Automatic Text Categorization

from Information Retrieval to Support Vector Learning

A text book for courses in

Computer Science and Computational Linguistics

by

Roberto Basili and Alessandro Moschitti
October 30, 2005

University of Rome
Tor Vergata

Preface

In the Summer of 2001, at the International Joint Conference on Artificial Intelligence held in Seattle, we presented a text categorization system based on an extension of the empirical approach known as "Rocchio" classifier, fully inspired by the Information Retrieval literature. At that time, Rocchio seemed to represent the best choice for our research projects (categorization of documents from major news agencies, e.g. Reuters and Ansa) given its computational advantages, i.e. very fast learning and classification run time and its reasonable accuracy.

The Rocchio text classifier derives category profiles (e.g. foreign politics or sports) from a representation of the data in a metric space. Such profile building is the actual Rocchio training and relates to the extraction of statistical properties from a vast number of examples of already classified documents. Once profiles are derived, the classification of an incoming news item d is reduced to the process of computing the "similarity" of the vector representing d and the vector expressing the profile of a target class: when these vectors are enough "close" each other the system will accept d as a correct member of the class, otherwise d is judged to not belong to such class. In principle, the larger the set of the "training" examples is, the more accurate the categorization results are.

The major problem of the Rocchio approach is its weakness in producing accurate profiles. The two most important reasons are the following. First, there is no principled way to decide which examples are more important (and how) for the induction of the classification function (i.e. to determine the "closeness" property). Second, no method is available to determine how many examples are needed to reach a given (and possibly required) categorization accuracy. The area in which the two above problems are studied is the Statistical Learning Theory which aims to characterize example-driven learning tasks by determining their feasibility, inherent complexity, convergence criteria and expected error rate. When faced with realistic problems (e.g. automatic categorization systems for news agencies, digital libraries or other Web applications) the above pieces of information are relevant as they critically impact on the engineering choices of the system.

Such theory is also critical for the design of modern Natural Language and Text Processing applications as it provides analytical techniques that are essential to study problems related to coverage, accuracy, robustness and portability of automatic systems. Activities like data collection, manual annotation

and induction of linguistic resources have become essential in the design of any Natural Language application since late '80. Nevertheless, the characterization of such learning tasks is still problematic. Questions like "How many examples do I need to train my system?", "Which features are important for my learning task?", "How do I have to annotate my data?" receive different, competing and often diverging answers in the scientific literature. Moreover, the extremely interdisciplinary nature of the area does not help: linguists are usually not able to figure out how to make use of (or to give computational sense to) their insights on data whereas engineers may collect huge data sets but have not enough insights to make the best use of them.

Any scholar in Artificial Intelligence, Natural Language Processing, Information Retrieval, Computational Linguistics as well as Computer Science, in general, has an inherent interest into the theoretical and methodological results in Statistical Classification. Indeed, such results can be exploited to effectively design a quite large number of different applications ranging from Document Categorization, Spam Filtering to Part-Of-Speech Tagging, Word Sense Disambiguation, Question Answering, Syntactic Parsing and Semantic Role Labeling. This book tries to provide the basic notions of the statistical learning theory and its application to Text Categorization with a specific emphasis on engineering perspectives and principles of best practice valid for real scenario applications.

We think that Automatic Text Categorization is a prototypical problem for several other NLP tasks, thus we hope that the collection of ideas and the empirical evidence discussed in this book (originated by several years of research) will be a useful guide to heterogeneous student communities (e.g. computer science as well as computational linguistics). Although most of the questions on learnability of linguistic phenomena will still remain unanswered after reading this book, some of the results discussed here (i.e. successes and defeats) are strongly connected to many general natural language applications. Hence this book may serve as a guide for the problem solutions of innovative and complex natural language systems.

Roberto Basili and Alessandro Moschitti

Roma. October, 30th 2005

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The success of modern Information Technologies and Web-based services critically relates to the access, selection and managing of large amounts of information usually expressed as textual data. Among the others the Information Retrieval (IR) research has studied and modeled methodology to organize and retrieve targeted data from large collection of unstructured documents.

Mainly, IR provides two facilities for the access to the target information: query based and category browsing. The former has originated search engines like *Google* or *Altavista* whereas an example of the latter, can be observed in the category hierarchy of *Yahoo*, e.g., *Arts*, *Business*, *Computers*, *Culture*, *Education*, *Entertainment*, *Health*, *News*, *Science* and so on. Users can more easily browse the set of documents of their own interests by firstly selecting the category of interest. Additionally, users can specify their information needs by means of a selection of basic categories so that sophisticated IR models can automatically author to them the pertinent documents, i.e. documents which belong to such categories.

