Automatic Expansion of Domain-Specific Lexicons by Term Categorization

HENRI AVANCINI
Consiglio Nazionale delle Ricerche, Italy
ALBERTO LAVELLI
ITC-irst, Italy
FABRIZIO SEBASTIANI
Consiglio Nazionale delle Ricerche, Italy
ROBERTO ZANOLI
ITC-irst, Italy

We discuss an approach to the automatic expansion of domain-specific lexicons by means of *term categorization*, a novel task employing techniques from information retrieval and machine learning. Specifically, we view the expansion of such lexicons as a process of learning previously unknown associations between terms and *domains* (i.e. disciplines, or fields of activity). The process generates, for each c_i in a set $C = \{c_1, \ldots, c_m\}$ of domains, a lexicon L_1^i, bootstrapping from an initial lexicon L_0^i and a set of documents θ given as input. The method is inspired by *text categorization*, the discipline concerned with labeling natural language texts with labels from a predefined set of domains, or categories. However, while text categorization deals with documents represented as vectors in a space of terms, we formulate the task of term categorization as one in which terms are (dually) represented as vectors in a space of documents, and in which terms (instead of documents) are labeled with domains. As a learning device we adopt a *boosting*-based method, since boosting (a) has demonstrated state-of-the-art effectiveness in a variety of text categorization applications, and (b) naturally allows for a form of "data cleaning", thereby making the process of generating a lexicon an iteration of generate-and-test steps. We present the results of a number of experiments using a set of domain-specific lexicons called WordNetDomains (which actually consists of an extension of WordNet), and performed using the documents in the Reuters Corpus Volume 1 as "implicit" representations for our terms.

Categories and Subject Descriptors: I.5.2 [**Pattern Recognition**]: Classifier design and evaluation; I.2.7 [**Artificial Intelligence**]: Natural Language Processing; H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing

General Terms: Algorithms, Experimentation, Theory

Additional Key Words and Phrases: Lexicons, Text Classification, Machine Learning

Author's address: H. Avancini and F. Sebastiani, Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, Via Giuseppe Moruzzi 1, 56124 Pisa, Italy. E-mail: henri.avancini@isti.cnr.it, fabrizio.sebastiani@isti.cnr.it. A. Lavelli and R. Zanoli, ITC-irst, Via Sommarive 18, 38050 Povo (TN), Italy. E-mail: lavelli@itc.it, zanoli@itc.it.

1. INTRODUCTION

The generation of *domain-specific lexicons* (i.e. lexicons consisting of terms pertaining to a given domain or discipline) is a task of increased applicative interest, since such lexicons are of the utmost importance in a variety of tasks pertaining to natural language processing and information access.

One of these tasks is to support text search and other information retrieval applications in the context of thematic, "vertical" portals (aka *vortals*)[1]. Vortals are a recent phenomenon in the World Wide Web, and have grown out of the users' needs for directories, services and information resources that are both rich in information and specific to their interests. This has led to Web sites that specialize in aggregating market-specific, "vertical" content and information. The evolution from the generic portals of the previous generation (such as Yahoo!) to today's vertical portals is just natural, and is no different from the evolution that the publishing industry has witnessed decades ago with the creation of thematic magazines, targeting specific categories of readers with specific needs. To read about the newest developments in ski construction technology, skiers read specialty magazines about skiing, and not generic newspapers, and skiing magazines is also where advertisers striving to target skiers place their ads in order to be the most effective. Vertical portals are the future of e-commerce and information seeking on the Internet, and supporting sophisticated information access capabilities by means of domain-specific lexical resources is thus for them of the utmost importance.

Domain-specific lexicons are also useful for *word-sense disambiguation* (WSD), the task of determining, given the occurrence o_w of a polysemous word w, the sense of o_w. Results from [Magnini et al. 2002] indicate that, given a word occurrence o_w whose possible senses $w_1, \ldots, w_s$ pertain each to a different domain $d_1, \ldots, d_s$, the domain d_i to which most of the terms occurring in the context of o_w pertain has a high probability of indicating the correct sense, i.e. there is a high probability that the right sense of o_w is w_i (see Figure 1). Domain-specific lexicons are then of fundamental importance for WSD, since it is important to know to which domains the terms occurring in the context of o_w pertain.

1.1 The problem

Unfortunately, the manual generation of domain-specific lexicons is expensive, since it requires the intervention of specialized manpower, i.e. lexicographers and domain experts working together. Besides being expensive, such a manual approach does not allow for fast response to rapidly emerging needs. In an era of frantic technical progress new disciplines emerge quickly, while others disappear as quickly; and in an era of evolving consumer needs, the same goes for new market niches. There is thus a need of cheaper and faster methods for answering application needs than manual lexicon generation. This manual approach is also prone to errors of omission, in that a lexicographer may easily overlook infrequent, non-obvious terms that are nonetheless important for many tasks.

Many applications also require that the lexicons be not only domain-specific, but also tailored to the specific data tackled in the application. For instance, in

[1]See e.g. `http://www.verticalportals.com/`

From the plush Connolly hide leather [sofa]$_F$ and [chairs]$_F$ in the [living room]$_F$ to the [Bang and Olufsen stereo]$_F$, and [remote control television]$_F$ complete with video, you're surrounded by the HIGHEST QUALITY. The [inlaid]$_F$ chequerboard top of the [coffee table]$_F$ houses all kind of [games]$_P$, including [backgammon]$_P$, [chess]$_P$ and [Scrabble]$_P$. You'll also find a selection of books, from Queen Victoria's Highland journals, to the very latest bestselling [thriller]$_P$. The [dinner table]$_F$ and [**chairs**]$_{??}$ are elegant yet comfortable, and you can be assured of the finest [tableware]$_F$ and crystal for meals at home.

Fig. 1. Word sense disambiguation by using domain information (example taken from Senseval-2, the international campaign for the experimental evaluation of WSD systems). Subscripts appended to terms indicate the domain to which the terms are known to belong (F=FURNITURE, P=PLAY, L=LITERATURE). The word occurrence to be disambiguated is the second occurrence of the word "chair" and its possible senses pertain to the domains LITERATURE, FURNITURE, and PLAY. Since most of the terms occurring in the context of this word occurrence belong to the FURNITURE domain, it seems reasonable to conclude that also this occurrence of "chair" should belong to FURNITURE.

query expansion for information retrieval systems addressing specialized document collections, terms synonymous or quasi-synonymous to the query terms are added to the query in order to retrieve more documents. In this case, the added terms should occur in the document collection, otherwise they are useless, and the relevant terms which occur in the document collection should potentially be added. Therefore, for this application the ideal domain-specific lexicon should contain all and only the technical terms that occur in the document collection under consideration, and should thus be generated directly from this latter.

1.2 Our proposal

In this paper we propose a methodology for the automatic expansion of domain-specific lexicons from a corpus of texts. This methodology relies on *term categorization*, a novel task that employs a combination of techniques from information retrieval (IR) and machine learning (ML). Specifically, we view the expansion of such lexicons as a process of learning previously unknown associations between terms and *domains* (i.e. disciplines, or fields of activity)[2].

The process generates, for each c_i in a set $C = \{c_1, \ldots, c_m\}$ of predefined domains, a lexicon L_1^i, bootstrapping from a lexicon L_0^i given as input. Associations between terms and domains are learnt from a set of *unlabeled* (i.e. not tagged with category or domain labels) textual documents θ (hereafter called *corpus*); iterating this process allows to expand the lexicon L_1^i into increasingly larger lexicons L_2^i, L_3^i, ..., as new corpora of unlabeled documents from which to learn become available. The process builds the lexicons $L_1 = \{L_1^1, \ldots, L_1^m\}$ for all the domains $C = \{c_1, \ldots, c_m\}$ in parallel, from the same corpus θ. The only requirement on θ is that at least some

[2]We want to point out that our use of the word "term" is somehow different from the one often used in natural language processing and terminology extraction, where it often denotes a *sequence* of lexical units expressing a concept of the domain of interest. Here we use this word in a neutral sense, i.e. without making any commitment as to its consisting of a single word or a sequence of words.

of the terms in each of the lexicons in $L_0 = \{L_0^1, \ldots, L_0^m\}$ should occur in it (if none among the terms in a lexicon L_0^j occurs in θ, then no new term is added to L_0^j).

The method we propose is inspired by *text categorization*, the activity of automatically building, by means of machine learning techniques, automatic text classifiers, i.e. programs capable of labeling natural language texts with (zero, one, or several) thematic categories from a predefined set $C = \{c_1, \ldots, c_m\}$ [Sebastiani 2002]. The construction of an automatic text classifier requires the availability of a set $\psi = \{\langle d_1, C_1\rangle, \ldots, \langle d_h, C_h\rangle\}$ of preclassified documents, where a pair $\langle d_j, C_j\rangle$ indicates that document d_j belongs to all and only the categories in $C_j \subseteq C$. A general inductive process (called the *learner*) automatically builds a classifier for the set C by learning the characteristics of C from a *training set* $Tr = \{\langle d_1, C_1\rangle, \ldots, \langle d_g, C_g\rangle\} \subset \psi$ of documents. Once a classifier has been built, its effectiveness (i.e. its capability to take the right categorization decisions) may be tested by applying it to the *test set* $Te = \{\langle d_{g+1}, C_{g+1}\rangle, \ldots, \langle d_h, C_h\rangle\} = \psi - Tr$ and checking the degree of correspondence between the decisions of the automatic classifier and those encoded in the corpus.

While the purpose of text categorization is that of classifying documents represented as vectors in a space of terms, the purpose of *term* categorization, as we formulate it, is (dually) that of classifying terms represented as vectors in a space of documents. In this task terms are thus items that may belong, and must thus be assigned, to (zero, one, or several) domains belonging to a predefined set. In other words, starting from a set Γ_0^i of preclassified terms, a new set of terms Γ_1^i is classified, and the terms in Γ_1^i which are deemed to belong to c_i are added to L_0^i to yield L_1^i. The set Γ_0^i is composed of lexicon L_0^i, acting as the set of "positive examples" of c_i, plus a set of terms known not to belong to c_i, acting as the set of "negative examples" of c_i.

For input to the learning device and to the term classifiers that this will eventually build, we use "bag of documents" representations for terms, dual to the "bag of terms" representations commonly used in text categorization. As the learning device we adopt the AdaBoost.MH^{KR} [Sebastiani et al. 2000; Nardiello et al. 2003], a more efficient variant of the AdaBoost.MH^{R} algorithm proposed in [Schapire and Singer 2000]. Both algorithms are an implementation of *boosting*, a method for supervised learning which has successfully been applied to many different domains and which has proven one of the best performers in text categorization applications so far. Boosting is based on the idea of relying on the collective judgment of a committee of classifiers that are trained sequentially; in training the k-th classifier special emphasis is placed on the correct categorization of the training examples which have proven harder for (i.e. have been misclassified more frequently by) the previously trained classifiers.

We chose a boosting approach not only because of its state-of-the-art effectiveness, but also because it naturally allows for a form of "data cleaning", which is useful in case a lexicographer wants to check the results and edit the newly generated lexicon. That is, in our term categorization context a boosting approach allows the lexicographer to easily inspect the preclassified terms for possible misclassifications, since the algorithm, apart from generating the new lexicon L_1^i, ranks the terms in L_0^i in terms of their "hardness", i.e. how successful the generated clas-

sifiers have been at correctly recognizing their domains. Since the highest ranked terms are the ones with the highest probability of being misclassified [Abney et al. 1999], the lexicographer can examine this list starting from the top and stopping where desired, removing the misclassified examples. The process of generating a domain-specific lexicon may then become an iteration of generate-and-test steps.

This paper is organized as follows. In Section 2 we describe how we represent terms by means of a "bag of documents" representation, while Section 3 describes ADABOOST.MH^{KR}, the boosting algorithm we employ for term classification [Sebastiani et al. 2000]. Section 4 discusses how to combine the indexing tools introduced in Section 2 with the boosting algorithm, and describes the possible roles of the lexicographer in the lexicon expansion phase. Section 5 describes the results of our experiments, in which we attempt to expand (in parallel) several domain-specific lexicons (42 in some experiments, 145 in others) by using a corpus of more than 800,000 documents. In Section 6 we review related work on the automated generation of lexical resources, and spell out the differences between our and existing approaches. Section 7 concludes, pointing to avenues for improvement.

2. REPRESENTING TERMS IN A SPACE OF DOCUMENTS

2.1 Text indexing

In text categorization applications, the process of building internal representations of texts is called *text indexing*. In text indexing, a document d_j is usually represented as a vector of term *weights* $\vec{d_j} = \langle w_{1j}, \ldots, w_{rj} \rangle$, where r is the cardinality of the *dictionary* and $0 \leq w_{kj} \leq 1$ represents, loosely speaking, the contribution of t_k to the specification of the (extensional) semantics of d_j. Usually, the dictionary is equated with the set of *terms* that occur at least once in at least α documents of Tr (with α a predefined threshold, typically ranging between 1 and 5). Different approaches to text indexing may result from different choices (i) as to what a term is, and (ii) as to how term weights should be computed. A frequent choice for (i) is to use single words (minus "stop words", i.e. topic-neutral words such as articles and prepositions, which are usually removed prior to indexing) or their stems. Different "weighting" functions may be used for tackling issue (ii), either of a probabilistic or of a statistical nature; a frequent choice is the (cosine-normalized) $tfidf$ function, which is spelled out in Section 2.2 and provides the inspiration for the "term indexing" function we are going to use in this work.

