Robust inference method for profile-based Text Classification[*]

Roberto Basili, Alessandro Moschitti, Maria Teresa Pazienza

University of Rome Tor Vergata
Department of Computer Science, Systems and Production
00133 Roma (Italy)
{basili,moschitti,pazienza}@info.uniroma2.it

Abstract

Text Classification is a relevant research area in modern Information Retrieval. In this paper a statistical inference technique is presented and its impact on profile-based classification models is measured. Extensive testing of models differing with respect to weighting policies and inference algorithms has been carried out. The relatively simple method proposed here is compared against other more complex models (i.e. Logistic Regression). Better performances have been obtained and their persepective in large scale and dynamic classification scenarios are discussed.

1. Introduction

Thematic text classification, able to retrieve and organise textual data within an existing and dynamic (sensible to user need) framework, is assuming an increasingly relevant role. The classification problem can be classically described as:

- Given a set of (possibly evolving) user needs expressed as a structure of topics/subtopics classes
- Given a variety of existing examples of these classes (also referred as training data set (Tr)),
- Build a decision function able to upgrade the existing example repository with the suitable new texts, (possibly automatically retrieved).

Classes ($C = \{C_1, ..., C_z\}$) are used to represents topics/areas of interest, and a potential hierarchical structure among them is the traditional representation structure for user needs. The decision function is thus asked to map newly incoming documents (d) in one (or more) class(es), according to their content.

Two main approaches to the construction of a non-parametric classifier have been proposed and experimented in literature (Lewis et al., 1996).

Profile-based (or linear) classifiers derive a description of each target class (C_i) in terms of a profile, usually a vector of weighted terms. These vectors are extracted from the training documents pre-categorized under C_i. This approach can be referred as *category-centred* classification. Classification is thus the evaluation of similarity between the incoming document d and the different profiles (one for each class). Early profile-based classifier made use of the vector space model (Salton and Buckley, 1988) to define similarity, by switching from an information retrieval to a text classification task. Some of the experiments in this paper used this

[*]The research described in this paper has been partially funded under the TREVI ESPRIT project, n. EP23311

method (mapping the SMART Information Retrieval system (Salton, 1991) into a classification system) as a baseline.

Example-based are other types of classifiers, in which the incoming document d is used as a query against the training data (Tr). Similarity between d and the documents in Tr is evaluated. The categories under which the training documents with the highest similarity are categorized, are considered as promising classification candidates for d. This approach is also referred as *document-centred* categorization. An example in this class is the k-NN algorithm used in the ExpNet system (Yang, 1994).

The design of a profile-based classifier included the following steps:

- *Features design*: the linguistic information that characterize a document (and a class) are here selected. The simplest ones are single terms. More complex features can be built as structured patterns (i.e. multiple word expressions), or by adding lexical information (i.e. word senses).
- *Weighting schemes definition*: Features assume usually different roles in documents, i.e. they are more or less representative. Different weights are associated to features thus producing different representations.
- *Learning the synthetic profile*:
  - A representation $\vec{d}$ of a document d is defined by means of the set of features t extracted from d. Weights of those features are usually included in such quantitative representation
  - A similar representation $\vec{C_i}$ of a class C_i summarizes the representation $\vec{d}$ of all documents that are positive instances for C_i (i.e. for those d such that $d \in C_i$)
- *Similarity estimation*, (s_{d,C_i}), is modeled via operations in spaces of features. This can be carried out between documents and the above defined profiles. Usually quantitative models (i.e. metrics) are adopted for this purpose.
- *Inference*: the similarity evaluation over document representations trigger a classification decision. Assignment of an incoming document to the suitable classes is based on a separate (and often independent) decision function over similarity scores (i.e. inference). Different criteria (i.e. purely heuristics or probability-driven rules) are here used.

The *inference mehtod* represents a crucial property of a profile-based classifer enabling the selection of the suitable classes for the incoming document. In general, the methods proposed in literature assign a document d to a category C_i only according to the score in such class (i.e. a function over the similarity score s_{d,C_i}). In fact, the ranking of candidate classes based on scores is not effective, due to the empirical nature of these values. They do not show analytically consistent behaviour so that they are not directly comparable. Logistic Regression (Agresti, 1989) is an attempt to avoid this trouble. LR is used to transform simple scores s_{d,C_i} into the conditional probabilities of a class given a document. In this way by comparing probabilities of classes for a given document the inference applies as the selection of the highest (i.e. most probable) ones.

