Text Categorization and Support Vector Machines

István Pilászy

Department of Measurement and Information Systems
Budapest University of Technology and Economics
e-mail: pila@mit.bme.hu

Abstract: Text categorization is used to automatically assign previously unseen documents to a predefined set of categories. This paper gives a short introduction into text categorization (TC), and describes the most important tasks of a text categorization system. It also focuses on Support Vector Machines (SVMs), the most popular machine learning algorithm used for TC, and gives some justification why SVMs are suitable for this task. After the short introduction some interesting text categorization systems are described briefly, and some open problems are presented.

Keywords: Text Categorization, Machine Learning, Support Vector Machines, Information Retrieval.

1 Introduction

Nowadays through the sudden growth of the Internet and on-line available documents, the task of organising text data becomes one of the principal problems. A major approach is text categorization, the task which tries to automatically assign documents into their respective categories. TC is used to classify news stories, to filter out spam and to find interesting information on the web. Until the late '80s the most popular methods were based on knowledge engineering, i.e. manually defining a set of rules encoding expert knowledge. In these days the best TC systems use the machine learning approach: the classifier learns rules from examples, and evaluates them on a set of test documents.

1.1 Supervised Machine Learning

The task of supervised machine learning is to learn and generalize an input-output mapping. In case of text categorization the input is the set of documents, the output is their respective categories. Consider the example of spam filtering: the input is emails and the output is 0 or 1 (i.e. spam or not spam), and a simple spam filter may use the following rule: “an email is spam if and only if it contains the expression »this is not a spam«”.

1.2 Learning and Testing Phase

The input-output mapping used to classify examples is called a model. Inducing such a model from examples (input-output relationships) is called learning, and these examples are called learning examples. After learning one wants to evaluate or use the model on another set of examples – test examples – this is called testing.

2 Text Categorization as a Supervised Machine Learning Problem

Supervised machine learning methods prescribe the input and output format. Mostly the input space is a vector space, the output is a single real number, or boolean (positive/negative). Machine learning algorithms use so-called features, i.e. questions that depend on the learning examples, e.g. how many times does that document contain the word “money”. Each feature corresponds to one dimension of the vector space.

2.1 Transforming Text Documents into Vector Space

Text documents in their original form are not suitable to learn from. They must be transformed to match the learning algorithm's input format. Because most of the learning algorithms use the attribute-value representation, this means transforming it into a vector space.

First of all, documents need to be pre-processed. This usually means stopword filtering for omitting meaningless words (e.g. a, an, this, that), word stemming for reducing the number of distinct words, lowercase conversion etc.

Then the transformation takes place. Each word will correspond to one dimension, identical words to the same dimension. Let the word w_i correspond to the ith dimension of the vectorspace. The most commonly used method is the so-called TF•IDF term-weighting method [1]. Denote TFIDF(i,j) the ith coordinate of the jth transformed document.

$$\mathrm{TFIDF}(i,j) = \mathrm{TF}(i,j) \cdot \mathrm{IDF}(i)$$
$$\mathrm{IDF}(i) = \log \frac{N}{\mathrm{DF}(i)}$$

Where TF(*i*,*j*) means how many times does the *i*th word occur in the *j*th document, *N* is the number of documents, and DF(*i*) counts the documents containing the *i*th word at least once.

The transformed documents together form the term-document matrix. It is desirable that documents of different length have the same length in the vector space, which is achieved with the so-called document normalization:

$$TFIDF'(i,j) = \frac{TFIDF(i,j)}{\sqrt[\beta]{\sum_i TFIDF(i,j)^\beta}}$$

The dimensionality of the vector space may be very high, which is disadvantageous in machine learning (complexity problems, overlearning), thus dimension reduction techniques are called for. Two possibilities exist, either selecting a subset of the original features, or merging some features into new ones, that is, computing new features as a function of the old ones.

3 Supervised Machine Learning Algorithms for Text Categorization

After pre-processing and transformations, a machine learning algorithm is used for learning how to classify documents, i.e. creating a model for input-output mappings.

A linear model is a model that uses the linear combination of feature-values. Positive/negative discrimination is based on the sign of this linear combination.

There are more sophisticated models – decision trees, neural networks, etc. –, however it is interesting that in case of TC a linear model is expressive enough to achieve good results.

There is a lot of linear models: perceptron, logistic regression, naïve bayes, support vector machines.

