. & SHAWE-TAYLOR, J. (2006). Kernel-based learning of hierarchical multilabel classification models. *J. Mach. Learn. Res.*, **7**, 1601–1626.

SALTON, G. & BUCKLEY, C. (1988). Term-weighting approaches in automatic text retrieval. vol. 24, 513–523, Pergamon Press, Inc., Tarrytown, NY, USA.

SEBASTIANI, F. (1999). A tutorial on automated text categorisation. In *Proceedings of ASAI-99, 1st Argentinian Symposium on Artificial Intelligence*, 7–35, Citeseer.

SEBASTIANI, F. (2002). Machine learning in automated text categorization. *ACM Comput. Surv.*, **34**, 1–47.

SEBASTIANI, F., SPERDUTI, A. & VALDAMBRINI, N. (2000). An improved boosting algorithm and its application to text categorization. In *Proceedings of the ninth international conference on Information and knowledge management*, CIKM '00, 78–85, ACM, New York, NY, USA.

SINGHAL, A., SALTON, G. & BUCKLEY, C. (1995). Length normalization in degraded text collections. Tech. rep., Ithaca, NY, USA.

SUN, A. & LIM, E.P. (2001). Hierarchical text classification and evaluation. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, 521 –528.

VAPNIK, V. (1982). *Estimation of Dependences Based on Empirical Data: Springer Series in Statistics (Springer Series in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

XHEMALI, D., HINDE, C. & STONE, R. (2009). Naive bayes vs. decision trees vs. neural networks in the classification of training web pages. *International Journal of Computer Science Issues(IJCSI)*, **4**, 16.

XUE, G.R., XING, D., YANG, Q. & YU, Y. (2008). Deep classification in large-scale text hierarchies. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, 619–626, ACM, New York, NY, USA.

YANG, Y. & PEDERSEN, J.O. (1997). A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, 412–420, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

YANG, Y., ZHANG, J. & KISIEL, B. (2003). A scalability analysis of classifiers in text categorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, 96–103, ACM, New York, NY, USA.

YING JIA, M., QUAN ZHENG, D., RU YANG, B. & XUAN CHEN, Q. (2009). Hierarchical text categorization based on multiple feature selection and fusion of multiple classifiers approaches. *Fuzzy Systems and Knowledge Discovery, Fourth International Conference on*, **1**, 192–196.

Appendix A

List of stopwords

The list with the considered stopwords, to be removed when computing the feature vectors, is as follows.

a	able	about	across	after	all
almost	also	am	among	an	and
any	are	as	at	be	because
been	but	by	can	cannot	could
dear	did	do	does	either	else
ever	every	for	from	get	got
had	has	have	he	her	hers
him	his	how	however	i	if
in	into	is	it	its	just
least	let	like	likely	may	me
might	most	must	my	neither	no
nor	not	of	off	often	on
only	or	other	our	own	rather
said	say	says	she	should	since
so	some	than	that	the	their
them	then	there	these	they	this
tis	to	too	twas	us	wants
was	we	were	what	when	where
which	while	who	whom	why	will
with	would	yet	you	your	