In this paper a new method for inference in text classification (called RDS) is proposed. Three profile-based classifiers are comparatively evaluated to measure the effectiveness and robustness of the method. An original property of the experimented classifiers is the use of syntactic features in profiles (i.e. Part-of-Speech categories of lemmas) (Basili et al., 1999). A novel weighting system is also used. Section 2 presents the specific weighting policies adopted during the tests. Section 3.2 presents the required formal definitions and the original aspects of the inference method. Section 4.2 discussing the results of large scale experiments over a traditional test case (i.e. Reuters).

2. Weighting schemes

The aim of this paper was to experimentally observe the impact of RDS on different profile-based classifiers. Differences in the tested models are related to weighting policies and inference methods. Three weighting schemes have been experimented. The first is the usual TF/IDF scheme widely used in the vector space model (as in (Salton, 1991)). A novel scheme is IWF, that is a slightly different version of the TF/IDF model (see 2.1). The third is the common weighting scheme associated with the Rocchio's classifier (Cohen and Singer, 1996) (see 2.2).

Any of these weighting schemes is based on the following definitions. Given a training set *Tr*, a set of features $\{t_1, ..., t_n\}$ describing *Tr*, a generic document d_h of the corpus and the target set of classes $\{C_1, C_2, ..., C_z\}$, the document representations are defined as vectors of weighted features, i.e.:

$$\vec{d_h} =<< t_1, \bar{\omega}^h_{t_1} >, ..., < t_n, \bar{\omega}^h_{t_n} >>$$

while, similarly, class profiles are:

$$\vec{C_i} =<< t_1, \omega^i_{t_1} >, ..., < t_n, \omega^i_{t_n} >>$$

Different weighting policies differ basically in the way $\bar{\omega}^h_{t_k}$ and $\omega^i_{t_k}$ are respectively defined to represent the relevance of feature t_k in documents d_h and classes C_i.

2.1. IWF-based weighting

Let:

- F_t be the occurrences of a feature $t \in \{t_1, ..., t_n\}$ in the training set,
- N be the overall occurrences of features (i.e. $\sum_t F_t$) and
- f^h_t the occurrences of t in the document d_h

In the IWF schemes document weights are

$$\bar{\omega}^h_t = (IWF)^2 f^h_t$$

and class weights are

$$\omega^i_t = (IWF)^2 \sum_{h \in C_i} f^h_t$$

The IWF (Inverse Word Frequency) is given by $IWF = log(\frac{N}{F_t})$.

On the above representations, $\vec{C_i}$, $\vec{d_h}$, the following similarity function can be applied:

$$s_{hi} = cos(\angle(\vec{C_i}, \vec{d_h})) = \frac{\sum_t \omega^i_t \bar{\omega}^h_t}{|\vec{C_i}||\vec{d_h}|} \quad (1)$$

The above model introduces two major differences with respect to the traditional weighting strategy TF/IDF used in $SMART$. First, the *Inverse Word Frequency* (Basili et al., 1999) is used in place of IDF. Its role is similar to IDF, as it penalizes high frequency (and less meaningful) terms (e.g. *be, have*) also recovering from systematic errors in POS tagging. Another significant difference with respect to SMART is the adoption of IWF squaring.

2.2. Rocchio weighting

The weighting scheme often called Rocchio's formula requires these quantities:

- M as the number of documents in the training set.
- n_t as the number of documents in which the term t appears, and
- $\grave{f}_t^h$ given by:

$$\grave{f}_t^h = \begin{cases} 0 & \text{if } f_t^h = 0 \\ log(f_t^h) + 1 & \text{otherwise} \end{cases}$$

Accordingly, given the usual $IDF(t)$ as $log(\frac{M}{n_t})$ the document weights will be

$$\bar{\omega}_t^h = \frac{\grave{f}_t^h \cdot IDF(t)}{\sqrt{\sum_{r=1}^n (\grave{f}_r^h \cdot IDF(r))^2}} \qquad (2)$$

and class weights will be

$$\omega_t^i = \max\left\{0, \frac{\beta}{|R_i|}\sum_{h\in R_i}\bar{\omega}_t^h - \frac{\gamma}{|\bar{R}_i|}\sum_{h\in \bar{R}_i}\bar{\omega}_t^h\right\}$$

Notice that R_i is the set of training documents examples of class C_i, while $\bar{R}_i$ represent documents not belonging to C_i. The parameters β and γ control the relative influence of positive and negative examples on the classifier[1].

