LETTER Communicated by Peter Tino

Feature Selection for Ordinal Text Classification

Stefano Baccianella
stefano.baccianella@isti.cnr.it
Andrea Esuli
andrea.esuli@isti.cnr.it
Fabrizio Sebastiani
fabrizio.sebastiani@isti.cnr.it
Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, 56124 Pisa, Italy

Ordinal classification (also known as ordinal regression) is a supervised learning task that consists of estimating the rating of a data item on a fixed, discrete rating scale. This problem is receiving increased attention from the sentiment analysis and opinion mining community due to the importance of automatically rating large amounts of product review data in digital form. As in other supervised learning tasks such as binary or multiclass classification, feature selection is often needed in order to improve efficiency and avoid overfitting. However, although feature selection has been extensively studied for other classification tasks, it has not for ordinal classification. In this letter, we present six novel feature selection methods that we have specifically devised for ordinal classification and test them on two data sets of product review data against three methods previously known from the literature, using two learning algorithms from the support vector regression tradition. The experimental results show that all six proposed metrics largely outperform all three baseline techniques (and are more stable than these others by an order of magnitude), on both data sets and for both learning algorithms.

1 Introduction

Text management tasks such as text search, text clustering, and text classification are usually tackled by representing the textual documents in vector form. The resulting vector spaces are always characterized by a high dimensionality (often in the range of the tens, and sometimes hundreds, of thousands dimensions), since the words (or word stems) occurring in the

This is a revised and substantially extended version of a paper appeared as Baccianella, Esuli, and Sebastiani (2010). The order in which the authors are listed is purely alphabetical; each author has made an equally important contribution to this work.

Neural Computation **26, 557–591** (2014)
doi:10.1162/NECO_a_00558

documents are normally used as features and since several tens of thousands of them occur in any reasonably sized document space. This very high dimensionality is not terribly problematic in text search, where the basic operation (computing the distance between two vectors in the vector space) can be performed quickly thanks to the sparse nature of the two vectors (one of them is actually generated from a query, which makes it even sparser). It is instead problematic in other tasks involving supervised or unsupervised learning, such as text classification or clustering.

For instance, many supervised learning methods often used for text classification, such as neural networks, do not scale well to large numbers of features, and even the learning algorithms that do scale well have a computational cost at least linear in the dimensionality of the vector space. While this has a negative impact on efficiency, effectiveness suffers too, since if the ratio of the number of training examples to the number of features is low, overfitting occurs. For all of these reasons, in text managements tasks involving learning, the high dimensionality of the vector space may be problematic.

Several techniques for reducing the dimensionality of a vector space for text learning tasks have been investigated, the main one being feature selection (FS; see e.g., Guyon & Elisseeff, 2003; Liu & Motoda, 2007). This consists of identifying a subset $S \subset T$ of the original feature set T (which coincides with the set of terms that occur in at least one training document) such that $|S| \ll |T|$ ($\xi = |S|/|T|$ being called the *reduction level*) and such that S reaches the best compromise between (1) the efficiency of the learning process and of the classifiers (which is, of course, inversely proportional to $|S|$) and (2) the effectiveness of the resulting classifiers.

While feature selection mechanisms have been extensively investigated for standard binary or multiclass text classification (see Forman, 2003, 2007; Yang & Pedersen, 1997) and, to a lesser extent, for text clustering (see Dash, Choi, Scheuermann, & Liu, 2002; Dy, 2007; Liu, Liu, Chen, & Ma, 2003), they have not for a related and important text learning task, ordinal text classification. Ordinal classification (OC; also known as ordinal regression) consists of mapping each object $x_i \in X$ into exactly one of an ordered sequence (which we call the rating scale) $C = \langle c_1 \prec \cdots \prec c_n \rangle$ of classes (or ratings). This problem is intermediate between standard multiclass classification, in which C is instead an unordered set, and metric regression, in which C is instead a continuous, totally ordered set (typically the set $\mathbb{R}$ of the reals). A key feature of ordinal regression is also that the distances between consecutive classes may be different from each other.

OC is of key importance in the social sciences, since ordinal (i.e., discrete) scales are often used to elicit human judgments and evaluations (of either a quantitative or a qualitative nature) from respondents or interviewees (Miller & Salkind, 2002). An example of such a task is customer relationships management, where customers may be asked to evaluate a product or service on a scale consisting of the values Disastrous, Poor, Fair, Good,

Excellent; the case of product reviews evaluated on a scale of one to five stars (see Goldberg & Zhu, 2006; Pang & Lee, 2005; Shimada & Endo, 2008) falls under this category. Another example is opinion polling, where respondents may be asked to rate their degree of agreement with a given statement on a scale consisting of Strongly Disagree, Disagree, Agree, and Strongly Agree.[1]

In this letter, we address the problem of feature selection for OC. The reason there is a need for FS methods explicitly tailored to OC is that the analogous methods originally developed for standard multiclass classification are suboptimal for OC. In order to see this, recall that FS methods for multiclass classification reward the features that are highly correlated with a specific class c. For instance, a feature t_1 that occurs in all training documents belonging to c and never occurs in any training document not belonging to c is considered by these methods a perfect feature, since the fact that t_1 occurs in a yet unlabeled document d will contribute strong evidence toward the fact that $d \in c$, and the fact that t_1 does not occur in d will contribute strong evidence toward the fact that $d \notin c$. A less perfect but still useful feature, t_2, is one such that, say, 80% of the documents in which it occurs belong to a specific class c, while the other 20% do not belong to c. In this case, the evidence contributed by this feature is less strong than in the preceding case but very valuable anyway. However, if our classes are ordered (say, Disastrous, Poor, Fair, Good, Excellent), an FS function for multiclass classification would consider equivalent (1) a feature t_3 that occurs 80% of the time in Excellent and 20% of the time in Good and (2) a feature t_4 that occurs 80% of the time in Excellent and 20% of the time in Disastrous. This is unsuitable. In fact, t_3 is much more valuable than t_4 in an OC context, since it tends to characterize a set of contiguous classes (i.e., Good and Excellent), while t_4 does not. That is, given a document d whose true label is Excellent and that contains both t_3 and t_4, t_3 might possibly mislead the classifier in labeling d as Good (not a very serious mistake), while t_4 might possibly mislead the classifier in labeling d as Disastrous (a much more serious mistake). In sum, FS methods developed for standard multiclass classification are suboptimal for OC since they do not pay attention to the distances between different classes. There is thus a need for FS methods explicitly tailored to OC, that is, that pay attention to the fact that different misclassification may bring about different costs.

We present six novel feature selection functions that we have specifically devised for ordinal classification and thoroughly test them on two data sets of product review data by using two SVM-based algorithms for ordinal regression. In these experiments, we use as baselines the only three FS techniques for OC known from the literature. The experimental results show

[1] In the social sciences, discrete rating scales for expressing (dis)agreement with a given statement are known as Likert scales (Likert, 1932).

that our proposed methods largely outperform all three baseline techniques on both data sets and for both learning algorithms.[2]

The letter is organized as follows. In section 2 we discuss related work, and in section 3 we present our six FS algorithms for OC. Section 4 reports the results of experiments we have conducted on these methods. Section 5 discusses issues of computational cost, and section 6 concludes.

2 Related Work

There are three main classes of strategies for FS (see John, Kohavi, & Pfleger, 1994; Saeys, Inza, & Larrañaga, 2007), and they differ in terms of how (and if) the feature selection phase interacts with the classifier learning phase.

The filter approach is a greedy approach in which a real-valued function σ is applied to each feature in T in order to compute its expected contribution to solving the classification task. Only the $x = |S|$ features with the highest value for σ are retained (x may be a predetermined number or may, more typically, be expressed as a predetermined percentage of $|T|$). The filter approach has all the pros and cons of greedy approaches. It may deliver suboptimal accuracy, since each feature is evaluated in isolation of other features (which means that only uninformative features are discarded, while redundant features are not), and since the chosen learning algorithm is not involved in the evaluation. However, the filter approach is efficient, since its computational cost is just $O(|T|(|Tr| + \log |T|))$.

The wrapper approach consists of repeatedly training and evaluating the chosen classifier using (in principle) all possible subsets of the original set of features; in practice, heuristics can be used in order to restrict consideration to a limited set of promising subsets. This approach is theoretically optimal, since it tests feature subsets directly with the chosen learning algorithm and since it evaluates feature subsets in their entirety. However, it suffers from computational problems due to its combinatorial nature, and is thus used in practice only when the dimensionality of the original feature space is small. It is thus completely inadequate for text learning tasks, in which the dimensionality of the original feature space is typically $O(10^5)$ or more.[3] The same happens for bioinformatics applications such as gene selection for patient classification (Guyon, Weston, Barnhill, & Vapnik, 2002).

Finally, the embedded approach consists of performing FS directly during the learning phase, without an explicit feature selection phase to be performed before learning begins. For instance, decision tree learning methods may be seen as embedding FS in the learning phase, since the

[2] The source code of all the techniques discussed in this letter (both the ones proposed here and the ones from the literature which we use as baselines) can be downloaded at http://hlt.isti.cnr.it/FS4OR/.

[3] Interestingly, the literature on FS for metric regression seems to have mostly, if not only, investigated wrapper approaches (Miller, 2002).

classifier hinges on a small subset of features selected during the learning phase, with the consequence that all other features can be discarded. The problem with the embedded approach is that it is not general, since only some learners perform, by construction, FS as a side effect of learning (example learning algorithms for ordinal classification that can indeed do this are the OMLVQ and OGMLVQ algorithms of Fouad & Tino, 2012). In conclusion, since the wrapper approach is computationally too expensive for ordinal text classification and since we are interested in methods that can be used in connection with any supervised learning algorithm (which rules out the embedded approach), in this letter, we consider only FS solutions for the filter approach.

2.1 Feature Selection for Ordinal Classification. To the best of our knowledge, only three papers (Shimada & Endo, 2008; Mukras, Wiratunga, Lothian, Chakraborti, & Harper, 2007; Baccianella, Esuli, & Sebastiani, 2009b) address feature selection for ordinal regression within the filter approach.

Mukras et al. (2007) propose an algorithm, probability redistribution procedure (PRP), that takes as input the distribution of the feature t_k across the classes (as deriving from the distribution across the classes of the training examples containing t_k and modifies it according to the notion that when t_k occurs in (a document belonging to) a class c_j, it is "also taken to occur," to a degree linearly decaying with the distance from c_j, in the classes close to c_j. The modified distribution is then used in selecting features through a standard application, as in binary classification, of the information gain function.

Shimada and Endo (2008) describe a method called minimum variance (*Var*). The basic idea underlying *Var* is that of measuring the variance of the distribution of the training documents containing feature t_k across the classes and retaining only the features that correspond to the smallest variance. The intuition behind *Var* is that a feature is useful iff it is capable of discriminating a small, contiguous portion of the rating scale from the rest, and that features with a small variance are those that tend to satisfy this property.

Baccianella et al. (2009b) propose an improvement on *Var*, round robin on minimum variance (*RR*(*Var*)). *RR*(*Var*) is based on a principle according to which "each class takes an equal share of features," choosing in turn from a common pool. In other words, *RR*(*Var*) is based on the observation (originally proposed in Forman, 2004, for binary text classification and for functions other than *Var*) that *Var* might select many features that all characterize well a specific class c', while selecting few or no features that characterize well another class c''. In order to solve this problem, in *RR*(*Var*) one (1) provisionally assigns each feature t_k to the class $c(t_k)$ closest to the mean of the distribution of the training documents containing t_k

(the class that t_k presumably best characterizes), (2) for each class, orders the features assigned to it, and then (3) allows the n classes to take turns (Forman, 2004, calls this a round robin—RR—policy) in picking features from the top-most elements of their class-specific orderings. In this way, all classes are represented by the same number of features in the final feature set.