2.2 Abstract indexing and term indexing

Text indexing may be viewed as a particular instance of *abstract indexing*, a task in which "objects" are represented by means of "features", and whose underlying metaphor is, by and large, that the semantics of an object corresponds to the *bag of features* that "occur" in it[3].

[3] "Bag" is used here in its set-theoretic meaning, as a synonym of *multiset*, i.e. a set in which the same element may occur several times. In text indexing, adopting a "bag of words" model means assuming that the number of times that a given word occurs in the same document is semantically significant. "Set of words" models, in which this number is assumed not significant, are thus particular instances of bag of words models.

In order to illustrate an example of abstract indexing, let us define a *token* τ to be a specific occurrence of a given feature $f(\tau)$ in a given object $o(\tau)$, let T be the set of all tokens occurring in any of a set of objects O, and let F be the set of features of which the tokens in T are instances. Let us define the *feature frequency* $ff(f_k, o_j)$ of a feature f_k in an object o_j as the number of times f_k occurs in o_j, i.e.

$$ff(f_k, o_j) = |\{\tau \in T \mid f(\tau) = f_k \ \wedge \ o(\tau) = o_j\}| \tag{1}$$

We next define the *inverted object frequency* $iof(f_k)$ of a feature f_k as a function of the number of objects in which f_k occurs, i.e.

$$iof(f_k) \ = \ \log \frac{|O|}{|\{o_j \in O \mid \exists \tau \in T : f(\tau) = f_k \ \wedge \ o(\tau) = o_j\}|} \tag{2}$$

and the *weight* $w(f_k, o_j)$ of feature f_k in object o_j as

$$w_{kj} \ = \ w(f_k, o_j) = \frac{ff(f_k, o_j) \cdot iof(f_k)}{\sqrt{\sum_{s=1}^{|F|} (ff(f_s, o_j) \cdot iof(f_s))^2}} \tag{3}$$

We may consider the $w(f_k, o_j)$ function of Equation (3) as an *abstract indexing function*; that is, different instances of this function are obtained by specifying different choices for the set of objects O and set of features F. The well-known text indexing function $tfidf$, mentioned in Section 2.1, is obtained by equating O with the set of training documents and F with the set of terms that are contained in them; T, the set of occurrences of elements of F in the elements of O, thus becomes the set of term occurrences. Dually, a term indexing function may be obtained by switching the roles of F and O, i.e. equating O with the set of training terms and F with the set of documents in which they are contained. Note that T, the set of occurrences of elements of F in the elements of O, is thus again the set of term occurrences [Schäuble and Knaus 1992].

It is interesting to discuss the kind of intuitions (i.e. the "monotonicity assumptions", according to the terminology of [Zobel and Moffat 1998]) that Equations (1), (2) and (3) embody in the dual cases of text indexing and term indexing:

—Equation (1) suggests that when a feature occurs multiple times in an object, the feature characterizes the object to a higher degree. In text indexing, this indicates that the more often a term occurs in a document, the more it is representative of the content of the document. In term indexing, this indicates that the more often a term occurs in a document, the more the document is representative of the content of the term.

—Equation (2) suggests that the fewer the objects a feature occurs in, the more representative the feature is of the content of the objects in which it occurs. In text indexing, this means that terms that occur in too many documents are not very useful for identifying the content of documents. In term indexing, this means that the more distinct terms a document contains (which by and large means: the longer it is), the less useful it is for characterizing the semantics of a term it contains.

—The intuition ("length normalization") that supports Equation (3) is that weights computed by means of $ff(f_k, o_j) \cdot iof(f_k)$ need to be normalized in order to

prevent "longer objects" (i.e. ones in which many features occur) to emerge (e.g. to be scored higher in document-document similarity computations) just because of their length and not because of their content. In text indexing, this means that longer documents need to be deemphasized. In term indexing, this means instead that terms that occur in many documents need to be deemphasized[4].

This approach to term representation is very elegant, in that it is based on a minimal set of assumptions (namely, the "extensional" assumption that objects can be represented as bags of features, and the assumption that occurrence can be used as "featurehood") and can be instantiated by means of any indexing technique (here we have used cosine-normalized $tfidf$), either from the tradition of text indexing or not. Note also that any program or data structure that implements a text indexing function may be used straightaway, with no modification, for term indexing: one needs only to feed the program with the term identifiers in place of the document identifiers and viceversa[5].

A straightforward consequence of this approach is that the semantic relatedness between terms is viewed as a function of *term co-occurrence*, a heuristic notion which has been widely used in many past approaches to lexicon generation (see Section 6). In fact, according to our chosen approach, two terms have the highest similarity when they occur exactly in the same documents and with the same frequency, and are very similar when they co-occur in a high proportion of documents and with similar frequencies. However, while the past approaches discussed in Section 6 only deal with a "pure" version of co-occurrence, our term indexing approach brings about, for free, a *weighted* and *length-normalized* notion of co-occurrence, thus injecting higher sophistication into the measure of term similarity.

2.3 Document selection

Many classifier induction methods (and boosting is no exception) are computationally hard, and their computational cost is a function of the size r of the set of features. It is thus of key importance to be able to work with vectors shorter than r, which in text categorization is usually a number in the tens of thousands or more. For this, *feature selection* (FS) techniques are often used in text categorization to select, from the original set of r features (i.e. terms), a subset of $r' \ll r$ features that, when used for document indexing, yield the best categorization effectiveness. In our term categorization task, a dual version of this technique is used, in which it is a subset of documents, and not terms, that is selected.

[4]Incidentally, it is interesting to note that in switching from text indexing to term indexing, Equations (2) and (3) switch their roles: the intuition that terms occurring in many documents should be deemphasized is implemented by Equation (2) in text indexing and Equation (3) in term indexing, while the intuition that longer documents need to be deemphasized is implemented by Equation (3) in text indexing and Equation (2) in term indexing.

[5]There is thus a clear symmetry between terms and documents, in the sense that one may determine the meaning of the other, depending on one's viewpoint. The only aspect for which the symmetry breaks *in practice* is that, both in document and term indexing, we tend to *directly* pick a set of documents to work with, and the set of terms we work with is (*indirectly*) determined as a consequence, in the sense that it coincides with the set of terms appearing in the chosen documents. Therefore, it is usually the case that documents are the independent variables and terms are the dependent variables of our problem, whatever the problem.

Feature selection is usually beneficial because it reduces *overfitting*, i.e. the phenomenon by which a classifier tends to be better at classifying the data it has been trained on than at classifying other data. The value $(1 - \frac{r'}{r})$ is called the *aggressivity* of the selection; the higher this value, the smaller the set resulting from FS, and the higher the computational benefits. On the other hand, a high aggressivity may curtail the ability of the classifier to correctly "understand" the meaning of a document, since information that in principle may contribute to specify document meaning is removed. Therefore, deciding on the best level of aggressivity usually requires some experimentation.

A widely used approach to FS is the so-called *filtering* approach [John et al. 1994], which consists in selecting the $r' \ll r$ features that score highest according to a function that measures the "importance" of the feature for the categorization task. In a thorough comparative experiment, performed across different classifier induction methods and different benchmarks, [Yang and Pedersen 1997] have shown that *information gain*, defined as

$$IG(t_k, c_i) = \sum_{c \in \{c_i, \overline{c}_i\}} \sum_{t \in \{t_k, \overline{t}_k\}} P(t, c) \cdot \log \frac{P(t, c)}{P(t) \cdot P(c)} \tag{4}$$

is one of the most effective "filtering" functions, allowing aggressivity levels in the range [.90,.99] with no loss (or even with a small increase) of effectiveness. This contributes to explain the popularity of IG as a FS technique in text categorization (see [Sebastiani 2002, Section 5]). In Equation (4), probabilities are interpreted on an event space of documents (e.g. $P(\overline{t}_k, c_i)$ indicates the probability that, for a random document x, feature t_k does not occur in x and x belongs to category c_i), and are usually estimated by maximum likelihood (ML), i.e.

$$\hat{P}(t, c) = \frac{|\{d_j \in Tr : t \in d_j \ \wedge \ d_j \in c\}|}{|Tr|} \tag{5}$$

(where $t \in \{t_k, \overline{t}_k\}$ and $c \in \{c_i, \overline{c}_i\}$, and estimates are indicated, as usual, by carets) or by a version of ML with Laplace smoothing, i.e.

$$\begin{aligned} \hat{P}(t, c) &= \frac{1 + |\{d_j \in Tr : t \in d_j \ \wedge \ d_j \in c\}|}{\sum_{c \in \{c_i, \overline{c}_i\}} \sum_{t \in \{t_k, \overline{t}_k\}} (1 + |\{d_j \in Tr : t \in d_j \ \wedge \ d_j \in c\}|)} \\ &= \frac{1 + |\{d_j \in Tr : t \in d_j \ \wedge \ d_j \in c\}|}{4 + |Tr|} \end{aligned} \tag{6}$$

(of course, marginal probabilities are computed by summing over joint probabilities).

Information gain as specified in Equation (4) evaluates the feature t_k with respect to a specific category c_i; in order to assess the value of a feature t_k in a "global", category-independent sense, a "globalization" technique such as the maximum $IG_{max}(t_k) = \max_{i=1}^{m} IG(t_k, c_i)$ of its category-specific values may be adopted.

In our experiments, information gain will thus be used in order to select, from an initial set of documents, those that are deemed most significant for the classification task.

3. BOOSTING FOR TEXT CATEGORIZATION

3.1 AdaBoost.MH^R

As a learning device for term categorization we use AdaBoost.MH^{KR}, a boosting algorithm described in [Sebastiani et al. 2000] and subsequently improved in [Nardiello et al. 2003], which modifies and improves upon the AdaBoost.MH^R algorithm described in [Schapire and Singer 2000].

Boosting is a method for generating a classifier committee. *Classifier committees* are based on the intuition that, given a task that requires expert knowledge to perform, S independent experts may be better than one if their individual judgments are appropriately combined. In text categorization, the idea is to apply S different classifiers $\Phi_1, \ldots, \Phi_S$ to the same task of deciding under which set $C_j \subseteq C$ of categories document d_j should be classified, and then combine their outcome appropriately. In boosting algorithms, the S classifiers $\Phi_1, \ldots, \Phi_S$ (here called the *weak hypotheses*) forming the committee are obtained by the same learning method (here called the *weak learner*) and work on the same text representation. The key intuition of boosting is that the S weak hypotheses are to be trained not in a conceptually parallel and independent way, as in other classifier committees, but sequentially. In this way, the training of hypothesis Φ_i may take into account how hypotheses $\Phi_1, \ldots, \Phi_{i-1}$ perform on the training documents, and may concentrate on getting right those training documents on which $\Phi_1, \ldots, \Phi_{i-1}$ have performed worst.

AdaBoost.MH^R works by iteratively calling a *weak learner* to generate a sequence $\Phi_1, \ldots, \Phi_S$ of weak hypotheses; at the end of the iteration the final hypothesis Φ is obtained by a summation

$$\Phi(d_j, c_i) = \sum_{s=1}^{S} \Phi_s(d_j, c_i) \tag{7}$$

of these weak hypotheses. A weak hypothesis is a function $\Phi_s : \mathcal{D} \times C \rightarrow \mathbb{R}$, where $\mathcal{D}$ is the set of all possible documents. We interpret the sign of $\Phi_s(d_j, c_i)$ as the decision of Φ_s on whether d_j belongs to c_i (i.e. $\Phi_s(d_j, c_i) > 0$ means that d_j is believed to belong to c_i while $\Phi_s(d_j, c_i) < 0$ means it is believed not to belong to c_i), and the absolute value of $\Phi_s(d_j, c_i)$ (indicated by $|\Phi_s(d_j, c_i)|$) as the strength of this belief.

At each iteration s AdaBoost.MH^R applies the newly generated weak hypothesis Φ_s to the training set and uses the results to update a distribution D_s of weights on the training pairs $\langle d_j, c_i \rangle$. The weight $D_{s+1}(d_j, c_i)$ is meant to capture how effective $\Phi_1, \ldots, \Phi_s$ were in correctly deciding whether the training document d_j belongs to category c_i or not. By passing (together with the training set Tr) this distribution to the weak learner, AdaBoost.MH^R forces this latter to generate a new weak hypothesis Φ_{s+1} that concentrates on the pairs with the highest weight, i.e. those that had proven harder to classify for the previous weak hypotheses. The final distribution D_S encodes how effective the generated classifier committee has been in classifying the training documents. In practice, the training documents have thus been ranked, as a side effect of the learning process, in order of how hard it was to classify them for the generated classifiers, and it has been shown [Abney

et al. 1999; Shinnou 2001] that this roughly corresponds to their probability of having been incorrectly labelled by the human expert.

