Language Sensitive 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

It is a traditional belief that in order to scale-up to more effective retrieval and access methods modern Information Retrieval has to consider more the text content. The modalities and techniques to fit this objectives are still under discussion. More empirical evidence is required to determine the suitable linguistic levels for modeling each IR subtask (e.g. information zoning, parsing, feature selection for indexing,...) and the corresponding use of this information. In this paper an original classification model sensitive to document syntactic information and characterized by a novel inference method is described. Extensive experimental evidence has been derived on real test data and also from well-established academic test sets. The results show that a significant improvement can be derived using the proposed inference model. Also the role of linguistic preprocessing seems to provide positive effects on the performance. POS tagging and recognition of Proper Nouns received a specific experimental attention and provided significant effects on measured accuracy.

1. Introduction

It is a common believe that in order to scale-up to more effective retrieval and access methods modern Information Retrieval has to take more content into account. Current IR can be seen as a combination of traditional techniques (e.g. vector space models) with advanced access/browsing facilities and content-sensitive methods able to support extraction of linguistic information and multilinguality. Language processing is thus a crucial step to Information Access and Knowledge Management systems because of the stress on content rather than on the retrieval/access functionalities. It has been often stressed (e.g. (Grefenstette, 1991)), that vector space models are inadequate to deal with retrieval from Web via commonly available simple and short queries.

Although content can play a role, the nature and methods for taking it into account is still under debate. Crucial problems are at least related:

- to detect the suitable linguistic levels for each subtask (e.g. information zoning, specific parsing, indexing)
- to optimize the processing of each level (e.g. optimal parsing of Proper Nouns, terminological expressions and Part-Of-Speech assignment)
- to suitably combine the linguistic processes cooperating to a target IR tasks (which specific dependencies? how to combine the differently derived information?).

Among others, Text Classification is an important task in a IR scenario. It is playing a major role in retrieval/filtering but also in the development of user-driven on-line services. Users here expect all and only relevant information, possibly customized according to their operational and publishing needs. Moreover, the design of intelligent methods for automatic management of information/knowledge (where IR is seen as a way to Knowledge Management) is tightly based on classification capabilities. In fact, authoring of the information to feed a user-own knowledge repository is usually based on the content of classified texts, and, thus, a preliminary classification is a necessary step.

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

Thematic text classification (TC), able to retrieve and organize textual data within an existing and dynamic framework, is thus assuming an increasingly relevant role. The classification problem is traditionally described as follows:

i) Given a set of (possibly evolving) user needs expressed into (hierarchical) structures of classes (i.e. topics/subtopics labels).
ii) Given a variety of existing examples of these classes,
iii) Build a decision function able to automatically classify new texts into classes and upgrade the existing example repository.

Classes ($C = \{C_1, ..., C_n\}$), used to represents topics/areas of interest, are usually organized into a hierarchical structure representing the user needs. The decision function is thus asked to map newly incoming documents (d) in one (or more) class(es), according to their content.

The role of linguistic content in TC is twofold: from one side it is embodied by specific information with respect to entities and facts mentioned in documents. Proper Nouns for Companies, Location and Persons, or the events involving those entities (e.g. managing succession events as indicators of some topics, *Industry News*) are example of such type of linguistic content. This information is widely used within the IE area, involved in very granular and specific recognition. On the other hand content refers also to the set of *typical* words, i.e. expressions and terminological units that co-occur in a document or in documents of the same class. This provide an overall picture of what a topic is, and what it deals with. This second form of linguistic content here refers to:

- a clear separation between content words (i.e. open syntactic classes like nouns, verbs and adjectives) and other less relevant information (e.g. functional classes like prepositions or complex functional expressions *as far as* or *in order to*)
- the identification of role of each word in its corresponding contexts, able for example to distinguish verbal from nominal uses of a lemma.
- the identification of linguistically motivated structures that behave non compositionally, and thus require a completely different treatment (even in the probabilistic modeling) with respect to other phenomena. Possibly complex Proper Nouns (e.g. *Shell Transport & Trading Co. PLC*) are an example, as they should not be modeled similarly to common nouns in TC.