It is worth noticing that the Rocchio's weights reflect normalization (as $\sqrt{\sum_{r=1}^n (\grave{f}_r^h \cdot IDF(r))^2}$ is used in 2). Therefore a different similarity function (that does not apply any normalization in $\vec{C}_i$ and $\vec{d}_h$, as opposed to Equation 1) is used:

$$s_{hi} = cos(\angle(\vec{C}_i, \vec{d}_h)) = \sum_t \omega_t^i \bar{\omega}_t^h$$

3. Inference methods in Text Classification

Inference is usually approached by thresholding over similarity scores. Three main approaches to thresholding have been proposed in literature (Yang, 1999) also depending on the constraints imposed by the specific application.

- probability thresholding ($Scut$): for each class C_i a threshold γ_i is adopted such that a document d will be categorized under C_i only if its membership score is greater than γ_i. The threshold γ_i is seen as an upper limit to the risk of misclassification and has a probabilistic origin: it measures the average number of potential misclassifications under a given assumption on the distribution

[1]In the experiments the standard values β=16 and γ=4 have been used, as in (Cohen and Singer, 1996))

- fixed thresholding ($Rcut$): given the assumption that k is the average number of classes valid for a generic document d_h, categories C_i are sorted by their membership scores, and the first k of them are selected.
- proportional thresholding ($Pcut$): the threshold is the percentage of documents that are to be categorized under C_i. It usually estimated from the training set.

3.1. Inference with Logistic Regression

The inference step would be systematic in case an actual estimate of $P(C_i|d)$, i.e. the probability that a document d belong to the class C_i, was available. However, similarity scores are not such probabilities, although they can be used as good predictors (Lewis and Gale, 1994).

In synthesis the Logistic Regression-based inference algorithm works as follows. First couples $<s_{hi}, belong\_flag>$ for each training document d_h and for each fixed class i are built: $belong\_flag$ is 1 in case $d_h \in C_i$ and 0 otherwise. The derived couples are input for the Logistic Regression algorithms. It produces two parameters α_i and β_i such that the logistic function (Ittner et al., 1995)

$$\frac{e^{\alpha_i+\beta_i \cdot s_{hi}}}{1+e^{\alpha_i+\beta_i \cdot s_{hi}}}$$

is a good approximation of $P(C_i|d_h) = P(C_i|s_{hi})$. It means that the output suggests the conditional probability $P(C_i|d_h)$ that a document d_h belongs to C_i, given that its similarity score s_{hi}.

Each class i will be assigned with its coefficients α_i and β_i. Decision is taken over the images of similiarity scores obtained via the logistic function. The $Rcut$ inference strategy is here applied.

It is worth noticing that the logistic function is monotonic ascending so that, given a class C_i, the ranking of documents according to $P(C_i|d_h)$ or on s_{hi} is the same. LR can thus influence the classification performance only via the $Rcut$ policy as $Pcut$ and $Scut$ just impose a threshold on such ranking.

3.2. Inference via Relative Difference scores

In order to select the suitable classes for a document, thresholding over s_{hi} is widely adopted as an empirical criteria (see (Lewis, 1992) for a comparative evaluation). We defined a thresholding policy based on a probabilistic approach (empirical estimation from training data) of the *differences between similarity scores*. Instead of the s_{hi} scores directly, a stochastic variable m_i, expressing the average difference between the score of the correct (i-th) class and the remaining classes, i.e.

$$m_i = \frac{\sum_{j=1}^{z} s_{hi} - s_{hj}}{z-1} \quad (3)$$

is used. For each class, the mean and standard deviation, (respectively as $E(m_i)$ and $StdDev(m_i)$ of m_i) are estimated over all documents d_h in the training set.
Given the vector $f(d_h) =< s_{h1}, ..., s_{hn} >$, we assign d_h to C_i if its corresponding m_{hi} has the following property:

$$m_{hi} > E(m_i) - \alpha_i StdDev(m_i) \quad (4)$$

where each α_i is a threshold (empirically determined to optimise recall and precision over the test data). Equation 4 is the inference method hereafter called RDS.