The initial distribution D_1 is uniform. At each iteration s all the weights $D_s(d_j, c_i)$ are updated to $D_{s+1}(d_j, c_i)$ according to the rule

$$D_{s+1}(d_j, c_i) = \frac{D_s(d_j, c_i) \exp(-C_j[c_i] \cdot \Phi_s(d_j, c_i))}{Z_s} \tag{8}$$

where $C_j[c_i]$ is defined to be 1 if $c_i \in C_j$ and -1 otherwise, and

$$Z_s = \sum_{i=1}^{|\mathcal{C}|} \sum_{j=1}^{|Tr|} D_s(d_j, c_i) \exp(-C_j[c_i] \cdot \Phi_s(d_j, c_i)) \tag{9}$$

is a normalization factor.

Each document d_j is represented by a vector $\langle w_{1j}, \ldots, w_{|T|j} \rangle$ of $|T|$ binary weights, where $T = \{t_1, \ldots, t_{|T|}\}$ is the set of terms. The weak hypotheses AdaBoost.MH^R uses are *real-valued decision stumps*, i.e. functions of the form

$$\Phi_s(d_j, c_i) = \begin{cases} a_{0i} & \text{if } w_{kj} = 0 \\ a_{1i} & \text{if } w_{kj} = 1 \end{cases} \tag{10}$$

where $t_k \in T$, a_{0i} and a_{1i} are real-valued constants. The choices for t_k, a_{0i} and a_{1i} are in general different for each iteration, and are made according to a policy that attempts to minimize Z_s, since this is known to be an error-minimization policy (although not an optimal one) [Schapire and Singer 1999].

AdaBoost.MH^R chooses weak hypotheses of the form described in Equation 10 by a two step-process:

(1) For each term $t_k \in T$ it pre-selects, among all weak hypotheses that have t_k as the "pivot term", the one (indicated by Φ^k_{best}) for which Z_s is minimum. From a computational point of view this is the easier step, since Schapire and Singer [Schapire and Singer 1999] provide the provably optimal values for Φ^k_{best} as a function of $D_t(d_j, c_i)$, w_{kj} and $C_j[c_i]$.

(2) Among all the hypotheses $\Phi^1_{best}, \ldots, \Phi^{|T|}_{best}$ pre-selected for the $|T|$ different terms, it selects the one (indicated by Φ_s) for which Z_s is minimum. From a computational point of view this is the harder step, since this is $O(|T|)$, which in text categorization applications is typically in the tens of thousands.

3.2 AdaBoost.MH^{KR}

In [Sebastiani et al. 2000] a variant of AdaBoost.MH^R, called AdaBoost.MH^{KR} (for AdaBoost.MH^R *with K-fold real-valued predictions*), has been proposed. This algorithm differs from AdaBoost.MH^R in the policy according to which weak hypotheses are chosen, since it is based on the construction, at each iteration s of the boosting process, of a *complex weak hypothesis* (CWH) consisting of a sub-committee of *simple weak hypotheses* (SWHs) $\Phi^1_s, \ldots, \Phi^{K(s)}_s$, each of which has the form described in Equation 10. These are generated by means of the same process described in Section 3.1, but for the fact that at iteration s, instead of selecting and using only the best term t_k (i.e. the one which brings about the smallest Z_s), it selects the best $K(s)$ terms and use them in order to generate $K(s)$

SWHs $\Phi_s^1, \ldots, \Phi_s^{K(s)}$. The CWH is then produced by grouping $\Phi_s^1, \ldots, \Phi_s^{K(s)}$ into a sub-committee

$$\Phi_s(d_j, c_i) = \frac{1}{K(s)} \sum_{q=1}^{K(s)} \Phi_s^q(d_j, c_i) \tag{11}$$

that uses the simple arithmetic mean as the combination rule. For updating the distribution it still applies Equations 8 and 9, where Φ_s is now defined by Equation 11. The final hypothesis is computed by plugging Equation 11 into Equation 7, thus obtaining

$$\Phi(d_j, c_i) = \sum_{s=1}^{S} \alpha_s \frac{1}{K(s)} \sum_{q=1}^{K(s)} \Phi_s^q(d_j, c_i) \tag{12}$$

The number $K(s)$ of SWHs to include in the sub-committee is obtained using a simple heuristics which adds a constant C to K every N iterations, using a fixed value for N and using a value of 1 for $K(1)$, i.e. $K(s) = 1 + C\lfloor \frac{s-1}{N} \rfloor$.

In this work we used an improved version of AdaBoost.MH^{KR} presented in [Nardiello et al. 2003] in which each SWH in a sub-committee is weighted by a value which is inversely proportional to its score. This means replacing Equation 11 with

$$\Phi_s(d_j, c_i) = \frac{1}{K(s)} \sum_{q=1}^{K(s)} \frac{Z_q^{-1}}{\sum_{j=1}^{K(s)} Z_j^{-1}} \cdot \Phi_s^q(d_j, c_i) \tag{13}$$

The rationale of this choice is that the weight associated to a SWH that contributes more to maximizing effectiveness should be higher. This is especially important in the first iterations since, as noted in [Sebastiani et al. 2000], it is here that the variance of the Z_s values of members of the same sub-committee is higher.

4. GENERATING DOMAIN-SPECIFIC LEXICONS BY SUPERVISED LEARNING

4.1 Operational methodology

We are now ready to describe the overall process that we will follow for the expansion of domain-specific lexicons. We start from a set of domain-specific lexicons $L_0 = \{L_0^1, \ldots, L_0^m\}$, one for each domain in $C = \{c_1, \ldots, c_m\}$, and from a corpus θ. We index the terms that occur in L_0 by means of the term indexing technique described in Section 2.2; this yields, for each term t_k, a representation consisting of a vector of weighted documents, the length of the vector being $r = |\theta|$. We then reduce the size of these vectors to the desired size $r' \ll r$ by applying the document selection technique described in Section 2.3. By using $L_0 = \{L_0^1, \ldots, L_0^m\}$ as a training set, we then generate m classifiers $\Phi = \{\Phi^1, \ldots, \Phi^m\}$ by applying the AdaBoost.MH^{KR} algorithm. While generating the classifiers, AdaBoost.MH^{KR} also produces, for each domain c_i, a ranking of the terms in L_0^i in terms of how hard it was for the Φ_i to classify them correctly, which basically corresponds to their probability of their being misclassified examples. The lexicographer can then, if desired, inspect L_0 and remove the misclassified examples, if any (possibly rerunning, especially if these latter were a substantial number, AdaBoost.MH^{KR} on the "cleaned" version of L_0). At this

point, the terms occurring in θ that AdaBoost.MH^{KR} has classified under c_i are added (possibly, after being checked by the lexicographer) to L_0^i, yielding L_1^i.

This bootstrapping process can be further iterated, by using new text corpora from which to "extract" new terms. In this case an alternative approach is to involve the lexicographer only after the last iteration, and not after each iteration. For instance, [Riloff and Shepherd 1999] and [Thelen and Riloff 2002] perform several iterations of a similar process, at each iteration adding to the training set (without human intervention) the new items that have been attributed to the category with the highest confidence. After the last iteration, a lexicographer inspects the list of added terms and decides which one to remove, if any. This latter approach has the advantage of requiring the intervention of the lexicographer only once, but has the disadvantage that spurious terms added to a lexicon at early iterations can cause, if not promptly removed, new spurious ones to be added in the next iterations, thereby generating a domino effect. It thus requires a high precision on the part of the process.

4.2 Experimental methodology

The process we have described in Section 4.1 is the one that we would apply in an operational setting. In an experimental setting, instead, we are also interested in evaluating the effectiveness of our approach on a benchmark. The difference with the process outlined in Section 4.1 is that at the beginning of the process the lexicon L_0 is split into a training set and a test set; the classifiers are learnt from the training set, and are then tested on the test set by checking how good they are at extracting the terms in the test set from the corpus θ.

We will comply with standard text categorization practice in evaluating term categorization effectiveness by a combination of *precision* (π), the percentage of positive categorization decisions that turn out to be correct, and *recall* (ρ), the percentage of positive, correct categorization decisions that are actually taken. Since most classifiers can be tuned to emphasize one at the expense of the other, only combinations of the two are usually considered significant. Following common practice, as a measure combining the two we will adopt their harmonic mean, i.e. $F_1 = \frac{2\pi\rho}{\pi+\rho}$. Effectiveness will be computed with reference to the contingency table illustrated in Table I. When effectiveness is computed for several domains, the results for individual domains must be averaged in some way; we will do this both by *microaveraging* ("domains count proportionally to the number of their positive training examples"), i.e.

$$\pi^{\mu} = \frac{TP}{TP+FP} = \frac{\sum_{i=1}^{|\mathcal{C}|} TP_i}{\sum_{i=1}^{|\mathcal{C}|}(TP_i+FP_i)}$$

$$\rho^{\mu} = \frac{TP}{TP+FN} = \frac{\sum_{i=1}^{|\mathcal{C}|} TP_i}{\sum_{i=1}^{|\mathcal{C}|}(TP_i+FN_i)}$$

and by *macroaveraging* ("all domains count the same"), i.e.

$$\pi^{M} = \frac{\sum_{i=1}^{|\mathcal{C}|} \pi_i}{m} \qquad \rho^{M} = \frac{\sum_{i=1}^{|\mathcal{C}|} \rho_i}{m}$$

Here, "μ" and "M" indicate microaveraging and macroaveraging, respectively, while

Domain c_i		expert judgments	
		YES	**NO**
classifier judgments	**YES**	TP_i	FP_i
	NO	FN_i	TN_i

Table I. The contingency table for domain c_i. Here, FP_i (*false positives wrt* c_i) is the number of test terms incorrectly classified under c_i; TN_i (*true negatives wrt* c_i), TP_i (*true positives wrt* c_i) and FN_i (*false negatives wrt* c_i) are defined accordingly.

the other symbols are as defined in Table I. Microaveraging rewards classifiers that behave well on *frequent domains* (i.e. domains with many positive test examples), while classifiers that perform well also on infrequent domains are emphasized by macroaveraging. Whether one or the other should be adopted obviously depends on the application requirements.

5. EXPERIMENTS

In order to test our approach according to the methodology described in Section 4.2 we need two types of resources: (i) a corpus θ, which provides the "implicit" representation for terms, and (ii) a set of domain-specific lexicons $L_0 = \{L_0^1, \ldots, L_0^m\}$.

We now describe the resources we used in our experiments.

5.1 The corpus

As the corpus θ we used the Reuters Corpus Volume 1 (RCV1), a set of documents recently made available by Reuters[6] for text categorization experimentation and consisting of the 806,812 news stories produced by Reuters from 20 Aug 1996 to 19 Aug 1997; all news stories are in English, and have 109 distinct terms per document on average [Rose et al. 2002]. Note that, although the texts of RCV1 are labeled by thematic categories, we did not make use of such labels (nor it would have made sense to use them, given that these categories are very different from, and basically unrelated to, the ones we are working with). The reason why we chose this corpus is that it provides a large enough set of data, and that it is domain-generic, i.e. that it is not related in any semantically significant sense to the domains we used in the experiments.

5.2 The lexicons

As the domain-specific lexicons we used an extension of WordNet called WordNet-Domains. WordNet [Fellbaum 1998] is a large, widely available, domain-generic, monolingual, machine-readable dictionary in which sets of synonymous words are grouped into 99,642 synonym sets (or *synsets*) organized into a directed acyclic graph. WordNet contains 173,941 word senses and 129,502 different lemmas; among these latter, 94,474 are nouns. In this work, we will refer to WordNet version 1.6.

In WordNet only a few synsets are labeled with thematic categories, mainly contained in the glosses. This limitation is overcome in WordNetDomains, an extension of WordNet built by [Magnini and Cavaglià 2000] in which each synset has been

[6] `http://www.reuters.com/`

bats-man *n* (*pl* -men) **1** (CRICKET) player who bats: *He's a good batsman but no good as a bowler.* **2** (AVIATION) man who uses a pair of bats (like those used in table-tennis) to guide an aircraft as and after it touches down (e g on the deck of an aircraft-carrier).

Fig. 2. Two domains (CRICKET and AVIATION) used as word sense discrimination devices in a dictionary entry for "batsman". The example is drawn from the Oxford Advanced Learner's Dictionary of Current English, 3rd edition.

labeled with one or more from a set of 164 thematic categories, called *domains*[7]. The 164 domains of WordNetDomains are a subset of the categories belonging to the classification scheme of Dewey Decimal Classification (DDC); example domains are ZOOLOGY, SPORT, and BASKETBALL[8]. In this work, we will refer to WordNetDomains "Version 1.0 - 070501".