Finally, Baccianella, Esuli, and Sebastiani (2013) discuss token-based variants of the feature selection functions presented in Baccianella, Esuli, and Sebastiani (2010). No new function is introduced in Baccianella et al. (2013), and ordinal regression is used only as a case study for assessing whether token-based versions of generic feature selection functions deliver better performance than the equivalent document-based versions.

2.2 Feature Selection for Monotonic Classification. A related strand in the literature is that of feature selection for monotonic classification. Monotonic classification (see e.g., Potharst & Feelders, 2002, section 2, for a particularly clear description) is a specialization of ordinal classification in which the features t_k and the target function Φ must jointly satisfy a monotonicity constraint, that is, if $w_{k1} \leq w_{k2}$ for each feature $t_k \in T$ (where w_{ki} denotes the value of feature t_k in document d_i), then it must be the case that $\Phi(d_1) \preceq \Phi(d_2)$—that the class to which d_1 belongs is equal to or precedes in the ranking the one to which d_2 belongs. To the best of our knowledge, the only known feature selection method for monotonic classification is the one proposed in Hu et al. (2012).

Such methods are not applicable to ordinal classification in general, since in standard ordinal classification, the rank of the document is not necessarily a monotonically increasing function of the value of the features. Indeed, this assumption would be evidently inconsistent with the fact that in ordinal text classification, some textual features may well characterize the midranks (such as, e.g., the term lukewarm in a set of product reviews).

2.3 Feature Selection for Ranking. Yet another related strand in the literature is that of feature selection for ranking. Ranking is similar to ordinal classification, since objects must be assigned a score of merit and then ranked in descending order of the assigned score. Example feature selection methods for ranking are the ones proposed in Geng, Liu, Qin, and Li (2007) and Kamishima and Akaho (2006).

These methods are not applicable either to ordinal classification since the final goal is just ranking a set of objects, and not assigning them to one of a predefined set of classes; that is, the assigned scores may typically be any real number. As a consequence, both the learning functions used and the evaluation measures used are different from those employed in ordinal regression.

3 Feature Selection for Ordinal Classification

Let us fix some notation and terminology. Let $C = \langle c_1 \prec \cdots \prec c_n \rangle$ be an ordered sequence of classes, or rating scale. Our task is to estimate (from a training set Tr) a target function $\Phi : D \rightarrow C$ that assigns each document $d_i \in D$ to exactly one class in C. The result of the estimation is a classifier $\hat{\Phi} : D \rightarrow C$, whose accuracy we will evaluate on a test set Te.[4]

Given any $j \in \{1, \ldots, (n-1)\}$, $\underline{C}_j = \{c_1, \ldots, c_j\}$ will be called a prefix of C, and $\overline{C}_j = \{c_{j+1}, \ldots, c_n\}$ will be called a suffix of C. Given a feature t_k, by Tr_k we denote the set $\{d_i \in Tr \mid t_k \in d_i\}$ of the training documents that contain feature t_k, while by T, we denote the set of features that occur in at least one document in Tr.

All of our feature selection methods will include a scoring phase in which each feature $t_k \in T$ is evaluated by means of a function σ that measures the predicted utility of t_k for the classification process (the higher the value of σ, the higher the predicted utility). Given a predetermined reduction level ξ, $|S| = \xi \cdot |T|$ features will be selected based on their σ score. For some methods, this will simply involve sorting the features in decreasing order of their σ score and picking the $|S|$ top-ranked ones. For some other methods, a class-specific σ_j value will be computed for each class c_j, a class-specific ranking of the features based on their σ_j scores will be generated, and the round-robin policy mentioned in section 2 will be applied to the class-specific rankings.

For convenience, we will often express reduction levels as percentages (e.g., writing 1% in place of 0.01).

3.1 The *Var* ∗ *IDF* Method. *Var* ∗ *IDF* is a variant of the *Var* method described in Shimada and Endo (2008). Recall from section 2 that *Var* is based on the intuition of retaining the features t_k that minimize the variance $Var(\Phi(Tr_k))$ of their distribution across the classes in the training set Tr, where we view $\Phi(Tr_k)$ as a random variable that associates to each $d_i \in Tr_k$ its true class $\Phi(d_i)$ and where we view the rating scale $C = \langle c_1 \prec \cdots \prec c_n \rangle$ as the sequence $\langle 1 \prec \cdots \prec n \rangle$ of the first n natural numbers.

Example 1. Let $C = \langle c_1 \prec \cdots \prec c_5 \rangle$ be a rating scale, and let Tr be a training set $\{d_1, \ldots, d_{10}\}$ labeled according to C such that feature t_k occurs only in documents d_1, d_3, d_9, and such that $\Phi(d_1) = c_4$, $\Phi(d_3) = c_5$, $\Phi(d_9) = c_1$. We ideally map $C = \langle c_1 \prec \cdots \prec c_5 \rangle$ to $C = \langle 1 \prec \cdots 5 \rangle$, so that we may assume that $\Phi(d_1) = 4$, $\Phi(d_3) = 5$, $\Phi(d_9) = 1$. The score $Var(\Phi(Tr_k))$ attributed by the *Var* method to feature t_k is then the variance of the set $\{4, 5, 1\}$, that is, $Var(\Phi(Tr_k)) = 4.33$.

[4] Consistent with most other mathematical literature, we use a caret to indicate estimation.

Note that a feature t_k that occurs only in (training documents assigned to) class c_j is such that the variance $Var(\Phi(Tr_k))$ is equal to zero. This feature seems obviously useful, since its presence in a test document d_i can be taken as an indication that d_i belongs to c_j. By the same token, if a feature t_1 occurs 90% of the time in c_j and the remaining 10% in a class contiguous to c_j, while a feature t_2 occurs 90% of the time in c_j and the remaining 10% in a class far away from c_j, then $Var(\Phi(Tr_1)) < Var(\Phi(Tr_2))$. Feature t_1 is also more useful than t_2 since the presence of t_1 in a test document d_i tends to indicate that d_i belongs to c_j or its vicinity, while t_2 gives a less clear-cut indication.

Var seems thus a reasonable measure of the usefulness of a feature t_k for the ordinal classification task. In sum, we are interested in retaining features t_k that give rise to a low $Var(\Phi(Tr_k))$ value and discarding ones that give rise to a high such value.

However, we here note that a feature t_k that occurs only once in the entire training set (e.g., in class c_j) is trivially such that $Var(\Phi(Tr_k)) = 0$, but is not useful, since the fact that it occurs exactly in c_j might be due to chance. The features that we are really interested in are those that have low variance *and* high frequency of occurrence (in the training set), since this high frequency of occurrence lends statistical robustness to the estimated value of their variance and tends to guarantee that the feature will be encountered often in the test set and will thus contribute to the classification of many test documents.

As a measure of the (in)frequency of occurrence of a feature t_k, we use the well-known inverse document frequency (IDF) (Spärck Jones, 1972), defined as

$$IDF(t_k) = \log_e \frac{|Tr|}{\#_{Tr}(t_k)},$$

where $\#_{Tr}(t_k)$ denotes the number of training documents that contain feature t_k. Intuitively, the higher $IDF(t_k)$ is, the fewer are the training documents in which t_k occurs, and the less useful t_k is going to be.

We formalize the $Var * IDF$ method by defining

$$\sigma(t_k) = -(Var(\Phi(Tr_k)) + \epsilon) * (IDF(t_k))^a \tag{3.1}$$

where

- ϵ is a small positive constant whose purpose is to prevent the first factor in the multiplication from evaluating to zero. Without it, a feature that occurs only once in the entire training set would receive a score of 0 and thus have the highest possible value of σ (since $\sigma(t_k) \in (-\infty, 0]$), which, as noted above, is undesirable.
- a is a nonnegative real-valued parameter (to be optimized on a validation set) that allows fine-tuning the relative contributions of variance

and *IDF* to the product.[5] The fact that it is nonnegative ensures that σ is a monotonically decreasing function of *IDF*, that is, features that occur in few documents tend to get a low value of σ.

The $Var * IDF$ method consists of retaining the x features with the highest σ value, as computed via equation 3.1, and discarding the others.

3.2 The *RR*(*Var* ∗ *IDF*) Method. Similar to *Var*, the $Var * IDF$ method runs the risk of exclusively serving the needs of a certain class and disregarding the others. If all the retained features mostly occur in class c_j and its neighboring classes, the resulting feature set will exclusively contain features good at discriminating c_j and its neighboring classes from the rest, while other classes might not be adequately discriminated by any of the remaining features.

Similar to the $RR(Var)$ method discussed in section 2, in order to pick the best x features, the $RR(Var * IDF)$ method thus (1) provisionally assigns each feature t_k to the class closest to the mean of its distribution across the classes; (2) sorts, for each class c_j, the features provisionally assigned to c_j in decreasing order of value of the σ function of equation 3.1; and (3) enforces a round-robin policy in which the n classes take turns picking a feature from the top-most elements of their class-specific orderings until x features are picked. In this way, for each class c_j, the final set of selected features contains at least the $\frac{x}{n}$ features that are best at discriminating c_j and its neighboring classes, which means that all the classes in the rating scale C are adequately championed in the final feature set S.[6]

[5] The intuition behind the $Var * IDF$ method is that both *Var* and *IDF* contribute to evaluating the usefulness of the feature, but this intuition does not say whether the two factors have the same importance and which of the two (if any) is more important. This is a similar situation to what happens with the well-known *TFIDF* function in information retrieval (see equations 4.3–4.5); *TFIDF* originally arose from the intuition that both *TF* (term frequency) and *IDF* (inverse document frequency) have an impact on how important a term is for representing the content of a given document, but this intuition did not say anything about their relative importance. In order to assess their relative importance, Salton and Buckley (1988) tested no fewer than 289 combinations of *TF* and *IDF*, where each combination differed in terms of subtle variations (e.g., logarithms, exponents, additive factors, normalization factors) of *TF* or *IDF* or both. The exact combination reproduced in equations 4.3–4.5 is the one that proved the best performer in Salton and Buckley (1988) and has stuck with the IR community ever since that paper was published. In the case of $Var * IDF$, rather than experimenting with 289 combinations, we have more simply decided to include an exponential weighting parameter a that allows us to fine-tune the relative contributions of *Var* and *IDF* to the product and optimize that parameter on a validation set.

[6] The "at least" here means that since the same feature may be a high-scoring feature for more than one class, strictly more than $\frac{x}{n}$ features per class may eventually get selected. Similar considerations apply to the $RR(IGOR)$ and $RR(AC * IDF)$ techniques to be discussed in sections 3.3 and 3.4, where a round-robin phase is also present.