This less granular form of linguistic content is very useful in TC as it usually provides the core information on which quantitative models are developed. The accuracy in the recognition of the different components is clearly reflected in the accuracy of the TC or retrieval process. Empirical evidence on this relationship are still necessary, and our study aims to add further information to this issue.

In this paper we present an original classification model sensitive to linguistic content and characterized by a novel inference method (see section 3). For comparative purposes we review and discuss the other well-known TC techniques, section 2.3. The presented model departs from existing ones in two aspects:

- it uses a language-sensitive indexing process, as syntactic processing supports indexing by separating functional structures and the recognized proper nouns within the candidate term set.
- a specific inference technique (called RDS) for the classification decision

Extensive experimentation on different benchmarking, as well as real corpora, are then presented in Section 4. A prototype system, called TREVI (Basili et al., 1998a), has been used as a platform for all the experiments. In TREVI specific technologies to deal with linguistic content have been adopted and the positive contributions are suggested by the experiments. The discussion of the original aspects of the proposed method with respect to the TREVI performances is carried out in Section 5.

2. Designing a Text Classifier

The design of text classifiers foresees a set of tasks universally recognized by the IR community:

- *Corpus Pre-processing*
  The first phase is the pre-processing of a corpus consisting of the following sub-steps:
  - *Corpus Validation*, i.e. filtering, and formatting all the documents belonging to the corpus.

  - *Extraction of relevant words*. In this step a *stop list* is applied to eliminate function words that exhibit approximately equal frequencies in all documents of the collections.
  - *Word Stemming*, i.e. common suffixes are removed from text words by suffix-deletion system. The words after stemming are usually called *terms*.
  - *Corpus Splitting*: two parts are derived from the source corpus, the test-set, used for performance evaluation and the training-set used for learning.
  - *Term selection*, which is an attempt to remove non-informative terms from documents to improve categorization effectiveness and reduce computational complexity. Typical selection criteria are χ^2, information gain or document frequency.
- *Features design*: the linguistic information that characterize a document (and a class) are here selected. The simplest ones are single words. More complex features can be built as structured patterns (i.e. multiple word expressions), or by adding lexical information (i.e. word senses).
- *Weighting*: 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.
- *Similarity estimation* is modeled via operations in spaces of features. This can be carried out between couples of documents or between more complex combination of features (e.g. profiles as a combination of features coming from different representative documents). Usually quantitative models (i.e. metrics) are adopted for this.
- *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.

2.1. Profile and Example-driven Text Classification

Two main approaches to the construction of a non-parametric classifier have been proposed and experimented in literature (Lewis et al., 1996). *Profile-based* 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 previously categorized under C_i. This approach can be referred as *category-centered* 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. The majors advantages of this approach are the computational efficiency and the easy implementation.

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-centered* categorization.

2.2. Designing a Profile-based Classifier

The development of a profile-based classifier requires specialization of the above mentioned tasks:

- *Features design* and *Weighting, i.e. 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* in profile-based classifiers is always between documents and the above defined profiles;
- *Inference*: the decision function usually depends on the similarity scores. The most widely used inference methods in literature are: probability, fixed and proportional thresholding. These are

respectively called in (Yang, 1999): $Scut$ (a threshold for each class establishes if a document belong to it), $Rcut$ (the best k-ranked classes are assigned to each document) and $Pcut$ (the test-set document are assigned to the classes proportionally to their size)(Yang, 1999).

2.3. Text Classification Models

Several classifi cation models have been proposed in literature and their distinctive aspects will be here briefly summarized.