Naïve bayes is very popular among spam filters, because it is very fast and simple for both training and testing: it has optimal training and testing time in the O() sense (proportional to read through the examples), simplicity to learn from new examples and the ability to modify an existing model.

Support Vector Machines (SVMs) have been proven as one of the most powerful learning algorithms for text categorization [3].

3.1 Support Vector Machines

SVMs are a generally applicable tool for machine learning. Suppose we are given with training examples $\mathbf{x}_i$, and the target values y_i□{-1,1}. SVM searches for a separating hyperplane, which separates positive and negative examples from each other with maximal margin, in other words, the distance of the decision surface and the closest example is maximal (see Fig. 1).

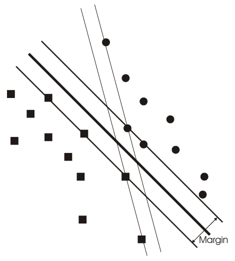

Figure 1
The optimal decision surface with maximal margin vs. a non-optimal decision surface

The equation of a hyperplane is:

$$\mathbf{w}^T \mathbf{x} + b = 0$$

The classification of an unseen test example **x** is based on the sign of $\mathbf{w}^T\mathbf{x} + b$. The separator property can be formalized as:

$$\mathbf{w}^T \mathbf{x}_i + b \geq 1 \text{ iff } y_i = +1$$
$$\mathbf{w}^T \mathbf{x}_i + b \leq 1 \text{ iff } y_i = -1$$

The optimization problem of SVM is the following:

Minimize over (**w**,b) the value of $1/2 \cdot \mathbf{w}^T \mathbf{w}$ subject to:

$$\forall_{i=1}^{n} : y_i[\mathbf{w}^T \mathbf{x_i} + b] \geq 1$$

The optimal **w** hyperplane has the minimum expected error of classification on an unseen and randomly selected example.

SVM can be generalized to handle non-separable cases in two way. First, with slack variables: minimize over (**w**,b) the value of $1/2 \cdot \mathbf{w}^T \mathbf{w} + C \cdot \sum_{i=1}^{n} \xi_i$ subject to:

$$\forall_{i=1}^{n} : y_i[\mathbf{w}^T \mathbf{x_i} + b] \geq 1 - \xi_i$$

Where C is a constant to trade off between margin and training error.

Second, with kernel functions. Let K($\mathbf{x_i}$,$\mathbf{x_j}$) denote a function that roughly speaking gives how similar two examples are. This is called a kernel-function if it satisfies the Mercers's condition [9]. The simplest case is when K($\mathbf{x_i}$,$\mathbf{x_j}$)=$\mathbf{x_i}^T \cdot \mathbf{x_j}$, but more complicated cases makes SVM suitable for non-linearly separable problems. The optimization problem is described in [9].

An interesting property of SVM is that the normal of the decision surface is a linear combination of examples. This means, that the decision function can be written in the following form: $f(x) = \mathrm{sign}(\sum_{i=1}^{N} \alpha_i \cdot y_i \cdot K(\mathbf{x}_i, \mathbf{x}) + b)$ [9].

Machine learning algorithms tend to overlearn when the dimensionality is high, for example when more dimension exists than example. SVM avoids this, because it does not combine features, but it linearly combines a function of the examples.

4 Text Categorization Tasks and their Characteristics

There are numerous tasks that can be automated with text categorization methods. The most important tasks are:

- Document Organization: for example organizing patent documents into categories for browsing
- Text Filtering: "»Text filtering is the activity of classifying a stream of incoming documents dispatched in an asynchronous way by an information producer to an information consumer«"[10]
- Document routing: "documents are to be assigned to one of two categories, relevant or non-relevant" [11].
- Spam filtering is a kind of document routing, with two categories: spam and not spam. Note that spam filtering is a cops and robbers game.
- Hierarchical categorization, web-directory building. "Webpage categorization has two essential peculiarities: the hypertextual nature of the documents, the hierarchical structure of the category set" [10]

Tasks that use similar techniques to text categorization:

- Word sense disambiguation: depending on the context of an ambiguous word, decide in which sense was it used.
- Information extraction (classifying extracted parts of texts): find company names, person names, places, etc. in text.

- Information retrieval: search engines may classify documents to be relevant or non-relevant for a given query, based on user feedback.