3.3 The *RR*(*IGOR*) Method. The round robin on information gain for ordinal regression (*RR*(*IGOR*)) method is based on the idea of adapting information gain, a function routinely employed in feature selection for binary classification (see Yang & Pedersen, 1997), to ordinal regression.[7]

In a binary classification task in which we need to separate class y_j from its complement $\overline{y}_j$ we may perform feature selection by scoring each feature t_k with the function

$$IG(t_k, y_j) = H(y_j) - H(y_j|t_k) = \sum_{y \in \{y_j, \overline{y}_j\}} \sum_{t \in \{t_k, \bar{t}_k\}} P(t, y) \log_2 \frac{P(t, y)}{P(t)P(y)}. \quad (3.2)$$

In this formula, $H(\cdot)$ indicates entropy, $H(\cdot|\cdot)$ indicates conditional entropy, and probabilities are evaluated on an event space of (training) documents, with $P(t_k)$ and $P(\bar{t}_k)$ representing the fractions of documents that contain t_k and do not contain t_k, respectively, and $P(y_j)$ and $P(\overline{y}_j)$ representing the fractions of documents that have label y_j and do not have label y_j, respectively. $IG(t_k, y_j)$ measures the reduction in the entropy of y_j obtained as a result of observing t_k, that is, measures the information that t_k provides on y_j. For binary classification, the x features with the highest $IG(t_k, y_j)$ value are thus retained, while the others are discarded.

RR(*IGOR*) is based on the metaphor according to which ordinal regression on rating scale $C = \langle c_1 \prec \cdots \prec c_n \rangle$ might be viewed as the simultaneous generation of $(n-1)$ binary classifiers $\dot{\Phi}_j$ (for $j = 1, \ldots, (n-1)$), each of them in charge of deciding whether a document belongs to (one of the classes in) prefix $\underline{C}_j = \{c_1 \prec \cdots \prec c_j\}$ or to suffix $\overline{C}_j = \{c_{j+1} \prec \cdots \prec c_n\}$.[8] For each feature t_k, we thus compute $(n-1)$ different $IG(t_k, y_j)$ values by taking $y_j = c_1 \cup \cdots \cup c_j$ and $\overline{y}_j = c_{j+1} \cup \cdots \cup c_n$, for $j = 1, \ldots, (n-1)$.

Similarly, to the *RR*(*Var* $*$ *IDF*) method of section 3.2, in order to pick the best x features, we sort, for each of the $(n-1)$ binary classifiers $\ddot{\Phi}_j$ above, the features in decreasing order of their $IG(t_k, y_j)$ value, and we enforce a round-robin policy in which the $(n-1)$ classifiers $\ddot{\Phi}_j$ above take turns, for at least $\frac{x}{n-1}$ rounds, in picking a feature from the top-most elements of their classifier-specific orderings. In this way, for each classifier $\ddot{\Phi}_j$ the final

[7] Any function routinely used for feature selection in binary classification, such as chi square or odds ratio, could have been used here in place of information gain. We have chosen information gain since it has systematically been one of the top performers in comparative evaluations aiming to test the relative merits of different feature selection metrics in text classification (see Forman, 2003; Yang & Pedersen, 1997).

[8] We want to stress that this is only a metaphor. In reality, the learning phase does not necessarily involve the generation of the classifiers $\ddot{\Phi}_j$ discussed here. This should also be clear from the fact that the feature selection techniques discussed in this letter are all learner independent.

set S of selected features contains the (at least) $\frac{x}{(n-1)}$ features that are best at discriminating the rating scale prefix $\underline{C}_j$ from the rating scale suffix $\overline{C}_j$, which means that all the $(n-1)$ classifiers $\ddot{\Phi}_j$ are adequately championed in the final feature set S.

The intuition here is that if test document d_i belongs to class c_j, classifiers $\ddot{\Phi}_1, \ldots, \ddot{\Phi}_{j-1}$ will, we hope, be represented in S by features that indicate d_i belong to their corresponding rating scale suffixes $\overline{C}_1, \ldots, \overline{C}_{j-1}$, while classifiers $\ddot{\Phi}_j, \ldots, \ddot{\Phi}_{n-1}$ will, we hope, be represented in S by features that indicate d_i belong to their corresponding rating scale prefixes $\underline{C}_j, \ldots, \underline{C}_{n-1}$. The following is a concrete example where this intuition is at work.

Example 2. In the Amazon-83713 data set described in section 4.1, the term bad is the one with the highest $IG(t_k, y_i)$ value when y_i is the prefix 1 Star $\cup$ 2 Stars $\cup$ 3 Stars, since it tends to occur frequently in documents belonging to one of the classes in y_i and infrequently in documents belonging to one of the classes in $\overline{y}_i =$ 4 Stars $\cup$ 5 Stars. Conversely, the term *good* is the one with the highest $IG(t_k, y_i)$ value when $y_i =$ 1 Star $\cup$ 2 Stars, since it tends to occur infrequently in documents belonging to one of the classes in y_i and frequently in documents belonging to one of the classes in $\overline{y}_i =$ 3 Stars $\cup$ 4 Stars $\cup$ 5 Stars. (Note that $IG(t_k, y_i) = IG(t_k, \overline{y}_i)$ by definition.) Being top scorers for their respective prefixes, both terms are selected by $RR(IGOR)$. Given a text containing the sentence *This camera is good and bad at the same time*, the presence of the term *good* will provide evidence that the text is in one of 3 Stars, 4 Stars, 5 Stars, while the presence of the term *bad* will provide evidence that the text is in one of 1 Star, 2 Stars, 3 Stars. The presence of both terms thus provides evidence that the text is in 3 Stars.

3.4 The $RR(AC * IDF)$ Method. A potential problem with the methods we have proposed up to now, and with the ones mentioned in section 2, is that none of them depends on (i.e., optimizes) the specific evaluation measure chosen for evaluating ordinal regression. The $RR(AC * IDF)$ method tries to address this shortcoming by including the chosen evaluation measure as a parameter and directly optimizing it.

Assume that E is the chosen evaluation measure and that $E(\hat{\Phi}, d_i)$ represents the error that classifier $\hat{\Phi}$ makes in classifying document d_i. For example, if $\hat{\Phi}(d_i) = c_1$, $\Phi(d_i) = c_2$ and E is absolute error (see section 4.2), then $E(\hat{\Phi}, d_i) = |c_1 - c_2|$. We define the anticorrelation of a feature t_k with a class c_j in the training set Tr as

$$AC_{Tr}(t_k, c_j) = \frac{\sum_{\{d_i \in Tr \mid t_k \in d_i\}} E(\tilde{\Phi}_j, d_i)}{|\{d_i \in Tr \mid t_k \in d_i\}|}, \tag{3.3}$$

where $\tilde{\Phi}_j$ is the "trivial" classifier that assigns all documents to the same class c_j. In other words, $AC_{Tr}(t_k, c_j)$ measures how bad an indicator of membership in class c_j feature t_k is, where "bad" is defined in terms of the chosen error measure. For instance, $AC_{Tr}(t_k, c_j)$ is equal to zero (the best possible value) when all the training examples that contain t_k are assigned to c_j, since the error E that we would thus make is zero.

It would now be tempting to define $\sigma(t_k, c_j)$ as $-AC_{Tr}(t_k, c_j)$, the opposite of the anticorrelation between t_k and class c_j, since $-AC_{Tr}(t_k, R(t_k))$ measures how well t_k characterizes c_j. While this is in principle reasonable, for the same reasons as outlined in section 3.1 we need to compensate for the fact that AC does not pay attention to frequency-of-occurrence considerations; this method might thus select features whose occurrence counts are not statistically robust.

This leads us to defining the score σ of a feature t_k with respect to class c_j as

$$\sigma(t_k, c_j) = -(AC_{Tr}(t_k, c_j) + \epsilon) * (IDF(t_k))^a, \tag{3.4}$$

where the ϵ and a parameters and the IDF function serve the same purpose as in equation 3.1. Much like what happens in the $RR(Var * IDF)$ and $RR(IGOR)$ methods, in order to select the best x features, we now apply a round-robin policy in which each class c_j is allowed to pick the (at least) $\frac{x}{n}$ features with the best $\sigma(t_k, c_j)$, so that each class in the rating scale is well served by the final set of features.[9]

It is interesting to note that this method has an alternative interpretation in terms of the so-called *earth mover's distance* (EMD – also known as *Wasserstein metric* (Rubner, Tomasi, & Guibas, 2000; Rüschendorf, 2001), a well-known measure of the distance between the probability distributions of two ordinal random variables (see Levina & Bickel, 2001, for a rigorous probabilistic interpretation of the EMD). Informally, if the two distributions are interpreted as two different ways of scattering a certain amount of "earth" across different "heaps," the EMD is defined to be the minimum amount of work needed for transforming one set of heaps into the other, where the work is assumed to correspond to the sum of the amounts of earth moved times the distance by which they are moved. If the distribution of a feature t_k across the rating scale $C = \langle c_1 \prec \cdots \prec c_n \rangle$ is thought of

[9]The earlier version of this letter (Baccianella et al., 2010) presented a different version of the $RR(AC * IDF)$ method (called $RR(NC * IDF)$). It is different from $RR(AC * IDF)$ in that it included an intermediate step in which feature t_k is assigned to the class $c(t_k)$ that is least anticorrelated with it. We have now removed this step since it does not allow t_k to be picked, in the round-robin phase, for a class c_j different from $c(t_k)$, which penalizes c_j, since t_k is not necessarily picked for $c(t_k)$. As a result, $RR(AC * IDF)$ is simpler and (as the experiments have shown) more effective than than $RR(NC * IDF)$.

as an ordinal random variable, it is trivial to observe that the anticorrelation $AC_{Tr}(t_k, c_j)$ between t_k and class c_j is exactly the EMD between the distribution of t_k in Tr and the "degenerate" distribution that represents a hypothetical perfect feature in which the training documents in which t_k occurs all belong instead to c_j.

3.5 The *RR*(*RankMergeU*) and *RR*(*RankMergeW*) Methods. It is well known from research in feature selection for binary text classification (see Yang & Pedersen, 1997) that some of the best functions used within the filter approach (such as, chi square, information gain, binormal separation, and odds ratio) often tend to perform comparably while at the same time selecting fairly different sets of features. Preliminary experiments that we had run in the early phases of this work had confirmed this to be the case as well for the techniques we have discussed in the previous sections.

The two methods described in this section try to combine the strengths of our different methods by rank merging (also known as rank aggregation), a technique that is fairly popular for applications such as metasearch (Akritidis, Katsaros, & Bozanis, 2011). Let $T = \{t_1, t_2, \ldots, t_{|T|}\}$ be a set of $|T|$ objects, and let $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_{|\Sigma|}\}$ be a set of functions of type $\sigma_i : T \to \mathbb{R}$. Each $\sigma_i \in \Sigma$ induces a ranking of the objects in T such that $pos_i(t') \preceq pos_i(t'')$ iff $\sigma_i(t') \geq \sigma_i(t'')$, where $pos_i(t)$ denotes the position of object t in the ranking and $\preceq$ denotes precedence in the ranking. Rank merging consists of generating a new ranking $\sigma_{|\Sigma|+1}$ whose characteristics are (it is hoped) "better" than those of each $\sigma_i \in \Sigma$.

Many different rank-merging strategies have been proposed in the literature (see Renda & Straccia, 2003, for a review and experimental comparison). Some of them leverage only the rank positions $pos_i(t)$ that object t has in the input rankings (since in some applications, the scores $\sigma_i(t)$ may not be available to the rank-merging mechanism), while some others also use the scores $\sigma_i(t)$. The latter have been shown to bring about better performance (Renda & Straccia, 2003); as a result, since the scores $\sigma_i(t)$ are indeed available to us, we opt for merging strategies that do make use of these scores.