3.2.1. Main characteristics of the method

The RDS method we propose produces an improvement of the breakeven point with respect to the policies discussed in (Lewis, 1992). It is in fact to be seen as an extension of *proportional thresholding* policy as it is estimated over the training data. RDS is independent from the document stream (i.e. the overall set of incoming data) as it applies individually to documents. RDS is expected to improve (and in fact it does) the system recall, keeping the same precision if compared with other policies, in fact is better suited to deal with those *odd* documents d_h that are not similar to the other texts in the training set , i.e. texts that have low s_{hi} values for each i. RDS is not influenced by the average membership scores of documents in the training set (it is thus less biased by the training data). It does not fix the number of classes (k) to be retained for a document. RDS has been shown more robust with respect to categories with different specificity.

4. Experimental Results

For comparative purposes, three profile based classifiers, using as features the document's lemmas associated with their restricted set of part-of-speech (POS) tag labels (i.e. nouns, verbs and adjectives), have been implemented: NL/RDS, *Rocchio* and $SMART$. The NL/RDS classification model, that we proposed in (Basili et al., 1999), adopts the IWF weighting scheme with relative similarity function (described in Section 2.1) and the RDS inference method. $Rocchio$ classifier (Ittner et al., 1995) uses the Rocchio's weighting system and the similarity function defined in Section 2.2).$Rocchio$ has been run with $Scut$ and $Rcut$ inference methods applying sometime Logistic Regression and RDS. $SMART$ classifier refers to the classical vector space model weighting scheme (TF/IDF) applied to profiles as IWF. $SMART$ has been run with RDS and $Scut$.

4.1. Description of the data set

The NL/RDS, $SMART$ and $Rocchio$ model have been used within the TREVI (Text Retrieval and Enrichment for Vital Information) system. TREVI, is a system for Intelligent Text Retrieval and Enrichment. TREVI (Basili et al., 1998) is a distributed system for text classification and enrichment, designed and developed by a European consortium under the TREVI ESPRIT project EP23311. Reuters is a member of the Consortium and has been used as a main "*User Case*" for the released prototype.
A specific subset of the classes (judged particularly meaningful to the Reuters customer service) is currently managed by the prototype. It includes 30 classes in different levels of the Reuters classification system. For these categorisation task, we received $29,026$ manually classified documents. Cross validation has been run using 90% of the overall data as training and testing on the remaining portion. The Breakeven point have been used as performance index.

4.2. Evaluation of the RDS

In Table 1 the breakeven point of the NL/RDS classifier with respect to the $SMART$ model is reported. Both statistical models have been run over the output of the linguistic processor (i.e. POS tagged lemma detected in documents). The $SMART$ model has been run using two different classification rules: $SMART$ ($Scut$) adopts probability thresholding, while $SMART + RDS$ carries out inference via the relative difference score (Eq. 4). The only difference between NL/RDS and $SMART$+RDS is the IWF squaring in place of IDF.

Table 1: Classification Accuracy

	SMART ($Scut$)	SMART+RDS	NL/RDS
Breakeven Point	63%	72%	76%

In Table 2 we can observe the breakeven point relative to different versions of the $Rocchio$ model. First and second column report the breakeven point achieved by the models that do not use LR. They differ for the adopted thresholding policy ($Rcut$ and $Scut$). The third and fourth column report the performance of the $Rocchio$ model using the LR and the RDS technique.