KK-Neighbor is an example-based classifi er, (Yang, 1994), making use of document to document similarity estimation that selects a class for a document through a kk-nearest heuristics. Rocchio (Ittner et al., 1995; Cohen and Singer, 1996) often refers to TC systems based on the Rocchio's formula for profi le estimation. RIPPER (Cohen and Singer, 1996) uses an extended notion of profi le, by learning contexts that are positively correlated with the target classes. A machine learning algorithms allows the "contexts" of a word w to affect how (or whether) presence/absence of w contribute actually to a classifi cation. CLASSI is a system that uses a neural network-based approach to text categorization (H.T. Ng and Low, 1997). The basic units of the network are only perceptrons. 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 allows to select relevant words (i.e. features), according to an information gain criterion, and, then, to predict categories according to the occurrence of word combinations in documents. CHARADE (I. Moulinier and Ganascia, 1996) and SWAP1 (Apté et al., 1994) use machine learning algorithms to inductively extract Disjunctive Normal Form rules from training documents. Sleeping Experts (EXPERTS) (Cohen and Singer, 1996) are learning algorithms that works on-line. They reduce the computation complexity of the training phase for large applications updating incrementally the weights of *n*-gram phrases. Naive Bayes (Tzeras and Artman, 1993) is a probabilistic classifi er that uses joint probabilities of words and categories to estimates the conditional probabilities of categories given a document. The naive approach refers to the assumption of word independence. Such assumption makes the computation of Naive Bayes classifi er far more effi cient than the exponential complexity of a pure Bayes approach (i.e. where predictors are made of word combinations).

Comparative analysis[1] has shown (e.g. (Yang, 1999)) that fi ve Reuters test sets exist and systems performs differently on them. In particular Table 1 reports the measured performances on the Reuters, version 3 (hereafter Reuters3 corpus).

Table 1: Breakeven points of widely known classifi ers using the $Scut$ policy on Reuters version 3

KK-neighbor	RIPPER	CLASSI	Dtree	SWAP1
85%	80%	80%	79%	79%

CHARADE	EXPERT	Rocchio	Naive Bayes	
78%	76%	75%	71%	

3. A Language-Sensitive Classification Model, $LSTC$

This section is dedicated to the defi nition of a novel profi le-based TC model, characterized by a systematic linguistic preprocessing of the data (section 3.1) and a statistical inference method (called RDS, section 3.3). This model will be hereafter called $LSTC$. Weighting in $LSTC$ is driven by different formulas, according to the experimental evidence that their performance is highly dependent on the corpora and tasks (i.e. number of classes). Next sections will introduce the related defi nitions for each specifi c aspect.

[1]It has been done by means of breakeven point that is the point where recall and precision assume the same value.

3.1. Language processing for Text Classification in TREVI

The TC model that is proposed in this paper has been used within the TREVI (Text Retrieval and Enrichment for Vital Information[2]), a system for Intelligent Text Retrieval and Enrichment.

Reuters is a member of the Consortium and has been used as a main "*User Case*" for the released prototype. TREVI components are servers cooperating to the processing, extraction, classification, enrichment and delivery of news. Basically two TREVI components contribute to the TC task:

- the $Parser$, i.e. a full linguistic preprocessor that take a normalized versions of the news and produces a set of grammatical (e.g. subj/obj relations) and semantic (e.g. word senses in an ontology) information for each text.
- a $Subject\ Identifier$, that according to the $Parser$ output and to the derived class profiles assigns one or more topics to each news. This is the proper TC (sub)system.

The $Parser$ in TREVI (Basili et al., 1998a) is a complex (sub)system combining tokenization, lemmatization (via a independent lexical server), Part-of-Speech tagging (Brill, 1992; Church, 1988) and robust parsing (Basili et al., 1998b). Details on the linguistic methods and algorithms for each phase can be found in the related publications (Basili et al., 1998a; Basili et al., 1998b) and will not be here described. The only relevant information is the outcome of the parsing process towards the $Subject\ Identifier$ component.

