Utility-Theoretic Ranking for Semiautomated Text Classification

GIACOMO BERARDI and ANDREA ESULI, Italian National Council of Research
FABRIZIO SEBASTIANI, Qatar Computing Research Institute

Semiautomated Text Classification (SATC) may be defined as the task of ranking a set $\mathcal{D}$ of automatically labelled textual documents in such a way that, if a human annotator validates (i.e., inspects and corrects where appropriate) the documents in a top-ranked portion of $\mathcal{D}$ with the goal of increasing the overall labelling accuracy of $\mathcal{D}$, the expected increase is maximized. An obvious SATC strategy is to rank $\mathcal{D}$ so that the documents that the classifier has labelled with the lowest confidence are top ranked. In this work, we show that this strategy is suboptimal. We develop new utility-theoretic ranking methods based on the notion of *validation gain*, defined as the improvement in classification effectiveness that would derive by validating a given automatically labelled document. We also propose a new effectiveness measure for SATC-oriented ranking methods, based on the expected reduction in classification error brought about by partially validating a list generated by a given ranking method. We report the results of experiments showing that, with respect to the baseline method mentioned earlier, and according to the proposed measure, our utility-theoretic ranking methods can achieve substantially higher expected reductions in classification error.

Categories and Subject Descriptors: H. [**Information Retrieval**]: Information Systems; Retrieval Tasks and Goals—*Clustering and classification*; I. [**Machine Learning**]: Computing Methodologies; Learning Paradigms—*Supervised learning*

General Terms: Algorithm, Design, Experimentation, Measurements

Additional Key Words and Phrases: Text classification, supervised learning, semiautomated text classification, cost-sensitive learning, ranking

ACM Reference Format:
Giacomo Berardi, Andrea Esuli, and Fabrizio Sebastiani. 2015. Utility-theoretic ranking for semiautomated text classification. ACM Trans. Knowl. Discov. Data 1, 1, Article 6 (July 2015), 32 pages.
DOI: http://dx.doi.org/10.1145/2742548

1. INTRODUCTION

Suppose an organization needs to classify a set $\mathcal{D}$ of textual documents under classification scheme $\mathcal{C}$, and suppose that $\mathcal{D}$ is too large to be classified manually, so that resorting to some form of automated text classification (TC) is the only viable option. Suppose also that the organization has strict accuracy standards, so that the level of effectiveness obtainable via state-of-the-art TC technology (including any possible improvements obtained via active learning (AL)) is not sufficient. In this case, the most plausible strategy is to train an automatic classifier $\hat{\Phi}$ on the available training data

---

This article is a revised and extended version of Berardi et al. [2012]. The order in which the authors are listed is purely alphabetical; each author has given an equal contribution to this work.
Authors' addresses: G. Berardi and A. Esuli, Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, Via Giuseppe Moruzzi 1, 56124 Pisa, Italy; emails: giacomo.berardi@isti.cnr.it; andrea.esuli@isti.cnr.it; F. Sebastiani, Qatar Computing Research Institute, PO Box 5825, Doha, Qatar; email: fsebastiani@qf.org.qa; F. Sebastiani is on leave from the Italian National Council of Research.

DOI: http://dx.doi.org/10.1145/2742548

Tr, improve it as much as possible (e.g., via AL), classify $\mathcal{D}$ by means of $\hat{\Phi}$, and then have a human editor validate (i.e., inspect and correct where appropriate) the results of the automatic classification. The human annotator will validate only a subset $\mathcal{D}' \subset \mathcal{D}$, for example, until she is confident that the overall level of accuracy of $\mathcal{D}$ is sufficient, or until she runs out of time. We call this scenario SATC.

An automatic TC system may support this task by ranking, after the classification phase has ended and before validation begins, the classified documents in such a way that, if the human annotator validates the documents starting from the top of the ranking, the expected increase in classification effectiveness that derives from this validation is maximized. This article is concerned with devising good ranking strategies for this task.

One obvious strategy (also used in Martinez-Alvarez et al. [2012]) is to rank the documents in ascending order of the confidence scores generated by $\hat{\Phi}$, so that the top-ranked documents are the ones that $\hat{\Phi}$ has classified with the lowest confidence. The rationale is that an increase in effectiveness can derive only by validating *misclassified* documents, and that a good ranking method is simply the one that top-ranks the documents with the highest probability of misclassification, which (in the absence of other information) we may take to be the documents which $\hat{\Phi}$ has classified with the lowest confidence.