These 164 domains had been chosen by Magnini and Cavaglià from the much larger set of DDC categories since they are the most popular labels used in dictionaries for sense discrimination purposes. Domains have long been used in lexicography (where they are sometimes called *subject labels*) to mark technical usages of words (see Figure 2 for an example). Although they convey useful information for sense discrimination, they typically tag only a small portion of a dictionary. WordNetDomains extends instead the coverage of domain labels to an entire, existing lexical database, i.e. WordNet.

A domain may include synsets of different syntactic types: for instance, the MEDICINE domain groups together senses from Nouns, such as `doctor#1` (the first among several senses of the word "doctor") and `hospital#1`, and from Verbs, such as `operate#7`. A domain may include senses from different WordNet subhierarchies. For example, SPORT contains senses such as `athlete#1`, which descends from `life_form#1`; `game_equipment#1`, from `physical_object#1`; `sport#1`, from `act#2`; and `playing_field#1`, from `location#1`. Note that domains may group different senses of the same word into different thematic clusters, with the side effect of reducing word polysemy in WordNet.

The annotation methodology used in [Magnini and Cavaglià 2000] for creating WordNetDomains was mainly manual, and based on lexico-semantic criteria which take advantage of the already existing conceptual relations in WordNet.

In some of the experiments reported in this paper, we used a coarser-grained variant of WordNetDomains, called WordNetDomains(42)[9]. This was obtained by Magnini and Cavaglià from WordNetDomains by considering only 42 highly relevant domains, and tagging by a given domain c_i also the synsets that, in WordNetDomains, were tagged by the domains immediately related to c_i in a hierarchical sense (that is, the parent domain of c_i and all the children domains of c_i). For instance, the domain SPORT is retained into WordNetDomains(42), and labels the synsets that it originally labeled in WordNetDomains *plus* the ones that in WordNetDomains

[7]From the point of view of our term categorization task, the fact that more than one domain may be attached to the same synset means that ours is a *multi-label* categorization task [Sebastiani 2002, Section 2.2].

[8]WordNetDomains is publicly available at
`http://tcc.itc.it/research/textec/topics/disambiguation/download.html`

[9]See Table II for the list of domains included in WordNetDomains(42).

Domains	Training Terms	Test Terms	Domains	Training Terms	Test Terms
Administration	1739	855	Law	722	382
Agriculture	128	61	Linguistics	568	297
Alimentation	1070	482	Literature	323	180
Anthropology	538	254	Mathematics	249	120
Archaeology	32	15	Medicine	1077	573
Architecture	1578	730	Military	661	320
Art	889	440	Pedagogy	269	149
Artisanship	49	31	Philosophy	83	64
Astrology	21	12	Physics	893	393
Astronomy	181	69	Play	514	230
Biology	3375	1433	Politics	569	333
Body-Care	72	30	Psychology	993	581
Chemistry	834	397	Publishing	208	101
Commerce	368	189	Religion	869	405
Computer-Science	263	89	Sexuality	150	72
Earth	2201	1088	Sociology	408	209
Economy	1446	700	Sport	1002	455
Engineering	372	164	Telecommunication	285	130
Fashion	492	182	Tourism	305	152
History	619	354	Transport	983	480
Industry	496	205	Veterinary	25	17

Table II. The domains in WordNetDomains(42), each with the number of training and test terms occurring at least once in the full RCV1.

were labeled by its children domains (e.g. Volley, Basketball, ...) or under its parent domain (Free-time), which are not retained in WordNetDomains(42). Since Free-time has another child (Play) which is also retained in WordNetDomains(42), the synsets originally labeled by Free-time will now be labeled also by Play, and will thus have multiple labels. However, that a synset may belong to multiple domains is true in general, i.e. these domains need not have any particular relation in the hierarchy.

This restriction to the 42 most significant domains is meant to bring about a good compromise between the conflicting needs of avoiding data sparseness and preventing the loss of relevant semantic information. These 42 domains belong to 5 groups, where the domains in a given group are all the children of the same WordNetDomains domain, which is however not retained into WordNetDomains(42); for example, one group is formed by Sport and Play, which are both children of Free-time (not included into WordNetDomains(42)).

5.3 Structure of the experiments

Figures 3 to 9 report several experiments run for different choices (i) of the subset of RCV1 chosen as the corpus θ, (ii) of the set L_0 of domain-specific lexicons, (iii) of the set of documents that act as features in the classification process, and (iv) of what counts as a "feature" in this process. We first describe the structure of a generic experiment, and then go on to describe the sequence of different experiments we run.

In our experiments so far we considered only nouns, thereby discarding words tagged by other syntactic types. Nouns are more relevant from an applicative point of view (e.g. in query expansion), and are probably easier to classify within domains, since they tend to be more domain-specific than e.g. verbs or adverbs. We plan to also consider words other than nouns in future experiments.

In each experiment, we perform a *document filtering* phase by discarding all documents that do not contain any term from the training lexicon Tr, since they do not contribute in representing the meaning of training terms, and thus could not possibly be of any help in building the classifiers. We then lemmatize all remaining

documents and annotate the lemmas with part-of-speech tags, both by means of the TreeTagger package [Schmid 1994]. We also use the WordNet morphological analyzer in order to resolve ambiguities and lemmatization mistakes.

Next, we perform a *term filtering* phase, in which we discard (i) all "empty" training terms, i.e. training terms that are not contained in any document of θ, since they could not possibly contribute to learning the classifiers; (ii) empty test terms, i.e. test terms that are not contained in any document of θ, since no algorithm that extracts terms from corpora could possibly extract them; (iii) terms that occur in θ but belong neither to the training set Tr nor to the test set Te, since they do not play any role in our experiments.

The final set of terms that results from this process is randomly divided into a training set Tr (consisting of two thirds of the entire set) and a test set Te (one third). As negative training examples of domain c_i we choose all the training terms that are not positive examples of c_i, since more sophisticated policies are useless with boosting[10]. We repeat each experiment several times by considering only training and test terms occurring in at least x documents, so the curves describing our various experiments all plot F_1 as a function of x. Each curve in Figures 3 to 9 is thus the result of six different experiments (one for each value of $x \in \{1, 5, 10, 15, 30, 60\}$), since for each different value of x the experiment has to be repeated anew[11].

Note that in this entire process we do not consider the grouping of terms into synsets; i.e., the lexical units of interest in our application are the terms, and not the synsets. The reason is that RCV1 is not a sense-tagged corpus and, since a given WordNet term may belong to more than one synset, for any term occurrence τ it is not clear to which synset τ refers to.

5.4 The results

5.4.1 *Experiment 1: The "basic experiment".* In our first set of experiments (see the curves marked by black triangles in Figure 3) we used only a subset of the RCV1 corpus (about 8% of its total size), corresponding to the news stories produced in an entire month (1 Nov 1996 to 30 Nov 1996 – 67,953 documents), with the purpose of getting a feeling for the dimensions of the problem that need investigation; for the same reason, in the same set of experiments we used only a small number of boosting iterations (500). There are 6,636 terms in WordNetDomains(42) after the term filtering phase described above.

[10] In general, choosing a more sophisticated policy for the selection of negative examples basically means adopting a technique for the selection of "quasi-negative" examples [Dumais and Chen 2000; Ng et al. 1997], i.e. negative examples that are closer to the decision boundary, since they are the most difficult to discriminate, and provide thus more information to the learning process. But this is useless with boosting, since the boosting process is already designed to emphasize the training examples that are most difficult to discriminate; therefore, such examples need not be selected in advance, since they will be identified and emphasized anyway during the learning process.

[11] For reasons of space we avoid to include precision and recall values, and only report F_1 values; a complete listing of all precision and recall values for the experiments reported in this paper is included as an "online appendix" of this paper, which can be downloaded from `http://faure.iei.pi.cnr.it/~fabrizio/DSL-experiments.xls`

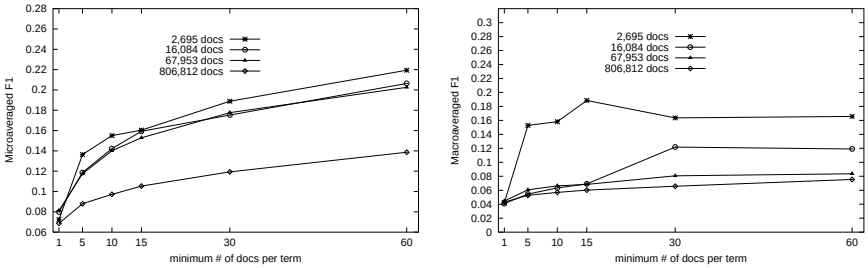

Fig. 3. Results obtained with the AdaBoost.MH^{KR} learner on subsets of RCV1 corresponding to one day's, one week's, one month's, and one year's worth of RCV1 data, and on the lexicons in WordNetDomains(42). Plots report micro-averaged F_1 (leftmost) and macro-averaged F_1 (rightmost) as a function of x, which represents the minimal number of documents in which training and test terms must occur in order to be taken into consideration.

The low values obtained for F_1 are mostly the result of low recall values, while precision tends to be much higher; for instance, the F_1^μ value of .203, obtained for $x = 60$, is the result of the values $\pi^\mu = .760$ and $\rho^\mu = .117$. One way of improving F_1 could be tuning AdaBoost.MH^{KR} so as to increase recall at the expense of precision, since F_1 is maximized when precision equals recall. Although this would be easy (e.g. by using the simple utility-theoretic technique described in [Schapire et al. 1998], which consists in altering the initial distribution D_1), we did not pursue this line of work, for the simple reason that it would not bring interesting insights into the problem[12].

The results in Figure 3 show a constant and definite improvement when higher values of x are used, despite the fact that, as we found, higher levels of x mean a higher degree of polysemy, i.e. a higher average number of labels per term (e.g. this increases from 1.66 for $x = 1$ to 2.25 for $x = 60$), which tends to confuse a learning device. This behaviour is not surprising, since when a term occurs e.g. in one document only, this means that only one entry in the vector that represents the term is non-null (i.e. significant); this means that the vector representation of this

[12]That our F_1 values result from the combination of low recall and high precision values is true throughout the experiments described in the next pages. We conjecture that this is due to the following fact. In term (and text) categorization, unlike in many other machine learning applications, the number of negative examples of a given category c_i is usually overwhelmingly higher than the number of its positive examples (e.g. there are far less terms belonging to the domain Agriculture than terms not belonging to it). If we use accuracy (the converse of error, i.e. $A = 1 - E$) as a measure of effectiveness, it will be very easy to generate an "effective" classifier, since we simply need to generate the classifier that assigns every item d_j to $\overline{c}_i$ (*the trivial rejector*); in this way, the very few positive examples will have been misclassified, but the very many negative examples will have been correctly classified. This means that, when we use accuracy or error as the effectiveness measures that guide the learning process in boosting (or other learning mechanisms based on explicit effectiveness maximization), we end up with a classifier that tends to behave like the trivial rejector, i.e. emphasize precision at the expense of recall. Note in fact that the trivial rejector has maximum precision (since it never wrongly classifies a document under c_i) but minimum recall (since it never correctly classifies a document under c_i). In sum, relying, as we did, on a learning device based on explicit *error* minimization means generating classifiers with very high precision but very low recall.

term is scarcely significant[13]. This is in sharp contrast with what happens in text categorization, in which the number of non-null entries in the vector representing a document equals the number of distinct terms contained in the document, and is usually at least in the hundreds.

In all the experiments that follow, we used the previously described set of experiments as a sort of "baseline", testing whether modifying our approach along one or the other dimension of the problem could bring about a significant performance improvement. Note that, although the set of documents considered in the previous set of experiments is a *subset* of RCV1, it is a quite sizeable one, since it consists of 67,953 documents (more than 5 times the documents of the Reuters-21578 collection, currently still the standard benchmark of text categorization research).

The next paragraphs discuss the various dimensions of the problem that we explored.

5.4.2 *Experiment 2: Using the full RCV1.* In our next set of experiments we run our system on the entire set of 806,812 RCV1 documents (this means 27,048 terms left in WordNetDomains(42) after term filtering), since we wanted to test whether performance can be improved by increasing significantly the number of documents from which terms have to be "extracted". That this should happen might seem a plausible hypothesis, since more documents mean, on average, a higher number of occurrences per term (on average, a training term occurs in 135.59 different documents in the "one month" set of experiments and in 1,201.87 ones in the "one year" set of experiments), hence a more reliable indication of the typical contexts in which a given term occurs.

The results of this set of experiments (reported in Figure 3) indicate that this is not the case, since in going from 67,953 documents to 806,812 documents, performance deteriorates. This trend resulting from the comparison of experiments run on one year's and one month's worth of documents is confirmed by two other sets of experiments we run using one week's (1 Nov 1996 to 7 Nov 1996 – 16,003 documents) and one day's (1 Nov 1996 – 2,689 documents) worth of documents (on average, a training term occurs in 13.88 different documents in the "one day" set of experiments and in 46.45 ones in the "one week" set of experiments); altogether, the four sets of experiments clearly show that the less the documents, the better the performance.

A clear explanation of this fact is that this move produces a sharp decrease of the ratio between the number of training objects and the number of features that describe these objects (see Figure 4), a ratio that is conceptually akin to the one between the constraints of a problem and the number of its free variables. In other words, using the full RCV1 brings about an underconstrained problem, and the classifier tends to overfit the training data.