The $Parser$ is able to detect in documents the following set of information:

1. Possibly complex tokens and lemmas. Simple words (e.g. $bank$, $match$) as well as complex terminological expressions (e.g. entire noun phrases as "*bond issue*" or functional expressions as *in order to*) are detected and treated as atomic units during the later phases;
2. Proper Nouns (PNs). Set of domain (i.e. User) specific Named-Entities are recognized by extensive catalogs as well as by the application of special-purpose grammars. A typed set of proper nouns is derived from each news and they are treated by a separate process with respect to the other lemmas.
3. Syntactic Categories for lemmas. Each units of text (i.e. simple or complex terms) is assigned with a single Part-of-Speech (POS), (e.g. `N` for nouns, `V` for verbs). Indexes are thus extended with their own POS, so that verbal and nominal occurrences of a given lemma are independent (e.g. can/`V` is different from can/`N`)
4. Major grammatical relations (i.e. `Subj/Obj` relations among words) are detected with a significant accuracy (see (Basili et al., 1998b) for an evaluation of the robust parser). Each news is thus annotated also with basic structures made of significant constituents (verbs and their modifiers).

The classification model that we propose, hereafter $LSTC$, is a profile-based classifier using as features the document's lemmas associated with their part-of-speech (POS) labels (point 3). Only nouns, verbs and adjectives are considered candidates features, and the resulting indexes are couples `<lemma, POStag>`.

Furthermore Proper Nouns (PNs) are not part of the profile. Information related to them is linguistically different from the other lemmas. Different spaces (i.e. one for lemmas and another for PNs) are determined so that they require a different treatment.

Moreover, no stop list is used in TREVI, as POS tagging supplies the corresponding, and linguistically principled, filtering ability. It is expected that the overall process (i.e. recognition of functional units, proper nouns and POS tag assignment) supports the selection of the suitable candidate features (i.e. terms).

It is to be noticed that no grammatical relation (point 4) is used for indexing, so that only points 1,2 and 3 will influence the classification. These are all contributions that the $Parser$ component provides to the

[2]TREVI (Basili et al., 1998a) is a distributed object-oriented system, designed and developed by a European consortium under the TREVI ESPRIT project EP23311.

$Subject\ Identifier.$

3.2. Weighting in LSTC

In $LSTC$ two weighting schemes have been implemented and experimental results will be shown for both. The first is novel introducing new corpus-derived parameters: it will be referred as IWF. The second is the common weighting scheme associated with the Rocchio's classifier (Ittner et al., 1995).

3.2.1. IWF-based weighting In order to define the IWF weighting policy, a number of definitions is necessary. Given a training set, a feature $t \in \{t_1, ..., t_n\}$ to describe it, a generic document d_h of the corpus and the target set of classes C_1, C_2, ..., let the following notations express:

- N, i.e. the overall occurrences of features,
- F_t i.e. the occurrences of a feature t in the training set,
- f_t^h, i.e. the occurrences of the features t in the document d_h,

The document representations can be defined as:

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

while class profiles are:

$$\vec{C_i} = << t_1, \omega_{t_1}^i >, ..., < t_n, \omega_{t_n}^i >>$$

where document weights

$$\bar{\omega}_t^h = (IWF)^2 f_t^h$$

and, accordingly, class weights

$$\omega_t^i = (IWF)^2 \sum_{h \in C_i} f_t^h,$$

are defined, and 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:

$$s_{ih} = cos(\angle(\vec{C_i}, \vec{d_h})) = \frac{\sum_t \omega_t^i \bar{\omega}_t^h}{|\vec{C_i}||\vec{d_h}|} \quad (1)$$

The above model introduces two major differences with respect to the traditional weighting strategy (as for example used in SMART (Salton, 1991)). First, the *Inverse Word Frequencies* (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. In fact, the product $IWF \cdot \sum_{h \in C_i} f_t^h$ is too biased by the (global) frequency term $\sum_{h \in C_i} f_t^h$. In order to balance the IWF contribution its square is preferred. A similar adjustment technique is proposed in (Hull, 1994).