In this work, we show that this strategy is, in general, suboptimal. Simply stated, the reason is that the improvements in effectiveness that derive from correcting a false positive or a false negative, respectively, may not be the same, depending on which evaluation function we take to represent our notion of "effectiveness." Additionally, the ratio between these improvements may vary during the validation process. In other words, an optimal ranking strategy must take into account the improvements mentioned earlier and how these impact on the evaluation function; we will thus look at ranking methods based on *explicit loss minimization*, that is, optimized for the specific effectiveness measures used.

The contributions of this article are the following. First, we develop new utility-theoretic ranking methods for SATC based on the notion of *validation gain*, defined as the improvement in effectiveness that would derive by correcting a given type of mistake (i.e., false positive or false negative). Second, we propose a new evaluation measure for SATC based on a probabilistic user model, and use it to evaluate our experiments on standard TC datasets. The results of these experiments show that, with respect to the confidence-based baseline method discussed earlier, our ranking methods are substantially more effective.

The rest of the article is organized as follows. Section 2 reviews related work, while Section 3 sets the stage by introducing preliminary definitions and notation. Section 4 describes our base utility-theoretic strategy for ranking the automatically labelled documents, while in Section 5 we propose a novel effectiveness measure for this task based on a probabilistic user model. Section 6 reports the results of our experiments in which we test the effectiveness of ranking strategies by simulating the work of a human annotator that validates variable-sized portions of the labelled test set. In Section 7, we address a potential problem deriving from the "static" nature of our strategy, by describing a "dynamic" (albeit computationally more expensive) version of the same strategy, and draw an experimental comparison between the two. In Section 8, we acknowledge the existence of two different ways ("micro" and "macro") of averaging effectiveness results across classes, and show that the methods we have developed so far are optimized for macro-averaging; we thus develop and test methods optimized for micro-averaged effectiveness. Section 9 concludes by charting avenues for future research.

2. RELATED WORK

Many researchers have tackled the problem of how to improve on the accuracy delivered by an automatic text classifier when this accuracy is not up to the standards required by the application (as, e.g., stipulated in a Service Level Agreement).

A standard response to this problem is to ask human annotators to label additional data that can then be used in retraining a (hopefully) more accurate classifier. This can be done via the use of AL techniques (see e.g., Hoi et al. [2006] and Tong and Koller [2001]), that is, via algorithms that rank unlabelled documents in such a way that the top-ranked ones bring about, once manually labelled and used for retraining, the highest expected improvement in classification accuracy. Still, the improvement in accuracy that can be obtained via AL is limited: even by using the best AL algorithm, accuracy tends to plateau after a certain number of unlabelled documents have been manually annotated. When this plateau is reached, annotating more documents will not improve accuracy any further [Settles 2012]. Similar considerations apply when AL is carried out at the term level, rather than at the document level [Godbole et al. 2004; Raghavan et al. 2006].

A related response to the same problem is to use *training data cleaning* (TDC) techniques (see e.g., Brodley and Friedl [1999], Esuli and Sebastiani [2013], and Fukumoto and Suzuki [2004]), that is, use algorithms that optimize the human annotator's efforts at correcting possible labelling mistakes in the training set. TDC algorithms rank the training documents in such a way that the top-ranked ones bring about, once their labels are manually checked and then used for retraining, the highest expected improvement in classification accuracy. In other words, TDC is to labelled training documents what AL is to unlabelled ones. Similarly to what happens in AL, in many applicative contexts high enough accuracy levels cannot be attained even at the price of carefully validating the entire training set for labelling mistakes.

Yet another response may be the use of some form of *weakly supervised learning/semisupervised learning*, that is, of techniques that allow training a classifier when training data are few, often leveraging unlabelled data along with the labelled training data [Chapelle et al. 2006; Zhu and Goldberg 2009]. This solution relies on the fact that unlabelled data is often available in large quantities, sometimes even from the same source where the training and test data originate. Similarly to the cases of AL and TDC, improvements with respect to the results of the purely supervised setting may be obtained, but these improvements are going to be limited anyway.