The input rankings σ_i that we will merge in this work are the ones generated by equations 3.1 and 3.4, as from the $RR(Var * IDF)$ and $RR(AC * IDF)$ techniques described in the previous sections. We do not use the rankings generated by equation 3.2 as from the $RR(IGOR)$ technique, since this generates not one ranking per class but one ranking for each possible separator between a prefix and a suffix of the rating scale. As such, the rankings generated by $RR(IGOR)$ are of a type different from the ones generated by $RR(Var * IDF)$ and $RR(AC * IDF)$ and cannot be merged with these latter.

The rankings we use are the class-specific ones generated before the round-robin step, and not the ones global to the entire rating scale generated

by the round-robin step. A round-robin step that combines the merged class-specific rankings will thus be needed. Note that since we use the class-specific rankings generated before the round-robin step, the ones generated by $Var * IDF$ are the same as those generated by $RR(Var * IDF)$, and will thus not be used.

As a merging policy, we test two techniques. Our first technique, called $RR(RankMergeU)$, where "U" stands for "unweighted," consists of

1. Rescaling the original scores $\sigma_i(t_k, c_j)$ via the function

$$\overline{\sigma}_i(t_k, c_j) = \frac{\sigma_i(t_k, c_j) - \min_{t_y \in T} \sigma_i(t_y, c_j)}{\max_{t_y \in T} \sigma_i(t_y, c_j) - \min_{t_y \in T} \sigma_i(t_y, c_j)} \tag{3.5}$$

   so that the normalized scores $\overline{\sigma}_i(t_k, c_j)$ deriving from the two techniques are all distributed on the [0,1] interval
2. Averaging the rescaled scores $\overline{\sigma}_i(t_k, c_j)$ obtained by a feature t_k for class c_j across our two scoring techniques, thus generating an aggregated score $\sigma(t_k, c_j)$
3. Performing the usual round-robin step in which each class c_j is allowed to pick the (at least) $\frac{x}{n}$ features with the best $\sigma(t_k, c_j)$ value, so that each class in the rating scale is well served by the final set of features

The use of equation 3.5 as a score normalization technique is discussed in Renda and Straccia (2003), show that it outperforms other more sophisticated normalization techniques (such as Z-score normalization), notwithstanding its simplicity. As Renda and Straccia (2003) showed, averaging the rescaled scores $\overline{\sigma}_i(t_k, c_j)$, as we do in step 2, has proven a better policy than simply averaging the input rank positions $pos_i(t)$.

Our second merging technique, which we call $RR(RankMergeW)$, where "W" stands for "weighted," consists of

1. Computing, on a held-out validation set, the error made by the classifier after either $RR(Var * IDF)$ or $RR(AC * IDF)$ has been used to perform feature selection
2. Running the same three steps as in the $RR(RankMergeU)$ method, but for the fact that the simple average of step 2 is replaced by a weighted average in which the technique that has performed better on the held-out validation set is given a higher weight. Specifically, we use the weighted average

$$\sigma(t_k, c_j) = \sum_i \frac{1}{E_i} \overline{\sigma}_i(t_k, c_j) \tag{3.6}$$

where i ranges on the two techniques above and E_i is a measure of the error made by the technique on the held-out validation set

Weighting the input rankings by the accuracy that can be obtained by the originating techniques is shown in Renda and Straccia (2003) to outperfom the simple unweighted average. The drawback of weighted techniques is, of course, the computational burden due to the need to obtain the weights by testing the algorithms that contribute the individual rankings on a held-out validation set. The rationale of testing both $RR(RankMergeU)$ and $RR(RankMergeW)$ is thus checking whether the additional computation time required by $RR(RankMergeW)$ is justified by superior performance.

3.6 A Note on Class Distance. As we noted in section 1, in ordinal classification, the distances between consecutive classes cannot always be taken to be equal. This may be the case, when, for example, a rating scale consists of classes Poor, Good, Very Good, and Excellent, since it might be argued that the distance between Poor and Good is higher than the distance between Very Good and Excellent.

Note that all of our FS functions, with the exception of $RR(IGOR)$, allow bringing to bear these distances in the feature selection phase. In fact, if it is not the case that distances should be considered all equal, one needs only to replace the nonnumerical classes with numerical labels whose values reflect the intended distances. For instance, one may want to replace rating scale ⟨Poor, Good, Very Good, Excellent⟩ with a numerical scale ⟨1,3,4,5⟩ so as to impose the constraint that the conceptual distance between Poor and Good should be considered higher than the conceptual distance between any other two consecutive classes. The $Var * IDF$ and $RR(Var * IDF)$ techniques are inherently sensitive to these distances because variance is. Concerning $RR(AC * IDF)$, whether it is also sensitive to these distances depends on the adopted error measure E; assuming E is (which is the case, e.g., if E is absolute error), $RR(AC * IDF)$ is also.

4 Experiments

4.1 Experimental Setting

4.1.1 The Data Sets. We have tested the proposed measures on two data sets whose characteristics are reported in Table 1.

The first is the TripAdvisor-15763 data set built by Baccianella et al. (2009b), consisting of 15,763 hotel reviews from the TripAdvisor website.[10] We use the same split between training and test documents Baccianella et al. (2009b) did, resulting in 10,508 documents used for training and 5255 for testing the training set contains 36,670 unique words. From the 10,508

[10]The data set is available for download from http://hlt.isti.cnr.it/reviewdata/.

Table 1: Main Characteristics of the Data Sets Used.

Data Set	$\|Tr\|$	$\|Te\|$	$\|T\|$	1 Star	2 Stars	3 Stars	4 Stars	5 Stars
TripAdvisor-15763	10,508	5255	36,670	3.9%	7.2%	9.4%	34.5%	45.0%
Amazon-83713	20,000	63,713	138,964	16.2%	7.9%	9.1%	23.2%	43.6%

Notes: The first three columns indicate the number of training documents, the number of test documents, and the number of unique words contained in the training documents, respectively. The last five columns indicate the fraction of documents that have a given number of "stars."

training documents, we have randomly picked 3941 documents to be used as a held-out validation set for parameter optimization.

The second data set is what we here call Amazon-83713, consisting of 83,713 product reviews from the Amazon website. Amazon-83713 is actually a small subset of the Amazon data set, consisting of more than 5 million reviews, originally built by Jindal and Liu for spam review detection purposes (Jindal & Liu, 2007), and contains all the reviews in the sections MP3, USB, GPS, Wireless 802.11, Digital Camera, and Mobile Phone.[11] We have randomly picked 20,000 documents to be used for training, and we use the remaining 63,713 documents for test; the training set contains 138,964 unique words. From the 20,000 training documents, we have randomly selected 4000 documents to be used as a held-out validation set. To the best of our knowledge, Amazon-83713 is now the largest data set ever used in the literature on ordinal text classification.

Both data sets consist of reviews scored on a scale from 1 to 5 stars: both data sets are highly imbalanced (see Table 1), with positive and very positive reviews by far outnumbering negative and very negative reviews.

4.1.2 Evaluation Measures. As our main evaluation measure we use the macroaveraged mean absolute error (MAE^M) measure proposed in Baccianella, Esuli, and Sebastiani (2009a) and defined as

$$MAE^M(\hat{\Phi}, Te) = \frac{1}{n}\sum_{j=1}^{n}\frac{1}{|Te_j|}\sum_{d_i \in Te_j}\left|\hat{\Phi}(d_i) - \Phi(d_i)\right|, \tag{4.1}$$

where Te_j denotes the set of test documents whose true class is c_j and the "M" superscript indicates macroaveraging. As Baccianella et al. (2009a) argued, the advantage of MAE^M over standard mean absolute error, defined as

$$MAE^{\mu}(\hat{\Phi}, Te) = \frac{1}{|Te|}\sum_{d_i \in Te}\left|\hat{\Phi}(d_i) - \Phi(d_i)\right|, \tag{4.2}$$

[11] The Amazon data set is available at http://www.cs.uic.edu/∼liub/FBS/sentiment-analysis.html.

where the μ superscript stands for "microaveraging," is that it is robust to class imbalance, which is useful, given the imbalanced nature of our data sets, while coinciding with MAE^{μ} on perfectly balanced data sets (those with exactly the same number of test documents for each class). For completeness, we have computed, in addition to MAE^{M}, MAE^{μ} results for all our experiments.

4.1.3 Learning Algorithms. We have tested our methods with two different SVM-based learning algorithms for ordinal regression: ϵ-SVR (Drucker, Burges, Kaufman, Smola, & Vapnik, 1997), originally devised for linear regression and which we have adapted to solve ordinal regression problems, and SVOR (Chu & Keerthi, 2007), which was specifically devised for solving ordinal regression.

ϵ-support vector regression (ϵ-SVR) is the original formulation of support vector regression as proposed in Drucker et al. (1997); we have used the implementation from the freely available LibSvm library.[12] ϵ-SVR can be adapted to the case of ordinal regression by mapping the rating scale onto a set of consecutive natural numbers (in our case, we have simply mapped the sequence $\langle$1 Star, . . . , 5 Stars$\rangle$ onto the sequence $\langle 1, \ldots, 5\rangle$) and rounding the real-valued output of the classifier to the nearest natural number in the sequence.

SVOR (Chu & Keerthi, 2007) consists of a newer algorithm that tackles the ordinal regression problem without using any a priori information on the classes, and by finding $(n-1)$ thresholds that divide the real-valued line into n consecutive intervals corresponding to the n ordered classes. Unlike the algorithm of Shashua and Levin (2003), the set of $(n-1)$ thresholds generated by SVOR is guaranteed to be properly ordered. The authors propose two different variants of SVOR: the first, called SVOREX (Support Vector Ordinal Regression with EXplicit constraints), takes into account only the training examples of adjacent classes in order to determine the thresholds, while the second, SVORIM (Support Vector Ordinal Regression with IMplicit constraints), takes into account all the training examples from all of the classes. Given that the authors have experimentally shown SVORIM to outperform SVOREX, the former (in the implementation available from (Chu & Keerthi, 2007) is the variant we have adopted for our experiments.[13]

Both ϵ-SVR and SVOR use the sequential minimal optimization algorithm for SVMs (Platt, 1999), and both map the solution on the real-valued line. The main difference between them is the use of a priori information. In fact, when using ϵ-SVR, the user needs to explicitly specify a mapping of the rating scale onto a sequence of naturals and set the thresholds between

[12] http://www.csie.ntu.edu.tw/∼cjlin/libsvm/.
[13] http://www.gatsby.ucl.ac.uk/∼chuwei/svor.htm.

these latter, while SVOR automatically derives all the needed information from the training set.

We have optimized the γ and C parameters of both learners on the validation sets described in section 4.1; the validation has been carried out with the full feature set ($\xi = 1$), and the values that have proven optimal have later been used for all reduction levels. The γ parameter has been optimized by testing all values from 2^{-15} to 2^{3} with step 2 in the exponent, while the C parameter has been optimized by testing all values from 2^{-11} to 2^{9} also with step 2 in the exponent. These ranges and steps are the ones recommended by the creators of LibSvm in the readme file.

4.1.4 Baselines. For all our experiments we have used three different baseline methods: the first is the PRP method of Mukras et al. (2007), the second is the *Var* method of Shimada and Endo (2008), and the third is the *RR*(*Var*) method of Baccianella et al. (2009b; see section 2 for details).