In future experiments we plan to use even smaller numbers of features, in order to determine the optimal number of features one should work with.

[13]By contrast, a fairly common term may have thousands of non-null entries in the vector that represents it. This means that we are dealing with an application in which the variance of the sparseness of the different vectors is tremendously high. This problem alone might deserve further investigation, since this situation is rather unique within the field of vectorial representations.

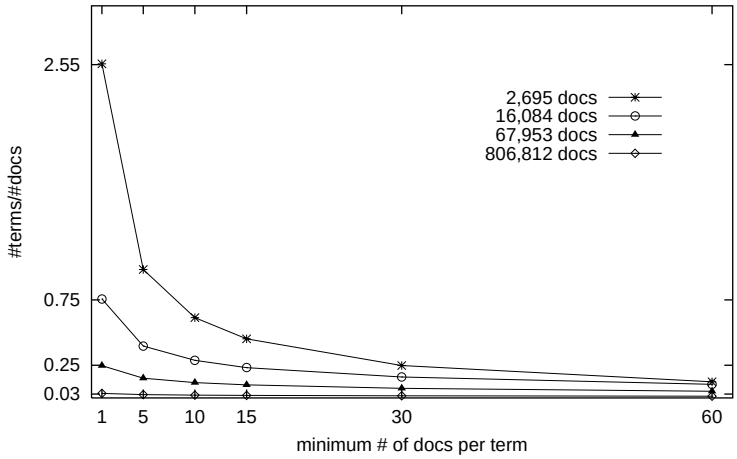

Fig. 4. Ratio between the number of training terms and the number of documents considered as a function of the x parameter.

5.4.3 *Experiment 3: Using document selection.* Our next set of experiments were aimed at verifying whether it may be the case that performance is depressed by RCV1 containing too many documents that are not significant enough in determining the "meaning" of our terms. These experiments consisted in first applying a pass of document selection aimed at selecting, from the 806,812 of RCV1, the documents that are the best discriminators between the presence and the absence of a domain, and then using the "shortened" term representations in which only the selected documents are retained as dimensions of the representation space. Following common text categorization practice, we scored each RCV1 document by the *information gain* function globalized by means of the f_{max} method (as described in Section 2.3), and retained only the top-scoring documents.

In the first set of experiments we retained the top 67,953 documents, corresponding to 8.72% of the original documents; this is exactly the same number of documents that were used in the "one month" set of experiments of Section 5.4.1, and this number was chosen in order to test what increase in performance (if any) could be achieved by replacing a set of k documents with a set of k *good* documents. For the same reason, our second set of experiments was run with the 16,084 top-scoring documents (corresponding to 2.06% of the original documents), and our third one with the 2,695 top-scoring documents (0.35% of the original documents), thus using exactly the same numbers of documents as used in the "one week" and "one day" experiments of Section 5.4.2.

The first observation (see Figure 5) is that each of these new sets of experiments produces an improvement with respect to the "one year" set of experiments of Section 5.4.2; the best performance was obtained with 0.35% of the original documents and the worst with 8.72%. However, at this point it is not clear whether this benefit is due to feature selection or is simply due to the trend already noticed in Section 5.4.2, according to which smaller numbers of features tend to produce

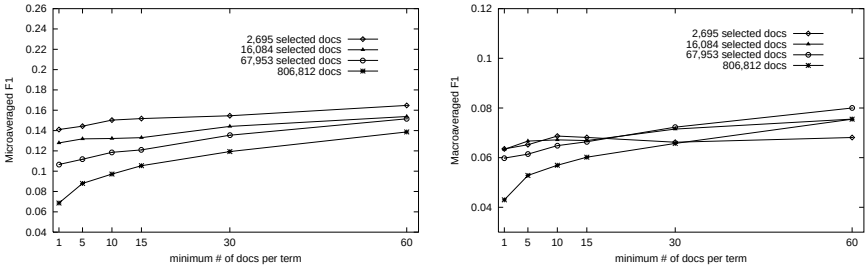

Fig. 5. Results obtained with the AdaBoost.MH^{KR} learner on the lexicons in WordNetDomains(42) and various amounts of documents selected from RCV1 via feature selection based on information gain.

better results. In order to determine this, in Figure 6 we compare the performance obtained by using s "random" documents (as from the experiments of Sections 5.4.1 and 5.4.2) with that obtained by using s "good" documents (i.e. obtained through feature selection), where s ranges on {67,953; 16,084; 2,695}.

Here, results are more difficult to explain, since it seems that feature selection is really helpful for low values of the x parameter (e.g. for $x = 1$ performance is always much higher when feature selection was performed) while it is detrimental for high values of x (e.g. for $x = 60$ performance is always higher when feature selection was *not* performed). The improvement for the low values of x is easy to explain, since classification theory tells us that s highly discriminating features are better than s random features when discrimination power is measured with respect to the target categories. The decrease in performance for the high values of x is more difficult to explain. We conjecture that this might be due to the fact that the experiments run in Sections 5.4.1 and 5.4.2 do not really use s *random* documents, since these documents are temporally coherent, i.e. they form a "subinterval" of the year being considered. Since the same subinterval typically contains several news stories about the same event, this means that when a term occurs in several temporally coherent documents there is a high chance that most if not all these occurrences pertain to the same sense (i.e. the degree of polysemy is reduced), and thus are more helpful in determining the domains of the words they co-occur with. In order to clarify this aspect in the future we intend to run further experiments on the monosemic portion of WordNet.

5.4.4 *Experiment 4: Using sentences instead of documents.* In our next set of experiments we reverted to the original set of documents of Section 5.4.1 and tested whether sentences can be better features than full documents in term categorization. That this might be the case is plausible since, given that (as observed in Section 2.2) our approach fundamentally relies on (the weighted and length-normalized version of) term co-occurrence, identifying the context in which co-occurrence is most significant is important. And it might be plausible to believe that the longer the context, the less significant the co-occurrence of two terms is in indicating that they belong to the same domain.

We run the set of experiments by segmenting each of the 67,953 documents

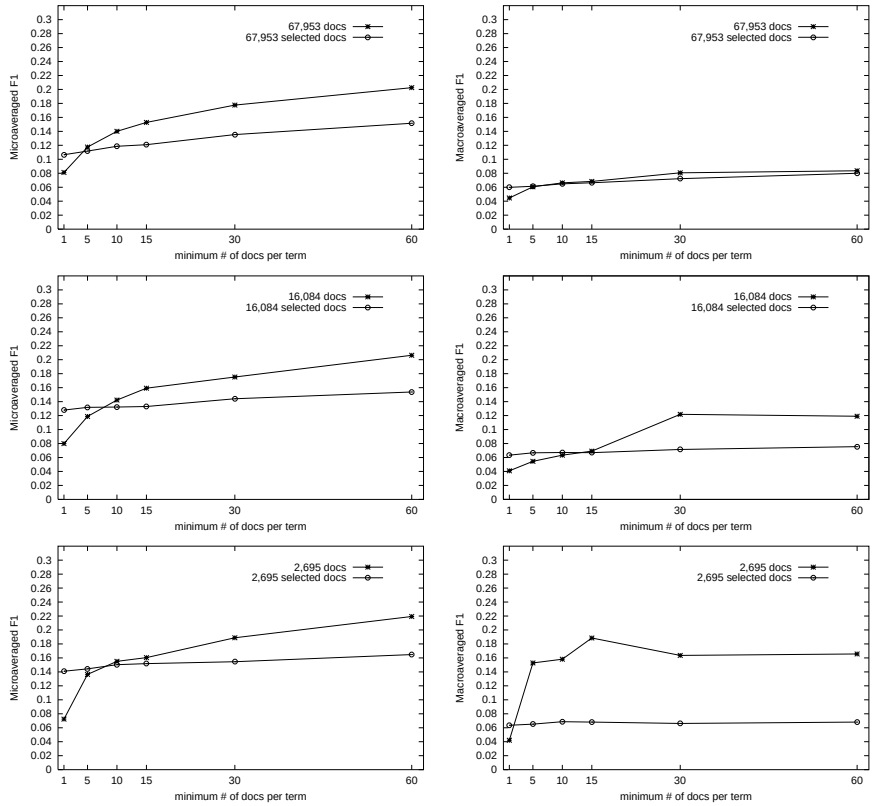

Fig. 6. Results obtained with the AdaBoost.MH^{KR} learner on the lexicons in WordNetDomains(42); each figure compares the difference between using s random documents and using s documents obtained via feature selection based on information gain ((top): $s = 67,953$; (mid): $s = 16,084$; (bottom): $s = 2,695$).

into sentences (i.e. using the full stop as separator) and considering each of the resulting 714,352 sentences as a "document"[14]. However, if we use all 714,352 sentences, we substantially decrease the ratio between the number of objects and the number of features that describe these objects, which means that, according to the observations we made in Section 5.4.2, our performance will likely decrease. In fact, this turns out to be the case, as shown in Figure 7.

As a result, we performed a further experiment in which we apply feature selection to the 714,352 sentences, until we obtain a set of 67,953 top-scoring sentences. This is the same number of documents that was contained in the set of documents resulting from feature selection in the experiments of Section 5.4.3; we deliberately chose this numbers, since we wanted to compare if s "good" sentences are better than s "good" documents at discriminating domains.

[14]We did not run an experiment using paragraphs instead of sentences (i.e. using a carriage return as the separator) since most paragraphs in RCV1 documents consist of single sentences.

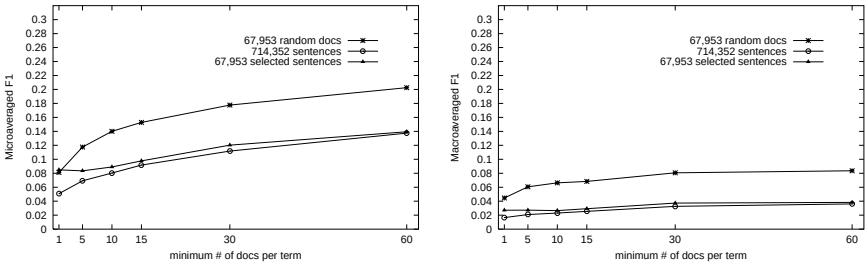

Fig. 7. Results obtained with the AdaBoost.MH^{KR} learner on the lexicons in WordNetDomains(42); each figure compares the difference between using 67,953 random documents, using all the sentences obtained by segmenting these 67,953 random documents, and using 67,953 sentences obtained from these latter via feature selection based on information gain.

The results (reported in Figure 7) seem to indicate that for very aggressive feature selection, i.e. when we retain only very few features, sentences are largely better features than documents, while this advantage tends to disappear for less aggressive levels. Note also that in the former situation performance is remarkably stable across all values of x, indicating that this setting is especially suitable for the situations in which one might want to extract very rare words.

This result seems consistent with an observation by Sahlgren [Sahlgren 2004], who conjectures that co-occurrence within shorter linguistic contexts tends to be more indicative of true synonymy (or quasi-synonymy), while co-occurrence in larger contexts tends to be more indicative of what he calls "topical relatedness", which is exactly our notion of a term being domain-specific.

5.4.5 *Experiment 5: Augmenting the number of boosting iterations.* In our next set of experiments we reverted to the full RCV1 with no feature selection and to using full documents instead of sentences, and tested whether augmenting the number of AdaBoost.MH^{KR} iterations could improve the performance significantly. The results of this set of experiments have been fairly encouraging, since for $x = 1$ the value of F_1^{μ} increases from .068 (for 500 iterations) to .099 (1500 iterations) to .116 (2500 iterations). This improvement is due to a sharp increase in recall, while precision stays basically constant. We plan to explore this dimension of the problem more thoroughly in future experiments.

5.4.6 *Experiment 6: Using the full WordNetDomains.* In a further set of experiments we reverted to the standard number of 500 iterations and we used the full WordNetDomains instead of its subset WordNetDomains(42); this actually means using 145 domains, and not the full set of 164 domains that make up WordNetDomains, since 19 domains from WordNetDomains consist only of terms than never occur in RCV1, and have thus to be removed from consideration.

The aim of this set of experiments was testing whether the use of domains that are more data-sparse than the ones we had previously used would deteriorate the performance significantly. Again, this would be a plausible hypothesis, since less training data usually means worse performance.