The large amount of documents involved in the above processes requires the design of systems that automatically assign category labels to the documents. In the last years, a large variety of models and approaches have been proposed for such task. As a consequence a subarea of IR called automated Text Categorization (TC) has emerged. Usually, TC is based on Machine Learning and statistical techniques inherited from the IR studies. Such methods are applied to a set of documents, manually assigned to the target categories (*training set*), to automatically learn the target classification function.

Learning algorithms attempt to derive statistical properties from the doc-

uments of a target category that are not held by the documents of the other categories. Such properties are then used at classification time to decide the document class. To allow the learning algorithm to successfully carry out such task, we need to provide it with document representations from which interesting properties can be extracted. For example, if we represent documents with the number of vowel types which are present in them, most likely, the algorithm will not extract properties able to distinguish between *Sport* and *Medicine*. On the contrary, the set of document words seems to be sufficient to achieve accurate representations.

Such representation is very common in IR and it is often referred to as *bag-of-words* as documents are encoded as simple multisets of words neglecting important linguistic aspect of natural language like morphological, syntactic and semantic structures. Nevertheless, this approach has shown high accuracy in automated classification, usually provided by the percentage of correct assignments within a set of documents not used for training, i.e. the *test set*.

Correctly measuring the accuracy of a classification system has become a crucial issue as some specific sectors, ranging from changes in management positions to business intelligence or information about terrorist acts, strongly relate on precise selection of the targeted data. Such necessity has produced more and more accurate TC learning models by studying two strategies: (a) improving categorization algorithms by using several theoretical learning models (e.g., [Joachims, 1998; Yang, 1999; Tzeras and Artman, 1993; Cohen and Singer, 1999; Salton and Buckley, 1988; Ng *et al.*, 1997; Moulinier *et al.*, 1996; Apté *et al.*, 1994; Quinlan, 1986; Hull, 1994; Schütze *et al.*, 1995; Wiener *et al.*, 1995; Dagan *et al.*, 1997; Lewis *et al.*, 1996; Ittner *et al.*, 1995]) and (b) designing document representations more sophisticated than *bag-of-words*.

Unfortunately, complex learning algorithms often show a high time complexity for both training and classification. This makes difficult the adoption of such algorithms for operational scenarios, where the number of instances is very large. For instance, web applications require effective data organization and efficient retrieval as for huge and growing amount of documents. To govern such complexity, the current trend is the design of efficient TC approaches [Lewis and Sebastiani, 2001]. A careful analysis of the literature reveals that linear classifiers are the most (computationally) efficient models [Sebastiani, 2002]. These are based on a vector representation of both documents and categories by means of feature weights derived via different approaches [Hull, 1994; Schütze *et al.*, 1995; Wiener *et al.*, 1995; Dagan *et al.*, 1997;

Lewis *et al.*, 1996; Cohen and Singer, 1999]. The decision if a document belongs or not to a category is then made measuring the similarity between the target vector pair (i.e., document and category).

Among others there are two linear classifiers which are the most representative of text categorization literature: Rocchio's text classifier [Rocchio, 1971] and Support Vector Machines (SVMs) [Joachims, 1999]. The former has origins in the IR literature, is one of the most computationally efficient classifier in both training and classification phases, is based on a heuristic derived from the experience in document retrieval and has been using since the early age of IR to take into account of the user feedback in search engine queries. The latter embodies the latest results in statistical learning theory [Vapnik, 1995], is considered one of the most accurate classifier and shows several important properties such (1) the ability to work in very high dimensional spaces and (2) the possibility via kernel functions to learn non-linear models.

With the same aim of improving accuracy of text classifier several researches on the use of a richer document representation have been carried out. Linguistic structures [Voorhees, 1993; Strzalkowski and Jones, 1996] could embed more information than the simple words which helps TC systems to learn the differences among different categories. Typical structures experimented in IR are complex nominals, *subject-verb-object* relations and word senses. This latter, is particularly useful in representing the document content unambiguously. For example the *slide* as transparency for projectors and the *slide* as sloping chute for children are the same words whereas the meaning is completely different. Such richer representations, are usually extracted by applying some automatic Natural Language Processing (NLP) techniques, but, at the moment they have failed to improve TC.