We also draw comparisons with the "trivial baseline," that is, the method that consists of trivially assigning all test documents to the "trivial class," defined as follows. For a given (binary, ordinal, or other) classification problem, a trivial classifier $\tilde{\Phi}_j$ may be defined as a classifier that assigns all documents to the same class c_j; accordingly, for a given error measure E, the trivial class $\tilde{c}$ may be defined as the class that minimizes E on the training set Tr across all trivial classifiers, that is, $\tilde{c} = \arg\min_{c_j \in C} E(\tilde{\Phi}_j, Tr)$. Baccianella et al. (2009a) shows that for both MAE^M and MAE^μ, the trivial class $\tilde{\Phi}_k$ need not be the majority class, as instead happens for standard multiclass classification when Hamming distance (the error rate) is the chosen error measure. For instance, for both the TripAdvisor-15763 and Amazon-83713 data sets, 4 Stars is the trivial class when the error measure is MAE^μ, since we obtain lower MAE^μ ($MAE^\mu = 0.805$ for TripAdvisor-15763 and $MAE^\mu = 1.171$ for Amazon-83713) in assigning all training documents 4 Stars than by assigning all of them 5 Stars, which is the majority class. Baccianella et al. (2009a) also shows that the trivial class(es) for MAE^M are always the middle class $c_{\frac{n+1}{2}}$ (when there is an odd number of classes) or the middle classes $c_{\lfloor \frac{n+1}{2} \rfloor}$ and $c_{\lceil \frac{n+1}{2} \rceil}$. In both the TripAdvisor-15763 and Amazon-83713 data sets, there is a single middle class, 3 Stars, which is then the trivial class for MAE^M (uniform assignment to the trivial class yields $MAE^M = 1.200$ for both data sets).

4.1.5 Experimental Protocol. As a vectorial representation, after stop word removal (and no stemming) we use standard bag of words. Feature weights have been obtained using the ltc variant (Salton & Buckley, 1988) of the well-known *TFIDF* class of weighting functions, that is,

$$TFIDF(t_k, d_i) = TF(t_k, d_i) \cdot \log \frac{|Tr|}{\#_{Tr}(t_k)}, \tag{4.3}$$

where $\#_{Tr}(t_k)$ denotes the number of documents in Tr in which t_k occurs at least once and

$$TF(t_k, d_i) = \begin{cases} 1 + \log \#(t_k, d_i) & \text{if } \#(t_k, d_i) > 0 \\ 0 & \text{otherwise} \end{cases}, \tag{4.4}$$

where $\#(t_k, d_i)$ denotes the number of times t_k occurs in d_i. Weights obtained by equation 4.3 are normalized through cosine normalization,

$$w_{ki} = \frac{TFIDF(t_k, d_i)}{\sqrt{\sum_{s=1}^{|\mathcal{T}|} TFIDF(t_s, d_i)^2}} \tag{4.5}$$

Note that as Baccianella et al. (2009b) discussed in detail, bag of words is certainly not the optimal method for generating the internal representations of product reviews: two expressions such as, "A great hotel in a horrible town!" and, "A horrible hotel in a great town!" would receive identical bag-of-words representations while expressing opposite evaluations of the hotel being reviewed. The reason that we have chosen bag of words in this letter (instead of the more complex and linguistically richer representations we have championed in Baccianella et al., 2009b) is to guarantee easier replicability by other researchers of the results presented here.

We have run all of our experiments for all the 100 reduction levels $\xi \in \{0.001, 0.01, 0.02, 0.03, \ldots, 0.99\}$. For the $Var * IDF$, $RR(Var * IDF)$, and $RR(AC * IDF)$ methods, we have set the smoothing parameter ϵ to 0.1. For the same methods, we have (individually for each method) optimized the a parameter on the validation sets described in section 4.1 and then retrained the optimized classifier on the full training set (i.e., including the validation set). During validation, all integer values in the range [1,20] were tested (values in [0,1) had already shown a dramatic deterioration in effectiveness in preliminary experiments and were thus not investigated in detail), and the best value for a given method was retained; neither 1 nor 20 ever turned out to be the best values, so no exploration outside the [1,20] interval was carried out. For all three methods the optimization was performed with $\xi = 0.10$ (since this is a paradigmatic reduction level in much of the literature on feature selection for text classification), and the value that proved optimal was chosen for all feature reduction levels. Previous experiments on the validation set had shown that in all cases, the chosen parameter value was optimal for any $\xi \in [0.001, 0.20]$, which are the most interesting reduction levels from an applicative point of view.

The optimal parameter values turned out to be fairly different from 1 (these were 8 for $Var * IDF$, 7 for $RR(Var * IDF)$, and 5 for $RR(AC * IDF)$). This confirms that the intuition of introducing a parameter a that modulates the impact of IDF on the entire formula is valuable.

For $RR(AC * IDF)$, the E error measure was taken to be $|\hat{\Phi}(d_i) - \Phi(d_i)|$ (i.e., absolute error), given that it is the document-level analogue of both MAE^M and MAE^μ. For $RR(RankMergeW)$ the error measure E in equation 3.6 was taken to be MAE^M.

4.2 Results. The results of our experiments are displayed in Figures 1 to 4, in which the effectiveness of each feature selection policy is plotted as a function of the tested reduction level. The two horizontal lines indicate the effectiveness (measured by MAE^M) obtained by using the full feature set ($\xi = 1$) or assigning all test documents to the trivial class.

The first observation that comes by observing Figures 1 to 4 is that the three baselines are dramatically inferior to the six novel techniques proposed in this letter. *PRP* is somehow comparable to our novel techniques for very aggressive reduction levels (e.g., $\xi = 0.01$) but drastically inferior to them for all other reduction levels, even underperforming, on Amazon-83713, the trivial baseline in the range $\xi \in [0.05, 0.70]$ with both learners (it somehow performs better, but still worse, than our six proposed techniques, on TripAdvisor-15763). *Var* is comparable to our techniques for the less aggressive reduction levels (i.e., $\xi \in [0.4, 1.0]$), but it yields very poor results for the more aggressive ones, even worse than the trivial baseline if $\xi \in [0.001, 0.15]$. This is likely due to the fact that the top-scoring features for the *Var* method—the only ones that get selected when the reduction level is very aggressive—are the so-called hapax legomena, features that occur in a single training document,[14] while when the reduction level is less aggressive "good" features (those with low variance and high frequency of occurrence) are also selected. In other words, *Var* clearly suffers from the inability to penalize features with low frequency of occurrence. As we will see, this inability is not present in $Var * IDF$. $RR(Var)$ performs uniformly worse than the proposed techniques for all reduction levels and on both data sets. It should be noted that for the more aggressive levels of reduction, $RR(Var)$ performs better than *Var*; this is likely due to the fact that the round-robin step is very important when the features are few, since its presence allows each class to be represented by at least some features that are highly discriminative for it. From now on, we will thus largely ignore the three baseline techniques and focus on discussing our six novel techniques and their differences.

A second observation (that immediately jumps to the eye once we look at Figures 1 to 4) is that our proposed techniques are fairly stable across $\xi \in [0.05, 1.0]$ and deteriorate, sometimes rapidly, only for the very aggressive levels, that is, for $\xi \in [0.001, 0.05]$). This is especially evident on the

[14] It is well known that the number of hapax legomena in a given corpus of texts is usually 40% to 60% of the total number of words that occur at least once in the entire corpus (Kornai, 2008).

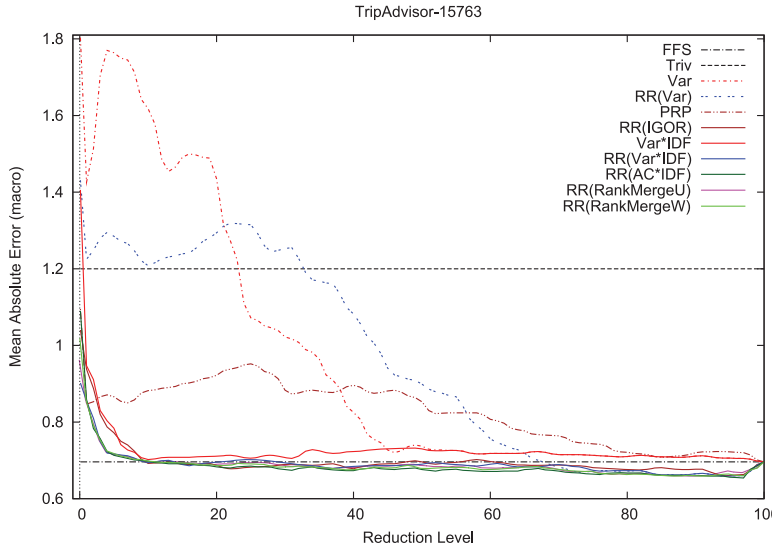

Figure 1: Results obtained with three baseline feature selection techniques (colored dotted lines) and our six novel techniques (colored solid lines) on the TripAdvisor-15763 data set with the ϵ-SVR learner. Results are evaluated with MAE^M; lower values are better. FFS: the full feature set (i.e., $\xi = 1$): Triv uniform assignment to the trivial class.

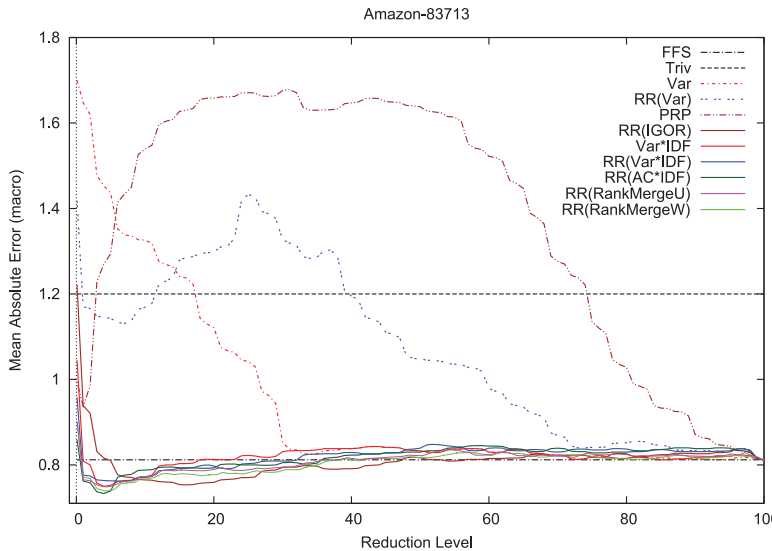

Figure 2: Same as Figure 1 but on Amazon-83713 instead of on TripAdvisor-15763.

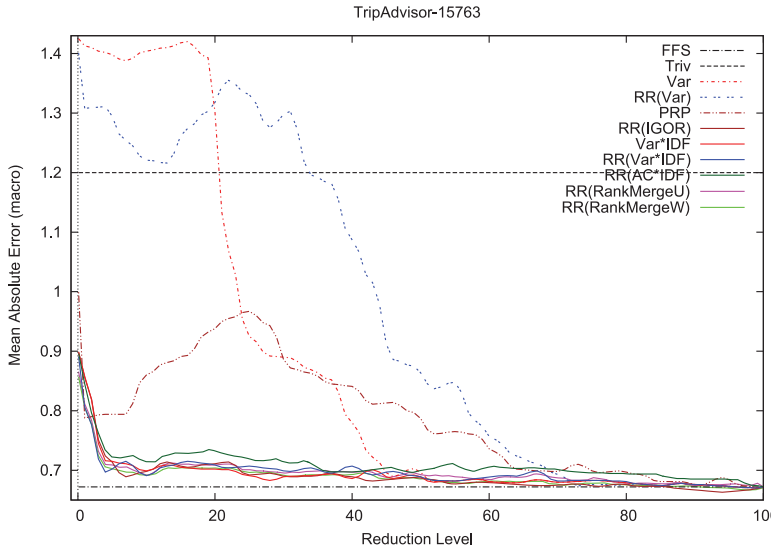

Figure 3: Same as Figure 1 but with the SVOR learner instead of the ϵ-SVR learner.