4.1 Common Characteristics

There are some characteristics of TC tasks that must be kept in mind when building a TC system. Some reasons, why machine learning is not suitable for text categorization:

- ML algorithms typically use a vector-space (attribute-value) representation of examples, mostly the attributes correspond to words. However word-pairs or the position of a word in the text may have considerable information, and practically infinitely many features can be constructed which can enhance classification accuracy. That is why fighting against spam is so hard.
- Categories are binary, but we would not assign documents so precisely. Often we would say about a document D that it belongs a bit to category X1 and a bit to category X2, but it does not fit well into any of the two, it would rather require a new category, as it is not similar to any of the documents seen before.
- There is an increasing number of words if we increase the number of documents. Heaps' law describes how the number of distinct words increase if we increase the size of a document-collection: $V=K \cdot N^{\beta}$, where V denotes the number of distinct words, N is the number of words in the collection, K and β are constant factors depending on the text and determined empirically [12].
- Representations use words as they are in texts. However, words may have different meanings, and different words may have the same meaning. The proper meaning of a word can (if can) be determined by its context, or in other words each word influences the meaning of its context. However, the usual (computationally practical) representation neglect the order of the words.

In exchange for the information loss caused by the representation, we can use a mathematically well-founded, efficient, practical algorithm, the SVM, and the representation is compact.

The transformation must aim at saving as much useful information as it can, if it is computationally feasible. Examples: instead of counting just the words, we can count word-pairs too, or counting exponentially-many features with dynamic programming, using an appropriate kernel function [13].

4.2 Benefits of SVM

SVMs can handle with exponentially or even infinitely many features, because it does not have to represent examples in that transformed space, the only thing that needs to be computed efficiently is the similarity of two examples. Redundant features (that can be predicted from another features), and high dimension are well-handled, i.e. SVM does not need an aggressive feature selection.

There are error-estimating formulas, which can help us in predicting how good a classification of an unseen example will be, elliminating the need for cross-validation techniques.

Joachims sums up why SVMs are good for TC [3]:

- High-dimensional input space.
- Few irrelevant features: almost all feature contain considerable information. He conjectures that a good classifier should combine many features and that aggressive feature selection may result in a loss of information.
- Document vectors are sparse: despite the high dimensionality of the representation, each of the document vectors contain only a few non-zero element
- Most text categorization problems are linearly separable.

5 Evalutating Text Classifiers

Text categorization systems may make mistakes. We want to compare different text classifiers to decide which one is better, that is why performance measures are for. Some of them measures the performance on one binary category, others aggregate per-category measures, to give an overall performance.

Denote TP, FP, TN, FN the number of true/false positives/negatives. The most important per-category measures for binary categories are [10]:

- Precision: p=TP/(TP+FP)
- Recall: r= TP/(TP+FN)
- F_β-measure: $F_\beta = \frac{(\beta^2+1)\cdot p \cdot r}{\beta^2 \cdot p + r} \quad 0 < \beta < \infty$
- precision-recall breakeven point (PRBP): we choose the TP mostly positive document, considering them as positive, and the rest as negative. We then calculate the precision and recall values, which will be equal in this case (ie. FP=FN).

The most important averages are: micro-average, which counts each document equally important, and macro-average, which counts each category equally important (see [10] for details).

5.1 Some Results from the Literature

The mostly used document collection is the Reuters-21578 corpora. The “ModApte” split leads to 9603 training documents, 3299 test documents, 90 categories with at least one training and one test example, and approx. 10000 distinct term. A simple TFIDF representation, eucledian normalization and linear SVM was enough to achieve 84.2% micro-averaged PRBP. With polynomial kernels – where $K(\mathbf{x}_i,\mathbf{x}_j)=(\mathbf{x}_i^T\cdot\mathbf{x}_j)^n$ – of degree n=4, 86.2% was achieved [3].

Another frequently used corpora is the 20-newsgroups text corpora, consistsing of 20000 messages taken from 20 newsgroups [5].

6 Kinds of Text Categorization Systems

6.1 Hierarchical Text Categorization Systems

A potential drawback of treating the category structure as flat is that the number of training examples for individual classes may be relatively small when dealing with a large number of categories [6]. Categories may form a hierarchical structure – e.g. patent documents, web catalogs – which can be exploited by advanced techniques, for example by using the divide and conquer approach: solving classification problems at each branch, yielding into fewer classes and more examples at higher levels, and fewer classes at lower levels.

6.2 Transductive Support Vector Machines

The presented supervised learning scheme is an inductive learning scheme. It induce a classifier from training examples. Support Vector Machines try to find a separating hyperplane that maximizes the margin, i.e. the distance between the decision boundary and the closest example.