In conclusion, when the required accuracy standards are high, neither TDC, nor AL, nor weakly supervised/semisupervised learning, nor a combination of them, may suffice to reach up to these standards. In this case, after either or all such techniques have been applied, we can only resort to manual validation of part of the automatically classified documents by a human annotator. Supporting this last phase is the goal of SATC.

All the techniques discussed earlier are different from SATC, since in SATC we are not concerned with improving the quality of the trained classifier. We are instead concerned with improving the quality of the automatically classified *test* set, typically after all attempts at injecting additional quality in the automatic classifier (and in the training set) have proved insufficient; in particular, no retraining/reclassification phase is involved in SATC.

Active learning. As remarked earlier, SATC certainly bears relations to AL. In both SATC and in the *selective sampling* approach to AL ([Lewis and Catlett 1994]; also known as *pool-based* approach [McCallum and Nigam 1998]), the automatically classified objects are ranked and the human annotator is encouraged to correct possible misclassifications by working down from the top of the ranked list. However, as

remarked earlier, the goals of the two tasks are different. In AL, we are interested in top-ranking the unlabelled documents that, once manually labelled, would maximize the information fed back to the learning process, while in SATC we are interested in top-ranking the unlabelled documents that, once manually validated, maximize the expected accuracy of the automatically classified document set. As a result, the optimal ranking strategies for the two tasks may be different too.

Some approaches to AL take into account the costs of misclassification, thus attributing different levels of importance to different types of error. In Kapoor et al. [2007], these costs are embedded into a decision-theoretic framework, which is reminiscent of our utility-theoretic framework. A value-of-information criterion is used in order to select samples which maximize profit, determined by the total risk of classification and the total cost of labelling. The total risk is formulated as a utility function in which the probability of each classification and the risk associated with it are taken into account. The concept of risk is reminiscent of the notion of "gain" defined in our utility function (see Section 4.2), but its purpose is to consider the human effort needed in correcting a misclassified sample [Vijayanarasimhan and Grauman 2009]. Therefore, this decision-theoretic strategy is not aimed to directly improve classification accuracy, but to minimize the manual work of the annotator, which is quantified by the risk and the cost of labelling.

Semiautomated TC. While AL (and, to a much lesser degree, TDC) have been investigated extensively in a TC context, SATC has been fairly neglected by the research community. While a number of papers (e.g., Larkey and Croft [1996], Sebastiani [2002], and Yang and Liu [1999]) have evoked the existence of this scenario, we are not aware of many published papers that either discuss ranking policies for supporting the human annotator's effort, or that attempt to quantify the effort needed for reaching a desired level of accuracy. For instance, while discussing a system for the automatic assignment of ICD9 classes to patients' discharge summaries, Larkey and Croft [1996] say "We envision these classifiers being used in an interactive system which would display the 20 or so top ranking [classes] and their scores to an expert user. The user could choose among these candidates (...)," but do not present experiments that quantify the accuracy that the validation activity brings about, or methods aimed at optimizing the cost effectiveness of this activity.

The recent paper [Martinez-Alvarez et al. 2012] tackles the related problem of deciding when a document is too difficult for automated classification, and should thus be routed to a human annotator. However, the method presented in the paper is not applicable to our case, since (a) it is undefined for documents with no predicted labels (a fairly frequent case in multilabel TC), and (b) it is undefined when the classification threshold is zero (again, a fairly frequent case in modern learning algorithms, including the two used in the present paper).

In a subsequent paper [Martinez-Alvarez et al. 2013], the same authors study a family of SATC methods that exploit "document difficulty", taking into account the confidence scores computed by the base classifiers. They also present a comparison between the techniques they propose and that presented in an earlier version of the present article [Berardi et al. 2012]; in this comparison, the former are claimed to outperform the latter on the Reuters-21578 dataset discussed in Section 6.4. However, this comparison is incorrect since the authors compare the results of their ranking methods as applied to confidence scores generated by support vector machines (SVMs), with those of the Berardi et al. [2012] ranking method as applied to confidences scores generated by a different learner (MP-Boost – see Esuli et al. [2006]). A correct comparison among ranking methods must instead be carried out by providing to all methods the same input, that is, the same confidence scores (whose generation is not part