3.2.2. Rocchio weighting A second weighting scheme in $LSTC$ is based on the Rocchio's formula (Ittner et al., 1995). In this scheme two more quantities are needed:

- M, i.e. the number of documents in the training set.
- n_t, i.e. the number of documents in which the term t appears.
- $\grave{f}_t^h$ is 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

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

and class weights will be

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

Where R_i is the set of training documents belonging to class C_i and $\bar{R}_i$ are the documents not belonging to class C_i. The parameters β and γ control the relative impact of positive and negative examples on the classifier. In the experiments, described in what follows, the standard values β = 16 and γ = 4 found in (Cohen and Singer, 1996) have been used.

The Rocchio's weighting system includes the document and profile normalization inside the weights themselves, therefore the derived similarity function does not contain the normalization with respect to $\vec{C_i}$ and $\vec{d_h}$ (see Equation 1) as we observe in the following:

$$s_{hi} = cos(\angle(\vec{C_i}, \vec{d_h})) = \sum_t \omega_t^i \bar{\omega}_t^h$$

3.3. Inference via Relative Difference Scores - RDS

In order to select the suitable classes for a document, thresholding over s_{hi} is a widely adopted empirical criteria (see (Lewis, 1992) for a comparative evaluation). We defined a thresholding policy based on the 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 is used. Formally, for a given document h, M^i assumes the following values:

$$m_{hi} = \frac{\sum_{j=1}^{n} s_{hi} - s_{hj}}{n-1} \tag{2}$$

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:

$$m_{hi} > E(M^i) - \alpha_i StdDev(M^i) \tag{3}$$

where each α_i is a threshold (empirically determined to optimize recall and precision over the test data). Equation 3 is the inference method hereafter called RDS.

The tree main approaches to thresholding are *probability-based* ($Scut$), *fixed* ($Rcut$) and *proportional thresholding* ($Pcut$). Lewis ((Lewis, 1992)) has shown that none of these is outperforming the others. 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.

4. Experimental Evaluation

In order to evaluate our TC methodology according to the following objectives:

- the impact of linguistic information in Text Classifi cation (i.e. as an indirect/operational performance evaluation of the $Parser$)
- the improvement of the RDS technique over real data (for user-driven evaluation) as well as on well-known (i.e. assessed and already measured) data sets
- the influence of different weighting schemes on different test sets,

several experiments have been designed and implemented.
For extensive evaluation we collected three corpora. The corpora of news coming from users involved in the TREVI project that are:

- Reuters news, collected in a set of about 26,000 documents, and distributed throughout 20 (structured) classes. The main topics of this corpus are fi nancial (e.g "*Corporate Industrial*" or "*Market/Share*" and general (e.g. "*Sport*" or "*Elections*" categories. We will refer to this collection as the TREVI-Reuters corpus.
- HOS (Health On-Line) news, a collection of short medicine-related abstracts. The HOS corpus is made of about 5,000 documents distributed throughout 11 classes. Typical classes are "*Clinical Oncology*" vs. "*Endocrinology*"

As a reference collection the Reuters, version 3, corpus (Yang, 1999) has also been used and it will be hereafter referred as Reuters3. It includes 11,099 documents for 93 classes, with a fi xed splitting between test and learning data (3,309 vs. 7,789).

A fi rst set of experiments has been carried out on the TREVI data sets (i.e. TREVI-Reuters and HOS) in order to compare weighting schemes and inference methods over real data sets. In particular three weighting schemes (i.e. SMART, Rocchio's and IWF) have been tested and two different inference methods have been measured: simple $Scut$ and RDS. A second test on the Reuters3 benchmark has been carried out to assess the contribution of the original aspects of $LSTC$ (i.e. IWF-based weighting and RDS) over a well-known collection. Finally, a third set of experiments aimed to evaluate the actual contribution of linguistic information, by comparing the performances of the best TC models (as observed over the TREVI-Reuters and Reuters3 data sets) applied with and without the POS information.

4.1. Description of the Experiments

Experiment 1: Performances in TREVI The fi rst experiment has been run over the TREVI user data and the performances of the classifi ers obtained by adopting different weighting schemes and different thresholding policies have been measured. The adopted features for document and class representation in all these classifi ers are the couples `<Lemma,POStag>`. SMART refers here to the classical vector space weighting scheme applied to profi les as IWF. Table 2 reports the results on the TREVI Reuters corpus while Table 3 relates the HOS data set.