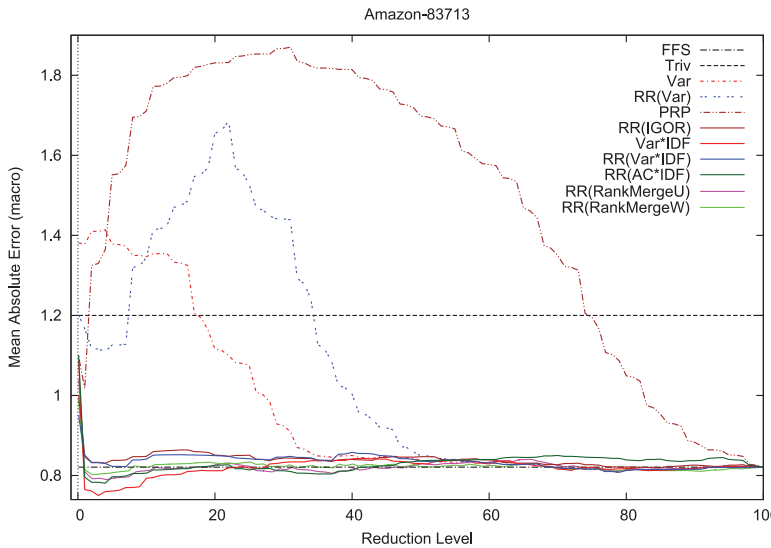

Figure 4: Same as Figure 1 but on Amazon-83713 instead of TripAdvisor-15763 and with the SVOR learner instead of the ϵ-SVR learner.

Amazon-83713 data set (by far the larger of the two) for both learners. This is in stark contrast with the instability of the baselines; for example, as already noted, both *PRP* and *Var* perform reasonably well for some reduction levels but disastrously for others. For $\xi \in [0.05, 1.0]$ the accuracy is, for each of our six techniques, more or less comparable to the accuracy obtained with the full feature set (i.e., with no feature selection).

In order to better focus on the differences among our six techniques, we analyze Figures 5 to 8, which present the same results of Figures 1 to 4, respectively, in close-up view, zooming in on our six techniques. Looking at these close-ups shows that the full feature set tends to be, although by a small margin, the best choice in the TripAdvisor-15763 data set, while the situation is less clear-cut in the much larger Amazon-83713 data set, with the proposed techniques slightly underperforming the full feature set for $\xi \in [0.3, 1.0]$ and outperforming it for $\xi \in [0.01, 0.3]$. This is very good news, since it indicates that one can reduce the feature set by an order of magnitude (with the ensuing benefits in terms of training time and, especially important, testing time efficiency) and obtain an accuracy equal to or even slightly superior (roughly a 10% improvement, in the best cases) to that obtainable with the full feature set. This is clearly reminiscent of the results obtained by Yang and Pedersen (1997), who, in their seminal paper on feature selection for binary text classification, showed that the best feature selection techniques could allow exactly that: an improvement in effectiveness of about 10% when the size of the feature set is reduced by one order of magnitude ($\xi = .10$), the reduction level at which the best performance was obtained.

In this respect, the stark difference between TripAdvisor-15763 and Amazon-83713 might be due to the fact that Amazon-83713 is a multi-topic data set (since it contains reviews about MP3, USB, GPS, Wireless 802.11, Digital Camera, and Mobile Phone), while TripAdvisor-15763 is focused on hotels only. This means that the vocabulary of the Amazon-83713 data set is much more varied, and likely contains many terms that are specific to just one subdomain instead of general utility. The vocabulary of the TripAdvisor-15763 data set is likely more focused and thus contains fewer terms of little utility.

When using ϵ-SVR on the Amazon-83713 data set (see Figure 6), some of our methods (e.g., $RR(AC * IDF)$) even marginally outperform the full feature set when the size of the feature set is reduced by two orders of magnitude ($\xi = .01$) and only marginally underperform it when the reduction is by three orders of magnitude ($\xi = .001$), regardless of the learner used. We think this is striking performance.

It is not easy to decide which of the six techniques we have proposed is the best; we need only to briefly glance at Figures 5 to 8 in order to realize that the curves corresponding to our six techniques keep intersecting each other, showing that none of them consistently outperforms the others regardless of data set and learner. For instance, when we use ϵ-SVR on

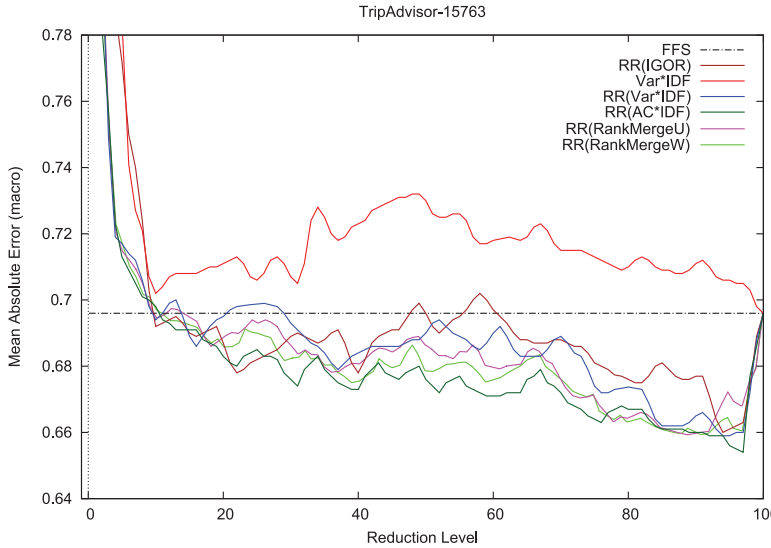

Figure 5: Close-up on the results obtained with six novel feature selection techniques on the TripAdvisor-15763 data set with the ϵ-SVR learner. FFS: the full feature set (i.e., $\xi = 1$).

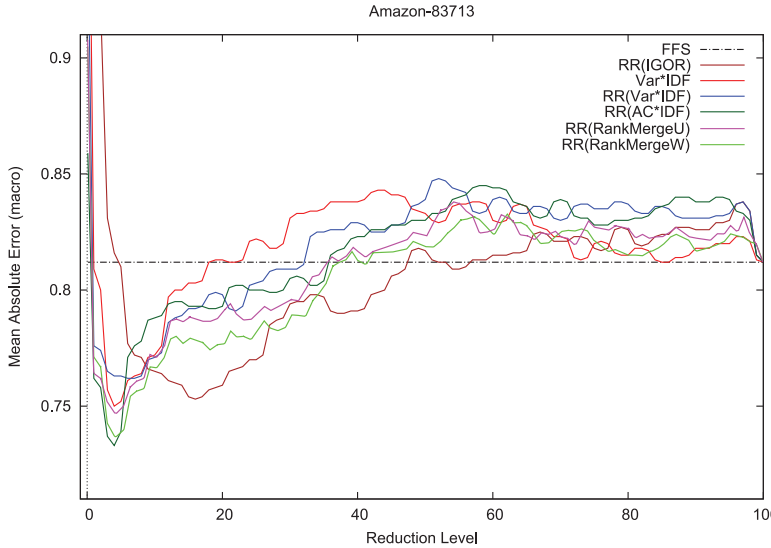

Figure 6: Same as Figure 5 but on Amazon-83713 instead of TripAdvisor-15763.

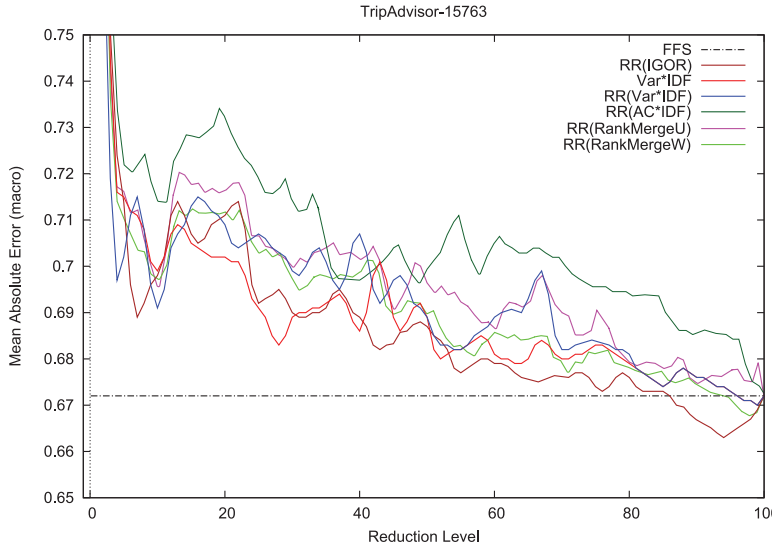

Figure 7: Same as Figure 5 but with the SVOR learner instead of the ϵ-SVR learner.

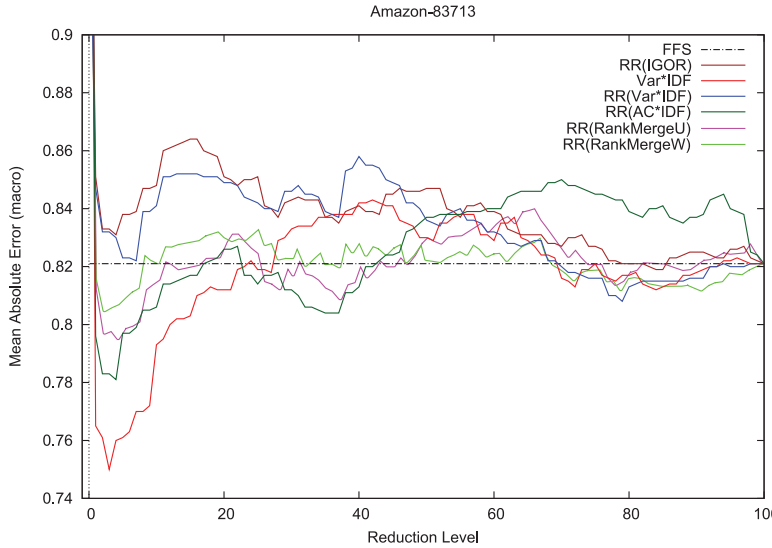

Figure 8: Same as Figure 5 but on Amazon-83713 instead of TripAdvisor-15763 and with the SVOR learner instead of with the ϵ-SVR learner.

the Amazon-83713 data set (see Figure 6), $RR(IGOR)$ is clearly the best when $\xi \in [0.10, 0.70]$, while when $\xi \in [0.001, 0.10]$, the best performers are $RR(AC * IDF)$ and $RR(RankMergeW)$. On the TripAdvisor-15763 data set (see Figure 5), $RR(AC * IDF)$ is the best performer when $\xi \in [0.20, 0.70]$, while for other reduction levels, several of our methods tend to perform equivalently.

In order to get a clearer sense of the relative merits of these six techniques, in Table 2 we report the performance of all the techniques discussed for two representative reduction levels, .01 and .10, averaged across the two data sets and the two learners used. The results reported show that when MAE^M is used as the evaluation measure, $RR(RankMergeU)$ and $RR(RankMergeW)$ are the best performers for $\xi = .01$, and they are close to being the best performers also for $\xi = .01$, where $Var * IDF$ outperforms them by a narrow margin. An interesting fact that emerges from Table 2 is that $RR(RankMergeU)$ is (slightly) better than $RR(RankMergeW)$ from every point of view (MAE^M and MAE^μ, $\xi = .01$ and $\xi = .10$).