of the method itself). The comparison reported in Martinez-Alvarez et al. [2013] is incorrect also because it is carried out in terms of the $ENER_\rho^\mu$ measure (see Section 5.3); instead, as stated in Berardi et al. [2012], the measure according to which the method of Berardi et al. [2012] should be evaluated is $ENER_\rho^M$, and not $ENER_\rho^\mu$, since it is $ENER_\rho^M$ that method was optimized for. In Section 8, we will indeed present SATC methods optimized for $ENER_\rho^\mu$.

An application of the method discussed in Section 7 to performing SATC in a market research context is presented in Berardi et al. [2014].

3. PRELIMINARIES

Given a set of textual documents $\mathcal{D}$ and a predefined set of classes $\mathcal{C} = \{c_1, \ldots, c_m\}$, (multiclass multilabel) TC is usually defined as the task of estimating an unknown *target function* $\Phi : \mathcal{D} \times \mathcal{C} \to \{-1, +1\}$, that describes how documents ought to be classified, by means of a function $\hat{\Phi} : \mathcal{D} \times \mathcal{C} \to \{-1, +1\}$ called the *classifier*[1]; $+1$ and -1 represent membership and nonmembership of the document in the class. Here, "multiclass" means that there are $m \geq 2$ classes, while "multilabel" refers to the fact that each document may belong to zero, one, or several classes at the same time. Multiclass multilabel TC is usually accomplished by generating m independent binary classifiers $\hat{\Phi}_j$, one for each $c_j \in \mathcal{C}$, each entrusted with deciding whether a document belongs or not to a class c_j. In this article, we will actually restrict our attention to classifiers $\hat{\Phi}_j$ that, aside from taking a binary decision $D_{ij} \in \{-1, +1\}$ on a given document d_i, also return a *confidence estimate* C_{ij}, that is, a numerical value representing the strength of their belief in the fact that D_{ij} is correct (the higher the value, the higher the confidence). We formalize this by taking a binary classifier to be a function $\hat{\Phi}_j : \mathcal{D} \to \mathbb{R}$ in which the sign of the returned value $D_{ij} \equiv sgn(\hat{\Phi}_j(d_i)) \in \{-1, +1\}$ indicates the binary decision of the classifier, and the absolute value $C_{ij} \equiv |\hat{\Phi}_j(d_i)|$ represents its confidence in the decision.

For the time being, we also assume that

$$F_1(\hat{\Phi}_j(Te)) = \frac{2TP_j}{2TP_j + FP_j + FN_j} \tag{1}$$

(the well-known harmonic mean of precision and recall) is the chosen evaluation measure for binary classification, where $\hat{\Phi}_j(Te)$ indicates the result of applying $\hat{\Phi}_j$ to the test set Te and TP_j, FP_j, FN_j, TN_j indicate the numbers of true positives, false positives, false negatives, true negatives in Te for class c_j. Note that F_1 is undefined when $TP_j = FP_j = FN_j = 0$; in this case, we take $F_1(\hat{\Phi}_j(Te)) = 1$, since $\hat{\Phi}_j$ has correctly classified all documents as negative examples. The assumption that F_1 is our evaluation measure is not restrictive; as will be evident later on in the article, our methods can be customized to any evaluation function that can be computed from a contingency table.

As a measure of effectiveness for multiclass multilabel TC, for the moment being we use *macro-averaged* F_1 (noted F_1^M), which is obtained by computing the class-specific F_1 values and averaging them across all the $c_j \in \mathcal{C}$. An alternative way of averaging across the classes (*micro-averaged* F_1) will be discussed in Section 8.

In this article, the set of unlabelled documents that the classifier must automatically label (and rank) in the "operational" phase will be represented by the test set Te.

[1]Consistently with most mathematical literature, we use the caret symbol (ˆ) to indicate estimation.