Table 2: Classifi er Performances on the TREVI-Reuters Corpus

	SMART	SMART+RDS	IWF+RDS	Rocchio	Rocchio+RDS
Breakeven Point	63%	72%	76%	62.78%	71.60%

Table 3: Classifier Performances on HOS Corpus

	Rocchio	Rocchio+ RDS	IWF	IWF+RDS
Breakeven Point	64.09 %	67.75%	45.85%	59.15%

The SMART model has been run using two different classification rules: SMART adopts probability thresholding ($Scut$), while SMART+RDS is applied by using the relative difference score (Eq. 3).

Experiment 2: Assessment over the Reuters3 corpus $LSTC$ has been also measured against the Reuters3 corpus. The breakeven points are reported in Table 4. In all these experiments the TREVI processing is adopted for indexing. In line with previous results RDS produces a significant improvement on both weighting schemes. Note that the performance of the Rocchio's formula alone (column 1) are different with respect to other results obtained in literature (e.g. 75% in (Yang, 1999)). This suggests that the linguistic processing (i.e. the only difference between other experiments and our measurement) provides some additional positive information. In order to better evaluate the impact of linguistic information, a further test of the Rocchio's model has been run (see column 5, Rocchio-POS+PN). The same lemmas described without POS tags have been used and combined with new indexes corresponding to the detected PNs[3]. A loss of performance suggests that linguistic processing has a non trivial effect on the system selectivity.

Table 4: Classifier Performances on Reuters 3 Corpus

	Rocchio	Rocchio+ RDS	IWF	IWF+RDS	Rocchio-POS+PN
Breakeven Point	78.46%	80.52%	62.21	66.80	76.72%

Experiment 3: Evaluating the Impact of Part-of-Speech information In order to understand the role of part-of-speech information in the above measured classification schemes we run the best performing classifiers without POS information. Indexing by lemmas has been applied, so that only lemmatization information is available during the weighting. Table 5 shows the result of this experiment with those previously obtained including POS tag information over the TREVI-Reuters corpus.

Table 5: Syntactic Information vs. Classification Accuracy on Trevi-Reuters

	IWF+RDS	IWF+RDS-Pos
Rec.	83.70%	83.02%
Prec.	70.86%	70.56%

It has to be stressed that, in the TREVI-Reuters corpus, in the set of 37,069 different indexes only 4,089 (11%) refer to ambiguous lemmas (i.e. lemmas with more than one POS tag). Table 6 describe recall and precision of the two indexing modalities over the Reuters3 corpus. In the Reuters3 corpus, we obtained 21,975 different indexes, and only 1,801 out of them (8%) refer to lemmas with more than one POS tag.

5. Discussion

Main objective of the experiments was to define an optimal version of the $LSTC$ model.

[3]In this experiment indexes with different POS tags after the $Parser$ processing have been merged (with cumulated frequencies) and PNs have been added as new indexes.

Table 6: Syntactic Information vs. Classification Accuracy on Reuters3

	Rocchio+RDS	Rocchio+RDS-Pos
Rec.	80.39%	79.91%
Prec.	80.68%	79.95%

A first result has been that RDS establishes as an effective method for classification inference. It always produces an increment of the performance with respect to any weighting scheme if compared with simpler thresholding policies (e.g. $Scut$). This has been shown on three large and heterogeneous corpora.

The increment varies from 13% (Table 2) to 2% (Table 4). These improvements are similar over any indexing policy. In Table 2 the increment is exactly the same for the two weighting models, SMART and Rocchio's. This systematic behavior suggests that RDS has an effect dependent mainly on the corpus and proportional to the inherent limitations of the weighting model. In Table 4 and 3 the weaker weighting policy (i.e. IWF) receives the best contribution (4.6% and 13.3% improvement).