Finally, note that we have also evaluated all the experiments reported here according to the MAE^μ measure; we do not report separate plots for them for reasons of space, and because, as fully argued in Baccianella et al. (2009a), we believe MAE^M more faithfully represents what a user wants from an ordinal regression device.[15] The MAE^μ results show better performance of $RR(Var * IDF)$, closely followed by $RR(RankMergeU)$.

The last row of Table 2 reports the variance of the MAE^M values obtained by the various methods across all the reduction factors $\xi \in [0.001, 0.99]$, the two learners, and the two data sets. The first thing to observe is that all of our six methods are characterized by lower variance (often one order of magnitude lower) than all of the three baseline methods, so our six methods largely improve on the three baselines in terms of sheer performance and stability. Among our six methods, the lowest variance is obtained by $RR(RankMergeU)$, which indicates better stability of the method across all values of ξ. This, along with the fact that Table 2 shows $RR(RankMergeU)$ to be one of the best performers, candidates $RR(RankMergeU)$ are our method of choice.

4.2.1 Statistical Significance Tests. We also tested if the measured differences among the three baseline methods and our six newly proposed methods are statistically significant. In order to do this, we have performed a Wilcoxon signed-rank test between every pair consisting of either one of the

[15] A spreadsheet with detailed figures for all 28×100 experiments conducted, along with large-size versions of all the plots, can be downloaded from http://hlt.isti.cnr.it/FS4OR/. Note that the results obtained with the techniques involving a round-robin step with the ϵ-SVR package are different from the analogous results presented in the earlier version of this paper (Baccianella et al., 2010) because of a bug in our software that we detected after Baccianella et al. (2010) had gone to print. The bug had the consequence that fewer features than actually declared were being selected by the techniques.

Table 2: Performance of Different Selection Features.

		Baselines				Our New Techniques					
	ξ	Trivial	*Var*	*RR*(*Var*)	*PRP*	*Var* ∗ *IDF*	*RR*(*Var* ∗ *IDF*)	*RR*(*IGOR*)	*RR*(*AC* ∗ *IDF*)	*RR*(*RankMergeU*)	*RR*(*RankMergeW*)
MAE^M	.01	1.200	1.466	1.214	0.885	0.833	0.809	0.885	0.863	**0.805**	0.806
	.10	1.200	1.422	1.231	1.238	**0.727**	0.735	0.738	0.737	0.729	0.731
MAE^μ	.01	0.988	1.027	1.251	0.656	0.644	**0.621**	0.640	0.643	0.623	0.626
	.10	0.988	0.978	0.949	0.831	0.605	**0.597**	0.618	0.620	0.604	0.606
$Var(MAE^M)$	.001-.99	—	7.93E-03	6.95E-03	1.64E-02	7.30E-03	6.90E-03	7.80E-03	6.60E-03	**6.20E-03**	6.30E-03

Notes: The first four rows show the performance of different feature selection functions at two representative reduction levels (.01 and .10), as averaged across the two different data sets and the two different learners used in this letter. The best performers are in bold. The last row indicates the variance of MAE^M across all the experiments.

three baseline methods and one of our six proposed methods or two of our six proposed methods (a total of $(3 \times 6) + (\frac{6\times5}{2}) = 33$ tests), based on the MAE^M values from all the 100 reduction factors and all four experimental setups (2 learners $\times$ 2 data sets).[16]

We have found that the differences for 30 of 33 pairs are statistically significant for a value of $p = 0.001$. The exceptions are the $(Var * IDF, RR(Var * IDF))$ and $(Var * IDF, RR(AC * IDF))$ pairs being statistically significant only for $p = 0.1$, and the $(RR(Var * IDF), RR(AC * IDF))$ pair for which $p > 0.1$. For these cases, the observed differences among these three methods cannot be considered strongly statistically significant (i.e., they are potentially due to chance). This indicates that the intuitions on which they are based are, even if radically different, equally good.

Altogether, the results of our significance tests indicate that $RR(RankMergeU) \succ RR(RankMergeW) \succ RR(IGOR) \succ \{RR(AC * IDF), RR(Var * IDF), Var * IDF\}$, where $\succ$ stands for "outperforms in a statistically significant sense" and $\{M_1, \ldots, M_n\}$ indicates that there are no statistically significant differences among methods $M_1, \ldots, M_n$.[17] All of our six methods outperform, in a statistically significant sense, all of the three baselines.

$RR(RankMergeU)$ turns out the best method of the lot. It is certainly not surprising that it outperforms the two methods it is based on, $RR(AC * IDF)$ and $RR(Var * IDF)$, since this is consistent with the findings of other authors (e.g., Renda & Straccia, 2003) who have witnessed the benefits deriving from rank aggregation. Rather, it is somehow surprising that $RR(RankMergeU)$ outperforms a more sophisticated method such as $RR(RankMergeW)$. Aside from being surprising, this is actually good news, since the conclusion we can draw is that we can obtain good accuracy results of $RR(RankMergeU)$ without incurring the additional computational cost that $RR(RankMergeW)$ entails, given that this latter requires a potentially expensive parameter estimation phase.

5 Computational Cost and Running Times

Before concluding, we analyze in detail the computational cost of our six FS methods.

[16] In a first version of this letter, instead of Wilcoxon signed-rank tests, we performed paired t-tests. However, as one of the anonymous reviewers pointed out, we had overlooked the need to test the gaussianity assumption. When we tested this assumption by means of the Shapiro-Wilk test, in many cases we obtained low p values, which indicate the rejection of the normality hypothesis. As a consequence, the use of the t-test is not legitimate in this case. We thus switched to the Wilcoxon signed-rank test, which does not require gaussianity as a prerequisite.

[17] Some of these results may appear to slightly contradict the results displayed in Table 2. However, there is no real contradiction, since the statistical significance tests are based on the results obtained for all 100 reduction factors, while Table 2 reports the results for only two such factors (.01 and .10).

The first method ($Var * IDF$) requires:

1. Computing the variance and IDF of each feature $t_k \in T$. For each such feature, we can safely assume that this can be done in constant time (so that the entire operation is $O(|T|)$), since computing variance requires only:
   - Computing the mean $\mu = \frac{1}{n}\sum_{j=1}^{n} j \cdot \#_{Tr}(t_k, c_j)$, which is facilitated by the fact that the number $\#_{Tr}(t_k, c_j)$ of training documents that contain feature t_k and belong to class c_j is computed at index construction time (and is thus already available at no cost), and by the fact that the number n of classes is usually small (it is 5 for the data sets we used, and in practical applications it is never higher than 10), and can thus be likened to a constant and neglected from the analysis;
   - Computing the variance $Var = \frac{1}{n}\sum_{j=1}^{n}(\#_{Tr}(t_k, c_j) - \mu)^2$, which is straightforward for the same reasons already described in the previous paragraph
   - Computing the IDF of feature t_k; however, this too is easily computed at index construction time and is thus already available at no cost
2. Sorting the features, which costs $O(|T| \log |T|)$

As a whole, $Var * IDF$ thus costs $O(|T| \log |T|)$. However, this does not include the optimization of parameter a of equation 3.1, which involves performing a learning and classification step for each value of a tested. Of course, the cost of this parameter optimization phase is difficult to quantify exactly, since it depends on the computational cost of the learning and classification algorithms chosen and on the number of different parameter settings that are being tested.

The second method ($RR(Var * IDF)$) is similar to the first, but for the fact that the feature ranking phase is replaced by the round-robin phase. This method involves (1) computing for each of the $|T|$ original features their mean, variance, and IDF, which can be done in $O(|T|)$: generating n different rankings of $|T|$ features, for which the overall cost is $O(|T| \log |T|)$: and scanning these rankings in a round-robin fashion while checking for potential duplicates, which costs $O(|S| \log |S|)$. In scanning, the $|S|$ factor corresponds to the cost of scanning the n rankings, while checking for potential duplicates is responsible for the $\log |S|$ factor; since $|S| \ll |T|$, the entire round-robin phase has thus an upper bound of $O(|T| \log |T|)$. As a whole, $RR(Var * IDF)$ is thus $O(|T| \log |T|)$. $RR(Var * IDF)$ also depends on parameter a, which requires optimization; the same considerations on parameter optimization as discussed for $Var * IDF$ apply here too.

The third method ($RR(IGOR)$) requires computing $(n-1)$ values of the IG function for each of the $|T|$ original features, which costs $O(|T|)$, and generating $(n-1)$ different rankings of $|T|$ features each, whose cost altogether is $O(|T| \log |T|)$. As for the case of $Var * IDF$, here we have

considered that the number $\#_{Tr}(t_k, c_j)$ of training documents that contain feature t_k and belong to class c_j is already available at no cost and that the number n of classes is usually small. After this, the same step as discussed for $RR(Var * IDF)$ needs to be carried out. Arguments analogous to the ones of the previous paragraph allow us to conclude that the cost of $RR(IGOR)$ is $O(|T| \log |T|)$.

The fourth method ($RR(AC * IDF)$) is, from a computational cost point of view, similar to $RR(IGOR)$, with the difference that n rankings (instead of $(n-1)$) need to be generated. The cost of $RR(AC * IDF)$ is thus, again, $O(|T| \log |T|)$. However, unlike $RR(IGOR)$, $RR(AC * IDF)$ depends on parameter a, which requires optimization; the same considerations on parameter optimization as discussed for $RR(Var * IDF)$ apply here too.

The fifth method ($RR(RankMergeU)$) requires, for each of the n classes, the computation of two rankings, one according to the scores generated by equation 3.1 and one according to those generated by equation 3.3. Similar arguments as the ones used above when discussing $RR(Var * IDF)$ and $RR(AC * IDF)$ allow us to conclude that this step is $O(|T| \log |T|)$. Merging, for each of the n classes, the two rankings and reranking the result altogether costs $O(|T| \log |T|)$ (where we use again the fact that n is usually small), and scanning these rankings in a round-robin fashion while checking for potential duplicates costs $O(|S| \log |S|)$. Since $|S| \ll |T|$, overall the process costs $O(|T| \log |T|)$. Again, $RR(RankMergeU)$ depends on parameter a (since the two methods being merged depend on it), which bring into play the considerations made above concerning parameter optimization.

The cost of the sixth method ($RR(RankMergeW)$) is obviously equivalent to that of $RR(RankMergeU)$ but for the fact that parameter optimization plays an even bigger role here, since (aside from a) the E_i parameters of equation 3.6 also need to be optimized.

In sum, the computational cost of all our six FS methods is $O(|T| \log |T|)$; however, all of the methods aside from $RR(IGOR)$ involve a parameter optimization phase, since the a parameter of equations 3.1 and 3.4 needs to be estimated for all methods other than $RR(IGOR)$. The $RR(RankMergeW)$ also requires the estimation of the E_i parameters of equation 3.6, but these are actually computed during the optimization of a, so no additional effort is involved. The cost of the parameter optimization phase is difficult to quantify exactly.

In practice, several improvements on the upper bounds indicated in the previous paragraphs can be made, for example, by using efficient index structures in secondary memory. In Table 3 we report the actual running times we have measured on the TripAdvisor-15763 and Amazon-83713 data sets for the six feature selection methods proposed in this letter. The times indicated refer to the entire process of (1) scoring the features, (2) ranking them according to the obtained scores, (3) performing (when applicable) the round-robin scheme, and (4) optimizing (in the $RR(RankMergeW)$ method)

Table 3: Running Times (in Seconds) of the Six Feature Selection Methods Proposed When Processing the TripAdvisor-15763 and Amazon-83713 Data Sets.