The optimization task of the Transductive Support Vector Machines is: [8]

Minimize over $(y_1^*,\ldots,y_n^*,\mathbf{w},b)$ the value of $1/2\cdot\mathbf{w}^T\mathbf{w}$, subject to:

$$\forall_{i=1}^{n} : y_i[\mathbf{w}^T\mathbf{x_i}+b]\geq 1$$
$$\forall_{j=1}^{k} : y_j^*[\mathbf{w}^T\mathbf{x_j^*}+b]\geq 1$$

Where y_i is the labeling of training examples (-1 or 1), y_j^* is the labeling of test examples. Solving this problem means finding a labelling of test data and a model (**w**,b) that separates both training and test data with maximum margin [8].

6.3 Semi-supervised Learning

The usual way of building a text classifier consists of two steps: training to get a model and applying that model on unlabeled examples. However, more sophisticated approaches exist. In [7] they survey some well-known approaches. A simple approach first trains a model on labeled examples, then label some unlabeled examples by that model, retrain with the new, larger set, and iterates until no more unlabeled example exists. They conclude that that less labelled data is required to obtain the performance comparable with the pure supervised approaches.

7 Open Problems

A text categorization system may have a lots of parameters. Often it is not clear, how to set them, it heavily depends on the nature of the problem. For example word stemming is a good idea, if we have few examples or too much distinct words with the same stem. Tuning the parameters to obtain the best results requires a disjoint set of examples, also known as validation set. However, this process is time-consuming. Is there a way to speed up this process? Can we say that – for example – the optimal choice of stemming and the normalization factor are independent? Can we create independent clusters of parameters? Or one parameter depends on the other but not vice versa?

What is the maximum that can be mined from the training examples? How far are we from that maximum now? Does it make any sense to fine-tune the algorithms to the Reuters-21578 or any other corpora? Or the corpora are too different from each other, and any new problem needs re-desinging the half of a text categorization system?

Using background knowledge: how can learning algorithms benefit from lexicons, vocabularies, ontologies, etc?

Conclusions

This paper gives some introduction into text categorization, and describes the common tasks of a TC system. Using Support Vector Machines as a machine learning algorithm is nowadays the most popular approach. Some aspects of SVMs are covered that reflect why they are suitable for TC. In the rest of the paper some interesting kind of TC systems are described briefly, and after all some open problems are posed.

References

[1] A. Aizawa: An information-theoretic perspective of tf-idf measures, Information Processing and Management: an International Journal archive, Vol. 39, Issue 1, 2003, pp. 45-65

[2] Charles Elkan: Boosting and Naive Bayesian Learning, Technical Report No. CS97-557, September 1997, University of California, San Diego

[3] Thorsten Joachims: Text categorization with support vector machines: learning with many relevant features, Proc. of ECML-98, 10th European Conference on Machine Learning, Springer Verlag, Heidelberg, DE, 1998, pp. 137-142

[4] David D. Lewis: Reuters-21578 text categorization test collection, 1997 http://www.daviddlewis.com/resources/testcollections/reuters21578/

[5] The 20 Newsgroups data set http://people.csail.mit.edu/jrennie/20Newsgroups/

[6] Lijuan Cai, Thomas Hofmann: Hierarchical Document Categorization with Support Vector Machines, ACM 13th Conference on Information and Knowledge Management, ACM Press, New York, NY, USA, 2004, pp. 78-87

[7] Bogdan Gabrys, Lina Petrakieva: Combining labelled and unlabelled data in the design of pattern classification systems, International Journal of Approximate Reasoning, 35, 2004, pp. 251-273

[8] Thorsten Joachims: Transductive Inference for Text Classification using Support Vector Machines. Proceedings of the 16th International Conference on Machine Learning (ICML), San Francisco, CA, USA, 1999, pp. 200-209

[9] Christopher J. C. Burges: A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2(2), 1998, pp. 955-974

[10] Fabrizio Sebastiani: Machine learning in automated text categorization, ACM Computing Surveys (CSUR), Vol. 34 Issue 1, 2002, ACM Press, New York, NY, USA, pp. 1-47

[11] Hinrich Schutze, David A. Hull, and Jan O. Pedersen: A comparison of classifiers and document representations for the routing problem. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1995 July, pp. 229-237

[12] Heaps' law: http://en.wikipedia.org/wiki/Heaps'_law

[13] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, Chris Watkins: Text Classification using String Kernels, The Journal of Machine Learning Research, Volume 2, March 2002, pp. 419-444