Although we are not able to design richer linguistic structures to improve TC systems. The current machine learning models based on bag-of-words are enough accurate and efficient to be applied to real scenario applications. Among other, Information Extraction (IE), *Question/Answering* (Q/A) and Text Summarization (TS) are useful applications that can be improved by the use of TC. TC can help in locating specific documents within a huge search space (*localization*) while IE or Q/A support the focusing on specific information within a document (*extraction* or *explanation*). Text classifiers provide for each document a set of categories that indicate what the main subjects of the documents are. For instance, text classifiers can assign categories to small texts also, e.g., paragraphs or passages. This knowledge can be exploited by IE, Q/A

and TS systems to respectively extract the events of a certain type, choose the answers related to the target subject or select the important passages related to a specific domain.

The many potential application fields for which TC may be successfully applied have aroused the interest in defining a methodology for TC system design. Indeed, there are a set of commonly recognized steps that should be applied to obtain an accurate TC system. The next section introduces such techniques.

1.1 Designing a Text Classifier

The design of generic text classifiers includes a set of steps universally recognized by the research community. We will briefly summarize them in the following by postponing to Chapter 3 their in depth discussion:

- *Features design*, where the following pre-processing steps are carried out:
  - *Corpus processing*: filtering and formatting of all the corpus documents.
  - *Extraction of relevant information*: a *stop list* is applied to eliminate function words (that exhibit similar frequencies over all classes) and the interesting linguistic information is extracted. The usual approaches use words as basic units of information but more complex features may be built, e.g. structured patterns like syntagmatic expressions or word senses.
  - *Normalization*: word stemming, carried out by removing common suffixes from words (words after stemming are usually called *stems*.). This is a classical method to approximate a conceptual representation, e.g. the *acquire* concept can also be expressed as *acquisition* and *acquires*. When more complex features are derived via linguistic analysis (i.e. words and/or complex nominals), normalization usually refers to the lemmatization process (i.e. detection of the base form of rich morphological categories such as nouns or verbs[1])

[1]Notice that this is very important for languages with a rich generative morphology where even hundreds of different forms can be derived from the same root.

  - *Feature selection*, which is an attempt to remove non-informative features from documents to improve categorization accuracy and reduce computational complexity. Typical selection criteria are based on statistical quantities like Chi Square, Mutual Information or document frequency.

- *Feature Weighting*: features usually assume different roles in different documents, i.e. they can be more or less representative. Different weights are associated with features via different and possibly diverging models.

- *Similarity estimation* which is modeled via operations in feature spaces. This can be carried out between pairs of documents or between two more complex feature groups (e.g. profiles which are the combination of features coming from different representative documents). Such estimation is typically adopted with quantitative models.

- *Application of a machine learning model*: once the similarity estimation has been defined in most of the cases the target machine learning algorithm can be applied. In this phase, a set of training documents whose correct classifications are known is needed. The output of the learning phase is a model of one or more categories.

- *Inference*: the similarity (or the membership function) between documents and category models is used to make classification decisions. The assignment of an incoming document to a target class is based on a decision function over the similarity scores. Different criteria (i.e. purely heuristics or probability-driven rules) are used in this task.

- *Testing*: the accuracy of the classifier is evaluated by using a set of prelabeled documents (i.e. *test set*) which are not used in the learning phase (*training set*). The labels produced by the target classifier are compared to those from the gold standard. The result of this phase is usually one or more numerical values which provide a measure of the distance between the human classifications (i.e. the Gold Standard classifications) and the target automatic classifier.

It should be pointed out that there are two types of classifiers: those that can make decisions over a set of categories (multiclassifiers) by outputting a set of category labels and those that can only decide if a document belongs or not

to a target category (binary classifiers). By training a binary classifier for each category, we can design a multiclassifier. A binary classifier can be trained by separating the corpus documents in two different sets: (1) the positive documents that are categorized in the target class and (2) the negative documents that are not categorized in it. To classify a new document, it is enough to apply the pool of binary classifiers and collect their decisions.

Such approach is usually applied to profile-based classifiers since they inherently express binary decisions. They describe each target class (C_i) in terms of a profile, i.e. a vector of weighted features. Such vector is extracted from documents previously categorized under C_i (training set). The classification phase is accomplished by evaluating the similarity between an incoming document d and the different profiles (one for each class).