4. A RANKING METHOD FOR SATC BASED ON UTILITY THEORY

4.1. Ranking by Utility

For the time being, let us concentrate on the binary case, that is, let us assume there is a single class c_j that needs to be separated from its complement $\overline{c}_j$. The policy we propose for ranking the automatically labelled documents in $\hat{\Phi}_j(Te)$ makes use of *utility theory*, an extension of probability theory that incorporates the notion of *gain* (or *loss*) that derives from a given course of action [Anand 1993; von Neumann and Morgenstern 1944]. Utility theory is a general theory of rational action under uncertainty, and as such is used in many fields of human activity; for instance, one such field is betting, since in placing a certain bet we take into account (a) the probabilities of occurrence that we subjectively attribute to a set of outcomes (say, to the possible outcomes of a given football game), and (b) the gains or losses that we obtain, having bet on one of them, if the various outcomes materialize.

In order to explain our method, let us introduce some basics of utility theory. Given a set $A = \{\alpha_1, \alpha_2, \ldots\}$ of possible courses of action and a set $\Omega = \{\omega_1, \omega_2, \ldots\}$ of mutually disjoint events, the *expected utility* $U(\alpha_i, \Omega)$ that derives from choosing course of action α_i given that any of the events in Ω may occur, is defined as

$$U(\alpha_i, \Omega) = \sum_{\omega_k \in \Omega} P(\omega_k) G(\alpha_i, \omega_k), \tag{2}$$

where $P(\omega_k)$ is the probability of occurrence of event ω_k and $G(\alpha_i, \omega_k)$ is the *gain* obtained if α_i is chosen and event ω_k occurs. For instance, α_i may be the course of action "betting on Arsenal FC's win" and Ω may be the set of mutually disjoint events $\Omega = \{\omega_1, \omega_2, \omega_3\}$, where ω_1 = "Arsenal FC wins", ω_2 = "Arsenal FC and Chelsea FC tie", and ω_3 = "Chelsea FC wins"; in this case,

—$P(\omega_1)$, $P(\omega_2)$, $P(\omega_3)$ are the probabilities of occurrence that we subjectively attribute to the three events ω_1, ω_2, ω_3;
—$G(\alpha_i, \omega_1)$, $G(\alpha_i, \omega_2)$, $G(\alpha_i, \omega_3)$ are the economic rewards we obtain if we choose course of action α_i (i.e., we bet on the win of Arsenal FC) and the respective event occurs. Of course, this economic reward will be positive if ω_1 occurs and negative if either ω_2 or ω_3 occur.

When we face alternative courses of action, acting rationally means choosing the course of action that maximizes our expected utility. For instance, given the alternative courses of action α_1 = "betting on Arsenal FC's win", α_2 = "betting on Arsenal FC's and Chelsea FC's tie", α_3 = "betting on Chelsea FC's win", we should pick among $\{\alpha_1, \alpha_2, \alpha_3\}$ the course of action that maximizes $U(\alpha_i, \Omega)$.

How does this translate into a method for ranking automatically labelled documents? Assume we have a set $D = \{d_1, \ldots, d_n\}$ of such documents that we want to rank, and that c_j is the class we deal with. For instantiating Equation (2) concretely, we need

(1) to decide what our set $A = \{\alpha_1, \alpha_2, \ldots\}$ of alternative courses of action is;
(2) to decide what the set $\Omega = \{\omega_1, \omega_2, \ldots\}$ of mutually disjoint events is;
(3) to define the gains $G(\alpha_i, \omega_k)$;
(4) to specify how we compute the probabilities of occurrence $P(\omega_k)$.

Let us discuss each of these steps in turn.

Concerning Step 1, we will take the action of validating document d_i as course of action α_i. In this way, we will evaluate the expected utility $U_j(d_i, \Omega)$ (i.e., the expected increase in the overall classification accuracy of Te) that derives to the classification accuracy of class c_j from validating each document d_i, and we will be able to rank the

documents by their $U_j(d_i, \Omega)$ value, so as to top-rank the ones with the highest expected utility.

Concerning Step 2, we have argued in the Introduction that the increase in accuracy that derives from validating a document depends on whether the document is a true positive, a false positive, a false negative, or a true negative; as a consequence, we will take $\Omega = \{tp_j, fp_j, fn_j, tn_j\}$, where each of these events implicitly refers to the document d_i under scrutiny (e.g., tp_j denotes the event "document d_i is a true positive for class c_j"). Our utility function has thus the form

$$U_j(d_i, \Omega) = \sum_{\omega_k \in \{tp_j, fp_j, fn_j, tn_j\}} P(\omega_k) G(d_i, \omega_k). \tag{3}$$

How to address Step 3 (defining the gains) will be the subject of Sections 4.2 and 4.3, while Step 4 (computing the probabilities of occurrence) will be discussed in Section 4.4.