Table 2: Classification Accuracy

	Rocchio ($Rcut$)	Rocchio ($Scut$)	Rocchio+LR ($Rcut$)	Rocchio+RDS
Breakeven Point	47.04%	62.78%	66.55%	71.60%

5. Discussion

In Table 2 we observe that the $Rcut$ policy has a poor performance. This is due to the complexity of the task in our Reuters test set. In fact classes are very rich and they need more than one profile to be suitably represented: the use of a single profile leads to high variability among scores depending on documents and classes. This makes a direct comparison among scores (adopted in $Rcut$) ineffective. The performances grow when $Scut$ is used as the comparison is carried out only within a class, where variability is less important. The LR technique, projecting all scores on the same interval ([0,1]), allows a direct comparison thus improving the system performance of about 19 %.
RDS produces a further increment with respect to the LR as it has characteristics similar to $Scut$ (a threshold for each class) and it also allows direct comparison. In fact, "odd" documents have low scores in all classes since have few words in common with the profiles of all classes and are usually rejected with an $Scut$ policy. If RDS is used instead they can be accepted, although scores are low. The RDS technique, by using the relative difference among scores, links the decision for a class to all the others, thus capturing more "information" than the $Scut$ policy. Similarly LR projects all scores in the [0,1] range and is sensitive, via an $Rcut$ policy, to the contribution of all classes. According to this first experimental setting, we might state that even the $Rcut$ policy alone links a decision for a class to all the others, but the adopted similarity and weighting models are not able to provide comparable scores in our test set.

A direct comparison between Logistic Regression and RDS (see Table 2) show that both are robust with respect to "high-variability" phenomena in score assignment. In both cases the transformed simlarity scores depend on all classes. According to our extensive testing, this property systematically improves the BEP of a profile-based text classification system. It is worth noticing that the RDS technique is simple and efficient to implement. The LR requires a more costly implementation and current experiments suggest that it produces improvements lower than RDS.

6. Conclusion

In (Basili et al., 1999) we defined an inference technique (RDS) for text classification that improves performances of different classifiers (NL/RDS, $Rocchio$ and $SMART$) (about 9% in BEP). In order to study the effects of the RDS inference model, we compared the best performing profile-based classifier (the $Rocchio$ model with LR) with NL/RDS. The $Rocchio$ system has been also coupled with the RDS inference rule. Again a relative increment of the BEP has been found, this representing a further evidence of the positive influence of RDS (9% in different cases). Moreover, the RDS technique is simple, efficient to implement while LR requires a more costly implementation. Further measurements are required to better assess the effectiveness of the proposed NL/RDS techniques over other standard test sets, where wide experimental evidences are already available for comparative purposes.

References

Agresti A. editor (1989). *Categorical Data Analysis*. Jhon Wiley, New York.

Basili R., Di Nanni M., Mazzucchelli L., Marabello M., and Pazienza M. (August 1998). Nlp for text classification: the trevi experience. In *Proceedings of the Second International Conference on Natural Language Processing and Industrial Applications, Universite' de Moncton, New Brunswick (Canada)*.

Basili R., Moschitti A., and Pazienza M. (1999). A text classifier based on linguistic processing. In *Proceedings of IJCAI 99, Machine Learning for Information Filtering, http://www-ai.cs.uni-dortmund.de/EVENTS/IJCAI99-MLIF/papers.html*.

Cohen W. W. and Singer Y. (1996). Context-sensitive learning methods for text categorization. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 96')*, pages 12–20.

Ittner D. J., Lewis D. D., and Ahn D. D. (1995). Text categorization of low quality images. In *Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 301–315, Las Vegas, US.

Lewis D. D. (1992). An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of SIGIR-92, 15th ACM International Conference on Research and Development in Information Retrieval*, pages 37–50, Kobenhavn, DK.

Lewis D. D. and Gale W. (1994). A sequential algorithm for training text classifiers. In *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, pages 3–12, Dublin, Ireland.

Lewis D. D., Schapiro R. E., Callan J. P., and Papka R. (1996). Training algorithms for linear text classifiers. In *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 298–306, Zürich, CH.

Salton G. (1991). Development in automatic text retrieval. *Science*, 253:974–980.

Salton G. and Buckley C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523.

Yang Y. (1994). Expert network: effective and efficient learning from human decisions in text categorisation and retrieval. In *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, pages 13–22, Dublin, IE.

Yang Y. (May, 1999). An evaluation of statistical approaches to text categorization. *Information Retrieval Journal*.