This turned out not to be the case, as can be seen in Figure 8. The likely reason

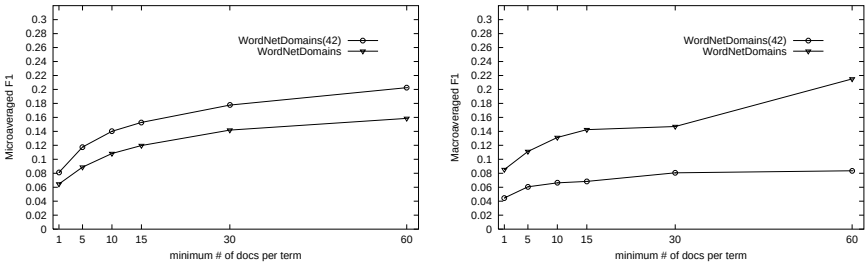

Fig. 8. Results obtained with the AdaBoost.MH^{KR} learner on one month's worth of RCV1 data and the lexicons either in WordNetDomains(42) or WordNetDomains, with documents as textual units.

for this apparently surprising result is that, although the domains in WordNetDomains contain less training data than the ones in WordNetDomains(42), the data they contain are more significant, since they are more focused. For instance, while the terms contained in the WordNetDomains domain Baseball are all focused on baseball, the terms contained in the WordNetDomains(42) domain Sport are more heterogeneous, since they pertain to different sports.

5.4.7 *Experiment 7: Switching to a different learner.* In order to check how dependent our results are on the choice of AdaBoost.MH^{KR} as learner, we performed a set of experiments in which AdaBoost.MH^{KR} is replaced by a support vector machine (SVM) learner implemented in the SVMlight package (version 3.5) [Joachims 1999], a well-known top performer of the text categorization field. SVM is a method that attempts to learn a hyperplane in $|\mathcal{T}|$-dimensional space that separates the positive training examples from the negative ones with the maximum possible margin, i.e. such that the distance between the hyperplane and the training examples that are closest to it is maximum; results in computational learning theory indicate that this tends to minimize the generalization error, i.e. the error of the resulting classifier on yet unseen examples. We simply opted for the default parameter setting of SVMlight; in particular, this means that a linear kernel was used, which should be a fairly good choice anyway since linear kernels have been found to be the best performing ones in applications involving text [Joachims 2001]. As in the case of boosting, as negative training examples of domain c_i we chose all the training terms that are not positive examples of c_i, since more sophisticated policies are useless with SVMs too[15]. For better comparability with AdaBoost.MH^{KR}, we run the same experiments as run with AdaBoost.MH^{KR} on one month's worth data (days from 01.11.1996 to 30.11.1996), with the same values of x as tested in the experiments of Figure 3.

As shown in Figure 9, SVMlight did not perform better than our AdaBoost.MH^{KR} learner; actually, the performance of SVMlight is almost uniformly worse than

[15]Similarly to the case of boosting, SVMs already identify and emphasize the negative examples that are close to the decision boundary (some of these will actually be the "support vectors" that determine the optimal decision surface), thus making their identification in a previous phase useless.

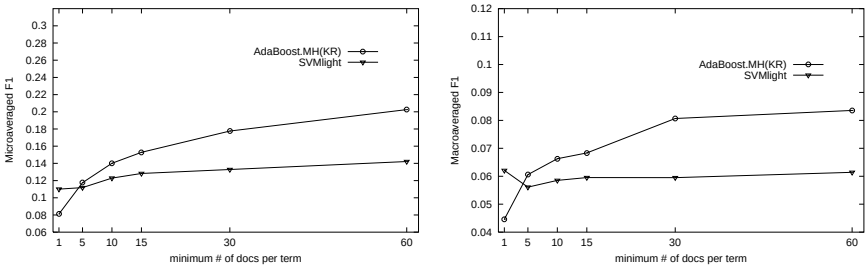

Fig. 9. Comparison of the results obtained on one month's worth of RCV1 data with the AdaBoost.MH^{KR} learner and with SVMlight.

that of AdaBoost.MH^{KR}, especially in the experiments with high values of x, and especially for macroaveraged effectiveness. However, we should note that the region on which SVMlight performs better than AdaBoost.MH^{KR} (namely, the low values of x) is an important one, since these are the terms that occur quite infrequently, and by virtue of this they may be the most important ones to extract automatically, since they are the ones whom a lexicographer might easily miss when manually generating a lexicon.

This is yet another confirmation that the lexicon expansion task is a hard one, since SVMs have been top performers practically in every machine learning application they have been put at, including text categorization.

As in the case of boosting, here too our F_1 values are the result of low recall and high precision. Again, the fact that SVMs are inherently biased towards error minimization (see Footnote 12) seems the likely cause for this fact.

5.4.8 *Experiment 8: Switching to a different representation.* In order to check how dependent our results are on the choice of the representation of terms discussed in Section 2.2, we performed a final set of experiments in which an alternative representation for terms was used. This representation, which we will call the *term co-occurrence representation* in order to contrast it with the *term occurrence representation* of Section 2.1 and the *document occurrence representation* of Section 2.2, represents a term t_j by a vector $\vec{t_j} = \langle w_{1j}, \ldots, w_{rj} \rangle$ of weighted *terms*, where r is the cardinality of the dictionary (i.e. the set of terms that occur at least once in at least α documents of Tr – see Section 2.1 for comparison purposes) and the weight $0 \leq w_{kj} \leq 1$ represents, loosely speaking, the degree of semantic association between t_j and t_k as measured by the frequency with which they co-occur in the documents (note that it is always the case that $w_{jj} = 1$; but the practical effect of this fact is irrelevant). This representation is quite popular in the computational linguistics literature (see [Dagan 2000] for a review), where it has been used for various purposes including word-sense disambiguation [Gale et al. 1993], the extraction of lexical collocations [Smadja 1993], and the extraction of syntactic relationships [Dagan et al. 1995].

Similarly to the case of the term/text occurrence representations, several alternative weighting functions may be used. Here we rely, again, on a normalized $tfidf$-like weighting function, analogous to the one discussed in Section 2.2. More

in detail,

—The $tf(t_k, t_j)$ component of weight w_{kj} is now the number of documents in which t_k and t_j co-occur. This component emphasizes the weight of terms t_k that often co-occur with the term t_j we want to represent.

—The $idf(t_k)$ component of weight w_{kj} is the logarithm of the ratio between the size of the vocabulary and the number of different terms in the vocabulary with which t_k co-occurs at least once. This component de-emphasizes the weight of terms t_k that tend to occur with many other terms, and whose co-occurrence with the term t_j we want to represent is thus less significant.

—The final weight w_{kj} is obtained by cosine-normalizing the results of the previous two steps, i.e. $w_{kj} = \frac{tf(t_k, t_j) \cdot idf(t_k)}{\sqrt{\sum_{s=1}^{r}(tf(t_s, t_j) \cdot idf(t_s))^2}}$.

The results of these experiments, which we have run with both AdaBoost.MH^{KR} and SVMlight as learners, are reported in Figure 10. It can be seen that the term co-occurrence representation performs significantly (although not breathtakingly) better. We think the reason may be due to the fact that this representation captures some phenomena related to semantic similarity better than the document occurrence representation. For instance, two perfect synonyms t_1 and t_2 may be represented by fairly dissimilar vectors according to the document occurrence representation, since an author might typically use, in a given document, either the one or the other term, but not both, for better consistency. If *all* authors used this policy, t_1 and t_2 would never co-occur in the same document, hence yielding two highly different vectors. This needs not be a problem with the term co-occurrence representation, since the two terms need not frequently co-occur with each other in order to be represented by highly similar vectors: they only need to frequently co-occur with the same terms, and this can plausibly happen given their perfect synonymy.

5.5 No baseline?

Note that in this paper we present no baseline, either published or new, against which to compare our results, for the simple fact that there are no previous results, in terms of both precision and recall, on the term categorization task as we conceive it here.

To our knowledge, only [Riloff and Shepherd 1999], [Roark and Charniak 1998] and [Thelen and Riloff 2002] have approached the problem of extending a domain-specific lexicon with new terms drawn from a text corpus. However, there are key differences between their evaluation methodology and ours, which make comparisons problematic and unreliable.

First, their training terms (which they call *seed words*) have not been chosen randomly from a domain-specific dictionary, but have been carefully selected through a manual process by the authors themselves. For instance, [Riloff and Shepherd 1999] choose words that are "frequent in the domain" and that are "(relatively) unambiguous". Of course, their approach makes the task easier, since it allows the "best" terms to be selected for training.

Second, [Riloff and Shepherd 1999] and [Roark and Charniak 1998] extract the

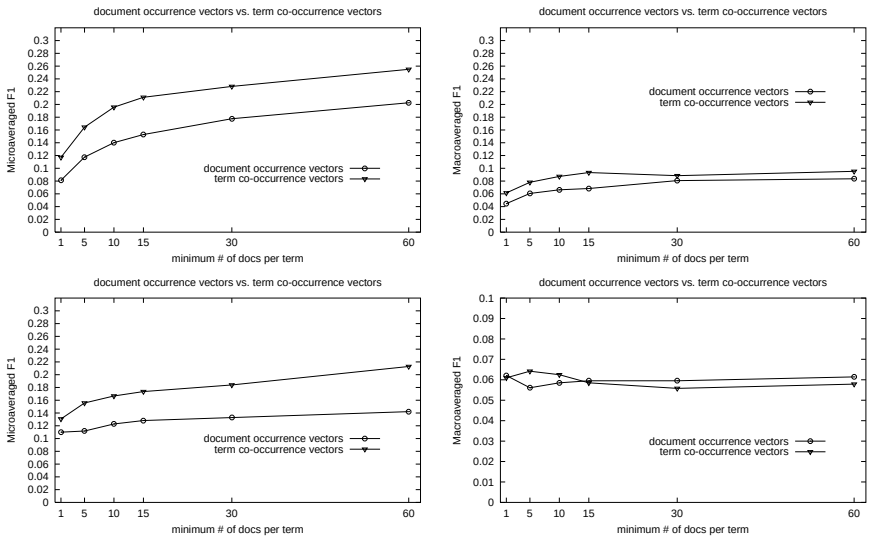

Fig. 10. Comparison of the results obtained with AdaBoost.MH^{KR} (top) and SVMlight (bottom) on one month's worth of RCV1 data with the standard "document occurrence" representation and the "term co-occurrence" representation.

terms from texts that are known to be strongly related to several among the domains of interest[16], which makes the task easier than ours. Conversely, by using generic texts, (i) we can also expand domains for which no collection of texts is available, and (ii) we are able to expand domain-specific lexicons for (any set of) multiple domains in parallel *from the same unlabeled text corpus*, a task that, to the best of our knowledge, has never been investigated.

Third, the evaluation methodology of [Riloff and Shepherd 1999] and [Roark and Charniak 1998] is manual and subjective, in the sense that the authors themselves manually check the results of their own experiments, judging, for each returned term, how reasonable the inclusion of the term in the lexicon is[17]. This sharply contrasts with our evaluation methodology, which is completely automatic (since we measure the proficiency of our system at discovering terms about the domain by the capability of the system to replicate the lexicon generation work previously performed by a lexicographer), can be replicated by other researchers, and is unaffected by possible experimenter's bias.

Fourth, checking one's results for "reasonableness", as [Riloff and Shepherd 1999] and [Roark and Charniak 1998] do, means that one can only ("subjectively") measure precision (i.e. whether the terms spotted by the algorithm do in fact belong to

[16]The texts used here are the MUC-4 corpus, which contains texts about terrorism. Categories of interest are, among others, Military, Terrorist, Weapon.

[17]For instance, [Riloff and Shepherd 1999] judge a word classified into a category correct also if they judge that "the word refers to a part of a member of the category", thereby judging the words `cartridge` and `clips` to belong to the domain Weapons. This looks to us a loose notion of category membership, and anyway points to the pitfalls of "subjective" evaluation methodologies.

the domain), but not recall (i.e. whether the terms belonging to the domain have actually been spotted by the algorithm)[18]. Again, this is in sharp contrast with our methodology, which ("objectively") measures precision, recall, and a combination of them. Also, note that in terms of precision, the only measure that [Riloff and Shepherd 1999] and [Roark and Charniak 1998] (subjectively) compute, our algorithm fares very well, mostly scoring higher than 70% (see Section 5.4.1)[19].

Aside from methodological issues, we should also note that our experiments are much larger in size that the ones presented in the quoted works. For instance, [Thelen and Riloff 2002] work with about 1,700 texts, while we work with about 806,000; they report results on 6 domains, while we do on 42 or 145.

5.6 Why is term categorization harder than text categorization?

A fundamental observation that can be made from the results of our experiments is that the F_1 scores are much lower than the ones usually obtained in text categorization, even if using the same kind of "extensional representation + supervised learning" approach and the same top-performing learners. It is well known, for instance, that the very same SVM learner we have used in our experiments obtains microaveraged F_1 scores higher than .80 on the very same corpus (RCV1) we have used [Ault and Yang 2001; Lewis et al. 2003]; it is true that the set of classes used in these latter experiments is different from the ones we have used in ours, but this fact alone cannot by itself justify such a large difference in performance. In other words, it is evident that term categorization is a harder task than text categorization.

Why this is so is not immediately clear, since the metaphor according to which the meaning of a text "coincides" with the terms it contains (a metaphor that has proven so successful in text categorization) is in principle no more powerful or intuitive than the dual metaphor according to which the meaning of a term "coincides" with the texts it is contained in (or with the similar metaphor according to which the meaning of a term "coincides" with the set of words it co-occurs with, as used in the experiments of Section 5.4.8).