RDS is a natural ways to think the classification inference that is more flexible with respect to fixed or proportional thresholding: it is less biased by the training set and can be easily adaptable to dynamically changing frameworks of use. 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. 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 to be more robust with respect to categories with different specificity. Finally, RDS 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.

The three evaluated corpora have shown that is very difficult to find out a TC model that is optimal over any corpus, as already pointed out in (Yang, 1999). The Rocchio's model performs better when well characterized profiles for *smaller* and more numerous classes are found. The IWF scheme is better performing on the first corpus that includes very generic classes poorly characterized by the profiles. For this reason the suitable selection of the weighting scheme is left underspecified in $LSTC$, and a pre-analysis should be always applied to the target corpora.

The other interesting aspect in $LSTC$ was the role of linguistic processing.

Reported results in literature (see Table 1) over the Reuters3 corpus suggests that the best performing classifier is the kNN (85%). A Rocchio based model is reported with a breakeven point of 75%. This contrasts with our evidence that is 78.46% (see Table 4). Note that the only difference with those experiments is the TREVI technology adopted for indexing. As discussed in section 3.1, the linguistic preprocessing differs from traditional methods as (i) no stoplist, (ii) no stemming are applied, (iii) recognition of proper nouns and (iv) POS information are available.

It is evident that lemmatization and POS tagging supply information similar to that obtained via stemming and stoplist adoption: in fact, only words POS-tagged as nouns, verbs and adjectives are used for indexing. Anyhow, results in Table 4 show that even without POS tagging and by adding proper nouns (i.e. using profiles that can be obtained via stoplists and stemming), performances are still higher than in (Yang, 1999). Note that the removal of POS tags and the inclusion of Proper Nouns has been applied over the set of indexes precalculated by the TREVI processor. This means that an influence over the selection of the candidate feature set (i.e. lemmas) persists in this experiment. As a result the small improvements, even in this degraded use of TREVI, is basically due to a better set of indexes than stemming and stoplist adoption can obtain. The improvement on the performances of the Rocchio's model on Reuters3 are thus to be imputed to the greater accuracy of the overall linguistic process and on the clear separation between lemmas (i.e. content words) and proper nouns.

The other departure from current text processing tools is the proper noun treatment. Actually they are not included in profiles, according to the fact that they cannot be reliably retained as *topic triggers*. The evaluation in Table 4 suggests that this is a correct policy as including proper nouns (+PN in column 5) produces a loss of performances (-1.8%).

It is also emerged that POS information, when added to the indexes, produces only small improvements (see Table 5 and 6). This is basically due to the small number of truly ambiguous lemmas (10% or 8%), so that the overall effect cannot be very evident.

Given the above evidences a quasi-optimal version of the $LSTC$ model can be obtained via a suitable use of weighting schemes and combining the TREVI linguistic processor with an RDS inference mechanism. This produces the best performance figure for linear classifiers on the Reuters3 data set (80.52%). Other more accurate classifiers have been also presented. The best figure on the Reuters3 corpus is obtained by the example-driven k-NN classifier (85%) and LLSF (85%), (Yang, 1994). However, their higher training and classification complexity makes their design and use more difficult within real operational domains. Other classifiers based on complex learning (e.g. Ripper, (Cohen and Singer, 1996)) or knowledge based algorithms (e.g. SWAP-1, (Apté et al., 1994)) show a similarly complex design, but their performances are comparable if not lower with respect to $LSTC$. For the relatively simple nature and applicability of the $LSTC$ model, it has been successfully adopted within the TREVI real application scenarios (Reuters and HOS). Its good performances are retained also in such new domains even with simpler weighting models (IWF vs. Rocchio's).

6. Conclusion

The proposed $LSTC$ model has shown very good performances on real and well-known (i.e. Reuters3) data sets. Extensive experimentation suggests the following as the main reasons:

- it makes use of a novel and effective inference technique,RDS
- $LSTC$ is based on a systematic linguistic preprocessing of texts able to
  - better support indexing by focusing on only the meaningful syntactic categories (e.g. real content words) and supporting lemmatization rather than stemming
  - produce an inherent corpus reduction by pruning the candidate feature set of less informative units, like function words and proper nouns
  - augment indexes by introducing the POS information. More indexes are thus derived.