More in detail, to design *Profile-based* classifiers, we apply phases which are slightly different from the previous design steps:

- *Features Weighting* for documents and profiles:
  - A representation vector $\vec{d}$ of a document d is derived by extracting its features f and assigning the weights $\vec{d}_f$ to them.
  - A representation vector $\vec{C}_i$ of a class C_i is evaluated by considering $\vec{d}$ belonging to C_i (i.e. $d \in C_i$) as positive examples and the vector $\vec{d} \notin C_i$ as negative examples.
- Application of a machine learning model: the output of the learning of profile-based classifiers is the vector $\vec{C}_i$. In this sense the machine learning algorithm can be seen as a weighing model for the category profile vector.
- *Similarity estimation* which is always carried out between unknown (i.e. not classified) documents and the profiles. The similarity is usually derived within the space determined by the features, i.e. between $\vec{d}$ and $\vec{C}_i$.
- *Inference*: A decision function is applied over the similarity scores. The most widely used inference methods are: probability, fixed and proportional thresholds. These are respectively called in [Yang, 1999]: $Scut$ (a threshold for each class is applied and is used to decide whether or not a document belongs to it), $Rcut$ (the best k-ranked classes are assigned to each document) and $Pcut$ (the *test set* documents are assigned to the classes proportionally to their size).

It should be noted that the choices made at each step determine a different classification model which may be more appropriate for an application domain rather than another. For example, it is well known that there is no optimal weighting scheme as its performance strongly depends on the application corpora.

The above rationale does not exactly apply for the choice of the machine learning algorithms as often there are a few models that are more accurate than all the others. In the next section, we briefly report the most famous machine learning approaches used for TC to give a flavor of this prolific research field.

1.2 Machine Learning Approaches to Text Categorization

In the literature several TC models based on different machine learning approaches have been developed. Whatever technology is adopted, the TC system suffers from the trade off between accuracy in retrieval and time complexity of the training/testing phases. Such complexity is critical in operational scenarios since it may prevent the processing of all required data. In the following, we briefly revisit the best known approaches as well as more recent ones. Particular carefulness to operational aspects will be devoted.

Support Vector Machines (SVMs), recently proposed in [Joachims, 1999], are applied to the vector space model described in Chapter 3 to find a category profile which produces the *lowest probability error*[2] on document classification. Such properties allow the SVMs to achieve one of the highest accuracy (about 86%) on a well known TC benchmark called Reuters corpus.

The main SVM problems are their application to operational scenarios where the number of training documents is thousands times higher than the number of documents contained in benchmarks. The disadvantage of SVMs is the training time which is quadratic in the numbers of training examples. The classification phase also can be very slow for nonlinear SVMs [Vapnik, 1995] since it is proportional to the number of support vectors which, in turn, increase proportionally with the number of training documents. This means that, to classify each single document, thousands of support vectors could be involved. As each support vector requires a scalar product with the input documents the time is usually very high.

[2]The exact meaning of *lowest probability error* will be explained in Section 2.

KNN is an example-based classifier [Yang, 1994] which makes use of document to document similarity estimation. It selects a class for a document through a k-Near Neighbor heuristic. For this the algorithm requires the calculation of all the scalar products between an incoming document and those available in the *training set*. The optimization proposed by the EXP-NET algorithm [Yang, 1994] reduces the computational complexity to $O(N \cdot log(N)$ time, where N is the maximum among the number of training documents, the number of categories and the number of features. The KNN time complexity is thus rather high.

$Rocchio$ [Ittner *et al.*, 1995; Cohen and Singer, 1999] often refers to TC systems based on the Rocchio's formula for profile estimation (described in Section 3.2.2). An extension of the algorithm was proposed in [Schapire *et al.*, 1998] and [Lam and Ho, 1998] but both approaches relevantly increase the complexity of the basic model.

PRC [Moschitti, 2003c] is the parameterized version of the Rocchio classifier. It will be presented in Section 3.5 to give an example of the positive impact of a correct parameterization.

RIPPER [Cohen and Singer, 1999] uses an extended profile notion based on co-occurrences and multiwords. A machine learning algorithm allows the contexts (e.g. a windows of n words) of a word w to decide how (or whether) the presence/absence of w contribute actually to the target document classification. As it is based on profiles, it can be very fast in on line classification task, but it has a noticeable learning time. Moreover, given the complexity to derive the suitable multiwords, it is not clear if it can be applied to millions of documents.

CLASSI is a system that uses a neural network-based approach to text categorization [Ng *et al.*, 1997]. The basic units of the network are the perceptrons. Given the amount of data involved in typical operational scenarios the size of the target network makes the training and classification complexity prohibitive.

Dtree [Quinlan, 1986] is a system based on a well-known machine learning method (i.e. decision trees) applied to training data for the automatic derivation of a *classification tree*. The Dtree model selects the relevant words (i.e. features) via an information gain criterion and predicts the target document's categories according to word combinations (see Section 2.1.1 for more details). It efficiently supports on line classification as the category assignment time is proportional to the time required to visit the decision tree.