Function	TripAdvisor-15763	Amazon-83713
$Var * IDF$	$3.77 + 1605.16$	$25.02 + 4933.50$
$RR(Var * IDF)$	$3.78 + 1616.83$	$25.06 + 4957.16$
$RR(IGOR)$	3.82	25.14
$RR(AC * IDF)$	$3.83 + 1583.50$	$25.33 + 4883.50$
$RR(RankMergeU)$	$4.10 + 3200.33$	$25.41 + 9840.66$
$RR(RankMergeW)$	$4.11 + 3200.33$	$25.40 + 9840.66$

the parameters on the validation set. For the methods requiring parameter optimization, times are indicated as a sum of the time required for phases 1 to 3 plus that required for phase 4. All times reported in this section were measured on a commodity machine equipped with a 6-Core Intel i7 3 Ghz processor and 16 GB RAM.

Table 3 allows us to draw the following observations. First, computation times are largely dominated by the parameter optimization phase when present, while the impact of the previous phases is negligible. For instance, on the Amazon-83713 data set (by far the larger of the two), $RR(IGOR)$ (which does not need any parameter optimization) requires a mere 25 seconds, while the second-cheapest method ($RR(AC * IDF)$) is (notwithstanding the use of all the six cores in parallel in the parameter optimization phase) 196 times slower, requiring 4883 seconds $\approx$ 1.35 hours, which is not much different from the times measured for both $Var * IDF$ and $RR(Var * IDF)$. The most expensive methods are obviously $RR(RankMergeU)$ and $RR(RankMergeW)$, since the a parameter is optimized separately for each of the two contributing methods, $RR(Var * IDF)$ and $RR(AC * IDF)$, which require 9840 seconds $\approx$2.73 hours each.

6 Conclusion

In this letter, we have proposed six novel feature selection techniques for ordinal classification, all based on (sometimes radically) different intuitions: (1) $Var * IDF$ introduces the notion that, other things being equal, features occurring in more documents are more valuable; (2) $RR(Var * IDF)$ is based on the idea that each rank in the rank set should receive equal attention and that this equal attention be delivered by a round-robin scheme; (3) underlying $RR(IGOR)$ is the idea of combining several calls, one for each pair of contiguous ranks, to a feature selection function originally devised for binary classification, with the intention of generating a good separator between a prefix and a suffix of the rank set; (4) $RR(AC * IDF)$ tries instead to directly minimize, within the feature selection function, the specific measure of error used for evaluation; and (5) $RR(RankMergeU)$ and

RR(*RankMergeW*) try to combine the strengths of some among the above intuitions using the rank aggregation techniques typical of metasearch.

We have tested all of our proposed techniques against three baseline techniques (the only techniques for feature selection for ordinal classification proposed so far in the literature) on two data sets of product review data; one of these data sets is, to the best of our knowledge, the largest data set of product review data ever tested for ordinal classification purposes.

The experiments that we have carried out with thorough parameter optimization, for an extensive range of reduction levels, and complete with statistical significance tests have unequivocally shown that all of our six techniques are clearly superior to all three baselines on both data sets in terms of both sheer accuracy and stability. The experiments on the Amazon-83713 data set (by far the larger of the two) seem to indicate that all techniques deliver fairly stable performance across the range [0.05,1.0] of reduction levels and that performance tends to peak close to the 0.10 level. This indicates that it is viable to downsize the feature set by one order of magnitude while at the same time retaining, and sometimes even moderately improving on, the effectiveness delivered by the full feature set.

Choosing which among our six techniques should be used probably depends on the kind of trade-off between accuracy and efficiency desired. If efficiency is a factor, the preferred method should probably be *RR*(*IGOR*); since it is parameter free, it offers a good level of accuracy for a fraction of the computational cost incurred by the other techniques (among which the two methods based on rank aggregation stand out as particularly expensive), and our statistical significance tests ranked it higher than $Var * IDF$, *RR*($Var * IDF$), and *RR*($AC * IDF$). If computational cost is not an issue, sheer accuracy and stability considerations indicate that *RR*(*RankMergeU*) should be the method of choice.

Acknowledgments

We thank Nitin Jindal and Bing Liu for kindly sharing with us their "Amazon" dataset, Salvatore Ruggieri for an important pointer to the literature, and Umberto Straccia for useful discussions on rank merging.

References

Akritidis, L., Katsaros, D., & Bozanis, P. (2011). Effective rank aggregation for metasearching. *Journal of Systems and Software*, *84*(1), 130–143.

Baccianella, S., Esuli, A., & Sebastiani, F. (2009a). Evaluation measures for ordinal text classification. In *Proceedings of the 9th IEEE International Conference on Intelligent Systems Design and Applications* (pp. 283–287). Piscataway, NJ: IEEE.

Baccianella, S., Esuli, A., & Sebastiani, F. (2009b). Multi-facet rating of product reviews. In *Proceedings of the 31st European Conference on Information Retrieval* (pp. 461–472). New York: ACM Press.

Baccianella, S., Esuli, A., & Sebastiani, F. (2010). Feature selection for ordinal regression. In *Proceedings of the 25th ACM Symposium on Applied Computing* (pp. 1748–1754). New York: ACM Press.

Baccianella, S., Esuli, A., & Sebastiani, F. (2013). Using micro-documents for feature selection: The case of ordinal text classification. *Expert Systems with Applications*, *40*(11), 4687–4696.

Chu, W., & Keerthi, S. S. (2007). Support vector ordinal regression. *Neural Computation*, *19*(3), 145–152.

Dash, M., Choi, K., Scheuermann, P., & Liu, H. (2002). Feature selection for clustering—a filter solution. In *Proceedings of the 2nd IEEE International Conference on Data Mining* (pp. 115–122). Piscataway, NJ: IEEE.

Drucker, H., Burges, C. J., Kaufman, L., Smola, A., & Vapnik, V. (1997). Support vector regression machines. In M. Mozer, M. I. Jordan, & T. Petsche (Eds.), *Advances in neural information processing systems, 9* (pp. 155–161). Cambridge, MA: MIT Press.

Dy, J. G. (2007). Unsupervised feature selection. In H. Liu & H. Motoda (Eds.), *Computational Methods of Feature Selection* (pp. 19–39). London: CRC Press/Taylor and Francis Group.

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, *3*, 1289–1305.

Forman, G. (2004). A pitfall and solution in multi-class feature selection for text classification. In *Proceedings of the 21st International Conference on Machine Learning* (pp. 38–45). New York: ACM Press.

Forman, G. (2007). Feature selection for text classification. In H. Liu & H. Motoda (Eds.), *Computational Methods of Feature Selection* (pp. 257–276). London: CRC Press/Taylor and Francis Group.

Fouad, S., & Tino, P. (2012). Adaptive metric learning vector quantization for ordinal classification. *Neural Computation*, *24*(11), 2825–2851.

Geng, X., Liu, T.-Y., Qin, T., & Li, H. (2007). Feature selection for ranking. In *Proceedings of the 30th ACM International Conference on Research and Development in Information Retrieval* (pp. 407–414). New York: ACM Press.

Goldberg, A. B., & Zhu, X. (2006). Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In *Proceedings of the HLT/NAACL Workshop on Graph-Based Algorithms for Natural Language Processing* (pp. 45–52). East Stroudsburg, PA: Association for Computational Linguistics.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, *3*, 1157–1182.

Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, *46*(1/3), 389–422.

Hu, Q., Pan, W., Zhang, L., Zhang, D., Song, Y., Guo, M., & Yu, D. (2012). Feature selection for monotonic classification. *IEEE Transactions on Fuzzy Systems*, *20*(1), 69–81.

Jindal, N., & Liu, B. (2007). Review spam detection. In *Proceedings of the 16th International Conference on the World Wide Web* (pp. 1189–1190). New York: ACM Press.

John, G. H., Kohavi, R., & Pfleger, K. (1994). Irrelevant features and the subset selection problem. In *Proceedings of the 11th International Conference on Machine Learning (ICML 1994)* (pp. 121–129). San Francisco: Morgan Kaufmann.

Kamishima, T., & Akaho, S. (2006). Dimension reduction for supervised ordering. In *Proceedings of the 6th IEEE International Conference on Data Mining* (pp. 330–339). Piscataway, NJ: IEEE.

Kornai, A. (2008). *Mathematical Linguistics*. Heidelberg: Springer.

Levina, E., & Bickel, P. (2001). The earth mover's distance is the Mallows distance: Some insights from statistics. In *Proceedings of the International Conference on Computer Vision* (pp. 251–256). San Mateo, CA: IEEE Computer Society Press.

Likert, R. (1932). A technique for the measurement of attitudes. *Archives of Psychology*, *140*(1), 1–55.

Liu, H., & Motoda, H. (Eds.). (2007). *Computational methods of feature selection*. London: CRC Press/Taylor and Francis Group.

Liu, T., Liu, S., Chen, Z., & Ma, W.-Y. (2003). An evaluation on feature selection for text clustering. In *Proceedings of the 20th International Conference on Machine Learning* (pp. 488–495). Cambridge, MA: AAAI Press.

Miller, A. (2002). *Subset selection in regression* (2nd ed.). London: Chapman and Hall.

Miller, D. C., & Salkind, N. J. (Eds.). (2002). *Handbook of research design and social measurement* (6th ed.). Thousand Oaks, CA: Sage.

Mukras, R., Wiratunga, N., Lothian, R., Chakraborti, S., & Harper, D. (2007). Information gain feature selection for ordinal text classification using probability re-distribution. In *Proceedings of the IJCAI 2007 Workshop on Text Mining and Link Analysis*. N.p.

Pang, B., & Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Meeting of the Association for Computational Linguistics* (pp. 115–124). East Stroudsburg, PA: Association for Computational Linguistics.

Platt, J. C. (1999). Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C.J.C. Burges, & Smola, A. J. (Eds.), *Advances in kernel methods: Support vector learning* (pp. 185–208). Cambridge, MA: MIT.

Potharst, R., & Feelders, A. J. (2002). Classification trees for problems with monotonicity constraints. *SIGKDD Explorations*, *4*(1), 1–10.

Renda, M. E., & Straccia, U. (2003). Web metasearch: Rank vs. score based rank aggregation methods. In *Proceedings of the 18th ACM Symposium on Applied Computing* (pp. 841–846). New York: ACM Press.

Rubner, Y., Tomasi, C., & Guibas, L. J. (2000). The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, *40*(2), 99–121.

Rüschendorf, L. (2001). Wasserstein metric. In M. Hazewinkel (Ed.), *Encyclopaedia of mathematics*. Dordrecht: Kluwer.

Saeys, Y., Inza, I., & Larrañaga, P. (2007). A review of feature selection techniques in bioinformatics. *Bioinformatics*, *23*(19), 2507–2517.

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

Shashua, A., & Levin, A. (2003). Ranking with large margin principle: Two approaches. In S. Becker, S. Thrün, & K. Obermayer (Eds.), *Advances in neural information processing systems, 15* (pp. 937–944). Cambridge, MA: MIT Press.

Shimada, K., & Endo, T. (2008). Seeing several stars: A rating inference task for a document containing several evaluation criteria. In *Proceedings of the 12th Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 1006–1014). New York: Springer-Verlag.

Spärck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, *28*(1), 11–21.

Yang, Y., & Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning* (pp. 412–420). San Francisco: Morgan Kaufmann.

Received May 23, 2013; accepted October 3, 2013.