We conjecture that the substantial difference in performance between the text and term categorization cases might be due to the fact that, while in text categorization there are often features with very high discriminative power, this does not seem to be the case for term categorization. To realize this, consider that in text categorization a perfect discriminator for category c_i is a term that occurs in all positive examples of c_i and never occurs in any negative examples of c_i. That such a perfect discriminator might exist in a real text categorization application is difficult but not impossible (when RCV1 is used as a text categorization collection,

[18] [Thelen and Riloff 2002] present "recall" figures too for their experiments. But their measure is not true recall, since their "gold standard" is not given by the set of all correctly classified terms, but by the set of correctly classified terms extracted in a preprocessing phase by a term extraction algorithm. This set might thus not contain all terms, and might contain non-terms, which means that in principle their measure is not true recall. Also, they do not specify whether their recall figures are microaveraged or macroaveraged.

[19] The precision figures presented in these works are not very high either. For instance, [Riloff and Shepherd 1999] report a macroaveraged precision-at-50 of .3363 (while our best macroaveraged precision figures are above .78).

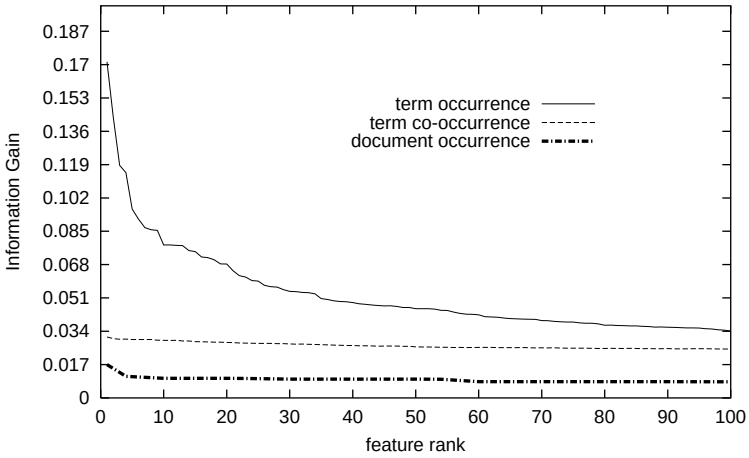

Fig. 11. Comparison of the absolute values of the information gain function for features in the term occurrence representation (as used in text categorization), in the document occurrence representation, and in the term co-occurrence representation (as used in term categorization), as a function of the rank of such features. The results are reported for the 100 top-scoring features only.

there are categories that have almost perfect discriminators). In our "document occurrence" representation for the term categorization task, instead, a perfect discriminator for category c_i is a document in which all terms belonging to c_i occur and in which no term not belonging to c_i occurs. That such a perfect discriminator might exist seems not only difficult, but plainly impossible, since when in natural language we want to state even the simplest fact about c_i we also require terms which are not about c_i in order to express our thoughts. It is also evident that *good* discriminators for the term categorization task (i.e. documents that contain most terms belonging to c_i and contain only few terms not belonging to c_i) will be extremely rare. Similar arguments can be made for the "term co-occurrence representation" we have used in the experiments of Section 5.4.8.

In order to have a more direct proof of this fact we have computed, for our three extensional representations, the absolute values of the information gain function (globalized by means of the f_{max} method, as usual) for the features that score highest in terms of such a function. The results, which are computed on the entire RCV1 corpus of documents, are plotted in Figure 11. From the figure we can see that, on average, a feature of the term occurrence representation (as used in text categorization) has a value an order of magnitude higher than the feature of the same rank in the document occurrence representation (as used in term categorization). The term co-occurrence representation of Section 5.4.8 gives improved results with respect to the document occurrence representation, thus confirming the impression that it might be a better representation than document occurrence for term categorization applications. However, its results are of the same order of magnitude of the document occurrence representation.

Since information gain is a direct measure of the discriminative power of a feature,

this comparison (although, as remarked above, not entirely "fair" because of the difference between the two category sets) is significant enough to indicate how much harder than text categorization term categorization is[20].

6. RELATED WORK

6.1 Automated generation of lexical resources

The automated generation of lexicons from text corpora has a long history, dating back at the very least to the seminal works of Lesk [Lesk 1969], Salton [Salton 1971] and Spärck Jones [Spärck Jones 1971], and has been the subject of active research throughout the last 30 years, both within the IR community [Crouch and Yang 1992; Jing and Croft 1994; Qiu and Frei 1993; Ruge 1992; Schütze and Pedersen 1997] and the NLP community [Grefenstette 1994; Hirschman et al. 1988; Riloff and Shepherd 1999; Roark and Charniak 1998; Tokunaga et al. 1995]. Most of the lexicons built by these works come in the form of *cluster-based thesauri*, i.e. graphs in which nodes are groups of synonymous or quasi-synonymous words, and edges connecting the nodes represent semantic contiguity. Most of these approaches follow the basic pattern of (i) measuring the degree of pairwise similarity between the words extracted from a corpus of texts, and (ii) clustering these words based on the computed similarity values.

Step (i) requires terms to be given explicit (i.e. vectorial) representations, so that similarity between them can be computed. For this, the two term representations we have tested in the experiments of Section 5.4.8, one in which the documents which contain the terms are the features [Crouch 1990; Crouch and Yang 1992; Qiu and Frei 1993; Schäuble and Knaus 1992; Sheridan and Ballerini 1996; Sheridan et al. 1997] and the other in which the co-occurring terms are the features [Schütze 1992; Schütze and Pedersen 1997], can be alternatively chosen. The first representation is based on "first-order co-occurrence" (i.e. two terms are considered similar when they frequently co-occur with each other), while the latter is based on "second-order co-occurrence" (i.e. two terms are considered similar when they frequently co-occur with the same terms). Variants of this latter approach are obtained by restricting the context of co-occurrence from the document to the paragraph, or to the sentence, or to a sliding, fixed-size window of text centered around the focus term [Lund and Burgess 1996]. Other authors instead reinterpret the notion of "co-occurrence" as meaning something different from the mere simultaneous presence of the two terms in the same text window. For instance, [Lin 1998; Pantel and Lin 2003] represent term t_j by vectors of *pairs* (t_k, r_k), where t_k is a term that co-occurs with t_j in some sentence and r_k is the grammatical relationship between t_j and t_k in this sentence; in this way, syntactic knowledge is brought to bear in what is otherwise an essentially knowledge-free approach.

When the lexical resources being built are domain-specific, whether a word belongs or not to the domain of interest is usually established by checking whether its frequency within documents belonging to the domain is higher than its frequency

[20]Note also that the performance of our experiments may also have been negatively influenced by the imperfect quality of the WordNetDomains resource, which was generated by a combination of automatic and manual procedures and did not undergo extensive checking afterwards [Magnini and Cavaglià 2000].

within generic documents [Chen et al. 1996; Riloff and Shepherd 1999; Schatz et al. 1996] (this property is often characterized as *salience* [Yarowsky 1992]). This literature has thus taken an approach which can be summarized in the recipe "from a set of documents about domain c_i and a set of generic documents (i.e. mostly not about c_i), extract the words that mostly characterize c_i (and organize them into a thesaurus)". Our work is different, in that its underlying "supervised" approach requires a starting kernel of terms about c_i, but does not require that the documents from which the terms are extracted be labeled according to the domains under consideration. This makes our technique particularly suitable for *extending* a previously existing domain-specific lexical resource, while the previously known unsupervised techniques tend to be more useful for generating one from scratch. This suggests an interesting methodology of (i) generating a domain-specific lexical resource by some unsupervised technique, and then (ii) extending it by our supervised technique.

As anyone involved in applications of supervised machine learning knows, labeled resources are often a bottleneck for learning algorithms, since labeling items by hand is expensive. Concerning this, note that our technique is advantageous, since it requires an initial set of labeled terms *only in the first iteration of the bootstrapping process*, i.e. for generating L_1 from L_0. Once a lexical resource has been extended with new terms, extending it further only requires a new *unlabeled* corpus of documents, but no other labeled resource. This is different from the techniques described above, which require, for extending a lexical resource that has just been built by means of them, a new *labeled* corpus of documents.

One advantage of our approach is that it solidly rests on the strong theoretical bases of indexing theory and supervised learning theory. It is also a "minimalist" approach, in the sense that nothing else than an indexing tool and a supervised learning tool are needed for it; indexing tools and supervised learning tools different from the ones used here can be plugged in and out straightforwardly. The approaches adopted in [Riloff and Shepherd 1999; Roark and Charniak 1998; Thelen and Riloff 2002], which have been discussed more in detail in Section 5.5, are also supervised (in the sense that they require the presence of training terms), but rest on heuristic pattern extraction techniques and scoring functions, rather than on general supervised learning techniques. Also, the approaches presented in the quoted works have been tested only on texts which are at least loosely connected with several among the domains of interest; it remains to be seen how their techniques would work on corpora unrelated to the domains.

6.2 Boosting

Boosting has been applied to several learning tasks related to text analysis, including POS-tagging and PP-attachment [Abney et al. 1999], clause splitting [Carreras and Màrquez 2001], word segmentation [Shinnou 2001], term extraction [Vivaldi et al. 2001], named entity extraction [Carreras et al. 2003; Wu et al. 2002], word sense disambiguation [Escudero et al. 2000], text categorization [Liu et al. 2002; Nardiello et al. 2003; Schapire and Singer 2000; Schapire et al. 1998; Sebastiani et al. 2000; Taira and Haruno 2001], document routing [Iyer et al. 2000; Kim et al. 2000], and e-mail filtering [Carreras and Màrquez 2001]. Among these works, the one somehow closest in spirit to ours is [Vivaldi et al. 2001], since it is concerned

with extracting medical terms from a corpus of texts. A key difference with our work is that the features by which candidate terms are represented in [Vivaldi et al. 2001] are not the documents they occur in, but the results of term extraction algorithms; therefore, our approach is simpler and more general, since it does not require the existence of separate term extraction algorithms.

7. CONCLUSION AND DISCUSSION

We have reported an approach to the automatic expansion of domain-specific lexicons by the combination of (i) a dual interpretation of IR-style text indexing theory and (ii) a supervised learning approach; this approach allows the extraction of domain-specific terms from domain-generic texts. The advantages of our method are that it is particularly suited (i) to the situation in which several domain-specific lexicons need to be extended in parallel and (ii) to the situation in which no labeled text corpora are available, and that it does not require pre-existing semantic knowledge.

We have exemplified our approach by running experiments in which we simultaneously expand a large number of domain-specific lexicons (up to 145) by extracting new terms from a large number of domain-generic texts (up to 806,812). These experiments have been run by using widely available resources (the WordNetDomains set of domain-specific lexicons and the RCV1 corpus) and standard evaluation measures (precision and recall); this means that our results constitute a useful experimental setting and a baseline for other researchers to improve upon. We also hope that this study will encourage researchers to adopt, for the lexicon expansion task, the more "objective" evaluation methodology exemplified here.

Our experiments suggest that our approach is viable, although large margins of improvement still exist: F_1 values are still low, at least if compared to the F_1 values that have been obtained in text categorization research on the same corpus, so work is still needed in tuning this approach in order to obtain significant categorization performance. One obvious direction is to combine all the parameter settings that we have found to work best in individual experiments exploring individual facets of the problem (e.g. using a small number of texts, a fine-grained set of domains, a high number of boosting iterations, etc., at the same time); but many other directions exist. Even so, we have provided a theoretical argument which explains why we cannot hope to obtain, for this "term categorization" task, effectiveness values comparable to the ones which are routinely obtained in text categorization experiments.

For the moment being we have run experiments consisting of one iteration only of the bootstrapping process. In future experiments we also plan to allow for multiple iterations, in which the system learns new terms also from previously learnt ones (similarly to the approach adopted in [Riloff and Shepherd 1999; Thelen and Riloff 2002]). This is quite reasonable since our system displays a high precision (usually above than .70), which tends to guarantee that it is safe for our system to learn from terms it has itself learnt in the previous iterations. The very fact that our system has proven to perform best when fed with small quantities of text (e.g. a day's worth of newswire stories) thus suggests a simple strategy of performing many iterations of the process, each time using a small quantity of text.

ACKNOWLEDGMENTS

This work has been carried out in the framework of the WebFAQ project, funded by the Provincia Autonoma di Trento. We thank Pio Nardiello for his assistance in tuning the AdaBoost.MH^{KR} code, Thorsten Joachims for making the SVMlight package available, and Alessandro Sperduti for valuable comments.

REFERENCES

Abney, S., Schapire, R. E., and Singer, Y. 1999. Boosting applied to tagging and PP attachment. In *Proceedings of EMNLP-99, 4th Conference on Empirical Methods in Natural Language Processing*. College Park, MD, 38–45.

Ault, T. and Yang, Y. 2001. kNN, Rocchio and metrics for information filtering at TREC-10. In *Proceedings of TREC-10, 10th Text Retrieval Conference*, E. M. Voorhees, Ed. National Institute of Standards and Technology, Gaithersburg, US, Gaithersburg, US, 84–93.