CHARADE [Moulinier *et al.*, 1996] and SWAP1 [Apté *et al.*, 1994] use machine learning algorithms to inductively extract Disjunctive Normal Form rules from the training documents. Sleeping Experts (EXPERTS) [Cohen and Singer, 1999] are learning algorithms that work on-line. They reduce the computation complexity of the training phase for large applications by updating incrementally the weights of *n*-gram phrases. The reduced complexity makes them appealing for a real application but their accuracy is far away from the *state-of-the-art*.

Naive Bayes [Tzeras and Artman, 1993] is a probabilistic classifier which uses joint probabilities of words and categories to estimate the conditional probabilities of categories given a document. The naive approach refers to the assumption of word independence. Such assumption makes the computation of the Naive Bayes classifier much more efficient than the exponential complexity of a pure Bayesian approach (i.e. where predictors are made of word combinations). In this case the only problem is its low classification accuracy on every corpus.

In order to establish which classification model is more accurate several referring TC benchmarks have been developed. The most famous one is the Reuters corpus. In particular, previous work has shown (e.g. [Yang, 1999]) that five Reuters versions exist and TC systems perform differently on them. In particular, Table 1.1 reports the performance of the above TC systems on Reuters 22173 or Reuters 21578. Both of these versions provide two splits between training and testing: Apté and Lewis modalities [Sebastiani, 2002]. It is worth noting that the same classifier can achieve different accuracy on different Reuters versions/splits. Thus, Table 1.1 provides only an indicative accuracy[3] comparisons of TC models[4].

Table 1.1 shows that the best figure on the Reuters corpus is obtained by the example-driven KNN classifier (82.3/85%) and by SVMs (86%). Unfortunately, they have a heavier training and classification complexity, which makes their use more difficult within real operational domains thus PRC seems more suitable. Other classifiers having a fast on line classification (e.g. RIPPER, SWAP-1) are based on complex learning whereas the others show lower accuracy.

[3]More precisely, the accuracy measurements used are the microaverage BEP and the microaverage f-measure, which will be defined in Section 3.4.2.

[4]Moreover, the same model is subject to several implementations or enhancements. For example, Yang reports two Naive Bayes BEPs: 71% [Yang, 1999] vs. 79.56% [Yang and Liu, 1999].

Table 1.1: Breakeven points of widely known classifiers on Reuters corpus

SVM	KNN	**PRC**	RIPPER	CLASSI	Naive Bayes
86%	85/82.3 %	**82.83**%	81/82%	80.2%	71/79.56%
SWAP1	CHARADE	EXPERT	$Rocchio$	Dtree	
79/80.5%	73.8/78.3%	75.2/82.7%	74.8/78.1%	79.4%	

1.3 Book Outline

This book aims to provide the reader with the technology suitable to design and implement moderns automated TC systems. In particular, such technology is based on the statistical learning theory which proposes the automatic design of classification function from examples. Other machine learning models have been developed but the proposed theory represents the current state-of-the-art on TC. Moreover, practical procedures often applied in the TC design are thoroughly described. These include the TC tuning phase and accuracy evaluation. Finally, some advanced topics in TC such as the use of complex document representations and interesting applications are illustrated.

More in detail, the book is organized as follows:

- Chapter 2 provides the basic notions of machine learning along with latest theoretical models developed in recent years. Traditional and simple algorithms based on probability theory such as the Naive Bayes and the decision tree classifiers are described. Then, the PAC learning theory is introduced as a general framework of the modern statistical learning theory. This along with the simple perceptron learning algorithm allows the reader to understand the basic ideas of Support Vector Machines, which, together with kernel methods, are the ultimate contribution of statistical learning theory.

- Chapter 3 describes the typical steps for designing a text classifier. In particular, several weighting schemes and the design of profile-based classifiers are shown in detail. Additionally, the learning and classification algorithms for Rocchio and Support Vector Machines have been comparatively analyzed. The important contributions of this chapter relates to the definition of the Parameterized Rocchio text Classifier (PRC)

and the performance evaluation over different corpora as they show to the reader practical procedures that should be followed in the design of text classifiers.

- Chapter 4 reports advanced TC topics by presenting some studies on the use of Natural Language Processing to extract advanced linguistic features for document representation. These may be divided in two main types: (a) those that use syntactic information, e.g., POS-tags and phrases and (b) those based on semantic information, i.e. word senses. Additionally, proposals on the advanced use of TC for interesting natural language applications, i.e., Information Extraction, Question Answering and Text Summarization are described.

The technical content of the different chapters require a lengthy use of basic notions of linear algebra and geometry with specific formalisms. The final appendixes offer a reference dictionary and a short mathematical compendium to help the reader.