In (Basili et al., 1999) we presented an inference technique (RDS) for text classification that improved performances of different classifiers (e.g SMART and Rocchio's) (about 9% in BEP). Current and extensive experiments over well-known data sets proofed that the improvement can be observed on any corpus, and with respect to different weighting policies. Moreover, the weaker is the weighting scheme the higher is the improvement produced by RDS.

On a reference (and widely tested) collection (Reuters, version 3) two weighting schemes have been compared (IWF and Rocchio's) and the best model (Rocchio's with RDS as an inference technique) has been also measured as the best performing linear classifier on such data set. This suggests that a model based on the linguistic preprocessing (as described in Section 3.1), the Rocchio's formula and RDS, that we call $LSTC$, can be effectively used in different (and heterogeneous) application domains. Some of the results reported in this paper have been derived during the development and design of an industrial prototype, the TREVI system. Test carried out within users (e.g. Reuters) operational scenarios have confirmed the accuracy of the results.

Further work is necessary to better establish the contribution of the different original aspects of $LSTC$ to the proofed outperforming results. First of all, other benchmarking data sets could be studied for comparative purposes. A final assessment of the RDS role would be obtained.

Second, the linguistic processing adopted in $LSTC$ seems to provide a better input information to the indexing schemes, so to result in a significant improvements to BEP figures reported in literature, (Yang, 1999). The systematic language processing adopted in TREVI supports input material of a better quality (i.e. POS tagged words, detection of complex functional units and proper nouns). A selective measurement of the contribution of each type of linguistic over more test collections is necessary to clearly establish their role.

However, $LSTC$ represents only the first potential language-sensitive approach to text classification, as it uses only a part of the material currently detected by the TREVI parser. Models that will be able to take into account other linguistic information (e.g. terminological units, descriptions of events and semantic typing of words) are under design. Accordingly, extension of the $LSTC$ model is also one of our short and medium term research objective.

7. Acknowledgement

We would like to thank our colleagues of the AI Research Group at Tor Vergata University of Rome for their contribution to methodological discussion and their support in making available experimental data.

References

Apté C., Damerau F., and Weiss S. (1994). Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12(3):233–251.

Basili R., Di Nanni M., Mazzucchelli L., Marabello M., and Pazienza M. (August 1998a). 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*.

Basili R., Pazienza M. T., and Zanzotto F. M. (1998b). Efficient parsing for information extraction. In *Proc. of the ECAI98*, Brighton, UK.

Brill E. (1992). A simple rule-based part of speech tagger. In *Proc. of the Third Applied Natural Language Processing, Povo, Trento, Italy*.

Church K. A. (1988). A stochastic parts program and noun phrase parser for unrestricted text. In *Proc. of Second Conference on Applied Natural Language Processing*.

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.

Grefenstette G. (1991). Short queries linguistic expansion techniques: Palliating one-word queries by providing intermediate structures to text. In Pazienza M. editor, *Information Extraction - A Multidisciplinary Approach to an Emerging Information Technology*. Springer Verlag, Berlin.

H.T. Ng W. G. and Low K. (1997). Features selection, perceptron learning, and a usability case study for text categorization. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 97')*, pages 67–73.

Hull D. (1994). Improving text retrieval for the routing problem using latent semantic indexing. In *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, pages 282–291, Dublin, IE.

I. Moulinier G. R. and Ganascia J. (1996). Text categorization: a symbolic approach. In *In Proceedings of the Fifth Annual Symposium on Document Analysis and Information Retrieval*.

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

Quinlan J. (1986). Induction of decision trees. In *Machine Learning*, pages 81–106.

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.

Tzeras K. and Artman S. (1993). Automatic indexing based on bayesian inference networks. In *Proceedings of 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 22–34.

Yang Y. (1994). Expert network: effective and effi cient 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*.