Carreras, X. and Màrquez, L. 2001. Boosting trees for anti-spam email filtering. In *Proceedings of RANLP-01, 4th International Conference on Recent Advances in Natural Language Processing*. Tzigov Chark, BG.

Carreras, X. and Màrquez, L. 2001. Boosting trees for clause splitting. In *Proceedings of CONLL-01, 5th Conference on Computational Natural Language Learning*, W. Daelemans and R. Zajac, Eds. Toulouse, FR, 73–75.

Carreras, X., Màrquez, L., and Padró, L. 2003. A simple named entity extractor using AdaBoost. In *Proceedings of CoNLL-03, 7th Conference on Natural Language Learning*, W. Daelemans and M. Osborne, Eds. Edmonton, CA, 152–155.

Chen, H., Schuffels, C., and Orwing, R. 1996. Internet categorization and search: A machine learning approach. *Journal of Visual Communication and Image Representation, Special Issue on Digital Libraries 7,* 1, 88–102.

Crouch, C. J. 1990. An approach to the automatic construction of global thesauri. *Information Processing and Management 26,* 5, 629–640.

Crouch, C. J. and Yang, B. 1992. Experiments in automated statistical thesaurus construction. In *Proceedings of SIGIR-92, 15th ACM International Conference on Research and Development in Information Retrieval*. Kobenhavn, DK, 77–87.

Dagan, I. 2000. Contextual word similarity. In *Handbook of Natural Language Processing*, R. Dale, H. Moisl, and H. Somers, Eds. Marcel Dekker Inc, New York, NY, Chapter 19, 459–476.

Dagan, I., Marcus, S., and Markovitch, S. 1995. Contextual word similarity and estimation from sparse data. *Computer Speech and Language 9,* 2, 123–152.

Dumais, S. T. and Chen, H. 2000. Hierarchical classification of Web content. In *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*, N. J. Belkin, P. Ingwersen, and M.-K. Leong, Eds. ACM Press, New York, US, Athens, GR, 256–263.

Escudero, G., Màrquez, L., and Rigau, G. 2000. Boosting applied to word sense disambiguation. In *Proceedings of ECML-00, 11th European Conference on Machine Learning*. Springer Verlag, Heidelberg, DE, Barcelona, ES, 129–141. Published in the "Lecture Notes in Computer Science" series, number 1810.

Fellbaum, C., Ed. 1998. *WordNet: An Electronic Lexical Database*. The MIT Press, Cambridge, US.

Gale, W., Church, K., and Yarowsky, D. 1993. A method for disambiguating word senses in a large corpus. *Computers and the Humanities 26,* 5/6, 415–439.

Grefenstette, G. 1994. *Explorations in automatic thesaurus discovery*. Kluwer Academic Publishers, Dordrecht, NL.

Hirschman, L., Grishman, R., and Sager, N. 1988. Grammatically-based automatic word class formation. *Information Processing and Management 11,* 1/2, 39–57.

Iyer, R. D., Lewis, D. D., Schapire, R. E., Singer, Y., and Singhal, A. 2000. Boosting for document routing. In *Proceedings of CIKM-00, 9th ACM International Conference on*

Information and Knowledge Management, A. Agah, J. Callan, and E. Rundensteiner, Eds. ACM Press, New York, US, McLean, US, 70–77.

Jing, Y. and Croft, W. B. 1994. An association thesaurus for information retrieval. In *Proceedings of RIAO-94, 4th International Conference "Recherche d'Information Assistee par Ordinateur"*. New York, US, 146–160.

Joachims, T. 1999. Making large-scale SVM learning practical. In *Advances in Kernel Methods – Support Vector Learning*, B. Schölkopf, C. J. Burges, and A. J. Smola, Eds. The MIT Press, Cambridge, US, Chapter 11, 169–184.

Joachims, T. 2001. A statistical learning model of text classification with support vector machines. In *Proceedings of SIGIR-01, 24th ACM International Conference on Research and Development in Information Retrieval*, W. B. Croft, D. J. Harper, D. H. Kraft, and J. Zobel, Eds. ACM Press, New York, US, New Orleans, US, 128–136.

John, G. H., Kohavi, R., and Pfleger, K. 1994. Irrelevant features and the subset selection problem. In *Proceedings of ICML-94, 11th International Conference on Machine Learning*. New Brunswick, US, 121–129.

Kim, Y.-H., Hahn, S.-Y., and Zhang, B.-T. 2000. Text filtering by boosting naive Bayes classifiers. In *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*, N. J. Belkin, P. Ingwersen, and M.-K. Leong, Eds. ACM Press, New York, US, Athens, GR, 168–75.

Lesk, M. E. 1969. Word-word association in document retrieval systems. *American Documentation 20,* 1, 27–38.

Lewis, D. D., Li, F., Rose, T., and Yang, Y. 2003. Reuters Corpus Volume I as a text categorization test collection. *Journal of Machine Learning Research*. Forthcoming.

Lin, D. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of ACL-98, 36th Annual Meeting of the Association for Computational Linguistics*. Montreal, CA, 768–774.

Liu, Y., Yang, Y., and Carbonell, J. 2002. Boosting to correct the inductive bias for text classification. In *Proceedings of CIKM-02, 11th ACM International Conference on Information and Knowledge Management*. ACM Press, New York, US, McLean, US, 348 – 355.

Lund, K. and Burgess, C. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instrumentation, and Computers 28,* 2, 203–208.

Magnini, B. and Cavaglià, G. 2000. Integrating subject field codes into WordNet. In *Proceedings of LREC-2000, 2nd International Conference on Language Resources and Evaluation*. Athens, GR, 1413–1418.

Magnini, B., Strapparava, C., Pezzulo, G., and Gliozzo, A. 2002. The role of domain information in word sense disambiguation. *Natural Language Engineering 8,* 4, 359–373.

Nardiello, P., Sebastiani, F., and Sperduti, A. 2003. Discretizing continuous attributes in AdaBoost for text categorization. In *Proceedings of ECIR-03, 25th European Conference on Information Retrieval*, F. Sebastiani, Ed. Springer Verlag, Pisa, IT, 320–334.

Ng, H. T., Goh, W. B., and Low, K. L. 1997. Feature selection, perceptron learning, and a usability case study for text categorization. In *Proceedings of SIGIR-97, 20th ACM International Conference on Research and Development in Information Retrieval*, N. J. Belkin, A. D. Narasimhalu, and P. Willett, Eds. ACM Press, New York, US, Philadelphia, US, 67–73.

Pantel, P. and Lin, D. 2003. Automatically discovering word senses. In *Proceedings of HLT-03, 3rd International Conference on Human Language Technology*. Edmonton, CA, 21–22.

Qiu, Y. and Frei, H.-P. 1993. Concept-based query expansion. In *Proceedings of SIGIR-93, 16th ACM International Conference on Research and Development in Information Retrieval*. Pittsburgh, US, 160–169.

Riloff, E. and Shepherd, J. 1999. A corpus-based bootstrapping algorithm for semi-automated semantic lexicon construction. *Journal of Natural Language Engineering 5,* 2, 147–156.

Roark, B. and Charniak, E. 1998. Noun phrase co-occurrence statistics for semi-automatic semantic lexicon construction. In *Proceedings of ACL-98, 36th Annual Meeting of the Association for Computational Linguistics*. Montreal, CA, 1110–1116.

Rose, T., Stevenson, M., and Whitehead, M. 2002. The Reuters Corpus Volume 1 – from yesterday's news to tomorrow's language resources. In *Proceedings of LREC-02, 3rd International Conference on Language Resources and Evaluation*. Las Palmas, ES, 827–832.

Ruge, G. 1992. Experiments on linguistically-based terms associations. *Information Processing and Management 28,* 3, 317–332.

Sahlgren, M. 2004. Random indexing of words in narrow context windows for vector-based semantic analysis. In *Acquisition and Representation of Word Meaning: Theoretical and Computational Perspectives*, A. Lenci, S. Montemagni, and V. Pirrelli, Eds. Istituti Editoriali e Poligrafici Internazionali, Pisa, IT.

Salton, G. 1971. Experiments in automatic thesaurus construction for information retrieval. In *Proceedings of the IFIP Congress*. Vol. TA-2. Ljubljana, YU, 43–49.

Schapire, R. E. and Singer, Y. 1999. Improved boosting algorithms using confidence-rated predictions. *Machine Learning 37,* 3, 297–336.

Schapire, R. E. and Singer, Y. 2000. BoosTexter: a boosting-based system for text categorization. *Machine Learning 39,* 2/3, 135–168.

Schapire, R. E., Singer, Y., and Singhal, A. 1998. Boosting and Rocchio applied to text filtering. In *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval*, W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson, and J. Zobel, Eds. ACM Press, New York, US, Melbourne, AU, 215–223.

Schatz, B. R., Johnson, E. H., Cochrane, P. A., and Chen, H. 1996. Interactive term suggestion for users of digital libraries: Using subject thesauri and co-occurrence lists for information retrieval. In *Proceedings of DL-96, 1st ACM Digital Library Conference*. Bethesda, US, 126–133.

Schäuble, P. and Knaus, D. 1992. The various roles of information structures. In *Proceedings of the 16th Annual Conference of the Gesellschaft für Klassifikation*, O. Opitz, B. Lausen, and R. Klar, Eds. Dortmund, DE, 282–290. Published by Springer Verlag, Heidelberg, DE, 1993.

Schmid, H. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*. Manchester, UK, 44–49.

Schütze, H. 1992. Dimensions of meaning. In *Proceedings of Supercomputing'92*. Minneapolis, US, 787–796.

Schütze, H. and Pedersen, J. O. 1997. A cooccurrence-based thesaurus and two applications to information retrieval. *Information Processing and Management 33,* 3, 307–318.

Sebastiani, F. 2002. Machine learning in automated text categorization. *ACM Computing Surveys 34,* 1, 1–47.

Sebastiani, F., Sperduti, A., and Valdambrini, N. 2000. An improved boosting algorithm and its application to automated text categorization. In *Proceedings of CIKM-00, 9th ACM International Conference on Information and Knowledge Management*, A. Agah, J. Callan, and E. Rundensteiner, Eds. ACM Press, New York, US, McLean, US, 78–85.

Sheridan, P. and Ballerini, J.-P. 1996. Experiments in multilingual information retrieval using the SPIDER system. In *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*. Zürich, CH, 58–65.

Sheridan, P., Braschler, M., and Schäuble, P. 1997. Cross-language information retrieval in a multi-lingual legal domain. In *Proceedings of ECDL-97, 1st European Conference on Research and Advanced Technology for Digital Libraries*, C. Peters and C. Thanos, Eds. Pisa, IT, 253–268. Lecture Notes in Computer Science, number 1324, Springer Verlag, Heidelberg, DE.

Shinnou, H. 2001. Detection of errors in training data by using a decision list and AdaBoost. In *Proceedings of the IJCAI-01 Workshop on Text Learning: Beyond Supervision*. Seattle, US.

Smadja, F. 1993. Retrieving collocations from text: Xtract. *Computational Linguistics 19,* 1, 143–178.

Spärck Jones, K. 1971. *Automatic keyword classification for information retrieval*. Butterworths, London, UK.

Taira, H. and Haruno, M. 2001. Text categorization using transductive boosting. In *Proceedings of ECML-01, 12th European Conference on Machine Learning*, L. D. Raedt and P. A. Flach,

Eds. Springer Verlag, Heidelberg, DE, Freiburg, DE, 454–465. Published in the "Lecture Notes in Computer Science" series, number 2167.

Thelen, M. and Riloff, E. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of EMNLP-02, 7th Conference on Empirical Methods in Natural Language Processing*. Philadelphia, US.

Tokunaga, T., Iwayama, M., and Tanaka, H. 1995. Automatic thesaurus construction based on grammatical relations. In *Proceedings of IJCAI-95, 14th International Joint Conference on Artificial Intelligence*. Montreal, CA, 1308–1313.

Vivaldi, J., Màrquez, L., and Rodríguez, H. 2001. Improving term extraction by system combination using boosting. In *Proceedings of ECML-01, 12th European Conference on Machine Learning*. Freiburg, DE, 515–526.

Wu, D., Ngai, G., Carpuat, M., Larsen, J., and Yang, Y. 2002. Boosting for named entity recognition. In *Proceedings of CoNLL-02, 6th Conference on Natural Language Learning*. Taipei, TW, 195–198.

Yang, Y. and Pedersen, J. O. 1997. A comparative study on feature selection in text categorization. In *Proceedings of ICML-97, 14th International Conference on Machine Learning*, D. H. Fisher, Ed. Morgan Kaufmann Publishers, San Francisco, US, Nashville, US, 412–420.

Yarowsky, D. 1992. Word-sense disambiguation using statistical models of Roget's categories trained on large corpora. In *Proceedings of COLING-92, 14th International Conference on Computational Linguistics*. Nantes, FR, 454–460.

Zobel, J. and Moffat, A. 1998. Exploring the similarity space. *SIGIR Forum 32,* 1, 18–34.