4.2. Validation Gains

We equate $G(d_i, fp_j)$ in Equation (3) with the average increase in $F_1(\hat{\Phi}_j(Te))$ that would derive by manually validating the label attributed by $\hat{\Phi}_j$ to a document d_i in FP_j. We call this the *validation gain* of a document in FP_j. Note that validation gains are independent of a particular document, i.e., $G(d', fp_j) = G(d'', fp_j)$ for all $d', d'' \in Te$. Analogous arguments apply to $G(d_i, tp_j)$, $G(d_i, fn_j)$, and $G(d_i, tn_j)$.

Quite evidently, $G(d_i, tp_j) = G(d_i, tn_j) = 0$, since when the human annotator validates the label attributed to d_i by $\hat{\Phi}_j$ and finds out it is correct, she will not modify it, and the value of $F_1(\hat{\Phi}_j(Te))$ will thus remain unchanged.

Concerning misclassified documents, it is easy to see that, in general, $G(d_i, fp_j) \neq G(d_i, fn_j)$. In fact, if a false positive is corrected, the increase in F_1 is the one deriving from removing a false positive and adding a true negative, that is,

$$\begin{aligned} G(d_i, fp_j) &= \frac{1}{FP_j}\left(F_1^{FP}\left(\hat{\Phi}_j(Te)\right) - F_1\left(\hat{\Phi}_j(Te)\right)\right) \\ &= \frac{1}{FP_j}\left(\frac{2TP_j}{2TP_j + FN_j} - \frac{2TP_j}{2TP_j + FP_j + FN_j}\right), \end{aligned} \tag{4}$$

where by $F_1^{FP}(\hat{\Phi}_j)$ we indicate the value of F_1 that would derive by correcting all false positives of $\hat{\Phi}_j(Te)$, that is, turning all of them into true negatives. Conversely, if a false negative is corrected, the increase in F_1 is the one deriving from removing a false negative and adding a true positive, that is,

$$\begin{aligned} G(d_i, fn_j) &= \frac{1}{FN_j}\left(F_1^{FN}\left(\hat{\Phi}_j(Te)\right) - F_1\left(\hat{\Phi}_j(Te)\right)\right) \\ &= \frac{1}{FN_j}\left(\frac{2(TP_j + FN_j)}{2(TP_j + FN_j) + FP_j} - \frac{2TP_j}{2TP_j + FP_j + FN_j}\right), \end{aligned} \tag{5}$$

where by $F_1^{FN}(\hat{\Phi}_j)$ we indicate the value of F_1 that would derive by turning all the false negatives of $\hat{\Phi}_j(Te)$ into true positives.

Equation (4) defines the gain deriving from the correction of a false positive as the *average* across the gains deriving from the correction of each false positive in the contingency table (and analogously for Equation 5). The advantage of such a definition is that such average gain can be computed once for all during the entire process. We will see a different definition, leading to a different SATC method, in Section 7.

4.3. Smoothing Contingency Cell Estimates

One problem that needs to be tackled in order to compute $G(d_i, fp_j)$ and $G(d_i, fn_j)$ is that the contingency cell counts TP_j, FP_j, FN_j are not known (since in operational settings we do not know which test documents have been classified correctly and which have been instead misclassified), and thus need to be estimated[2]. In order to estimate them, we make the assumption that the training set and the test set are independent and identically distributed. We then perform a k-fold crossvalidation (k-FCV) on the training set: if by TP_j^{Tr} we denote the number of true positives for class c_j resulting from the k-fold crossvalidation on Tr, the maximum-likelihood estimate of TP_j is $\hat{TP}_j^{ML} = TP_j^{Tr} \cdot |Te|/|Tr|$; same for $\hat{FP}_j^{ML}$ and $\hat{FN}_j^{ML}$[3].

However, these maximum-likelihood cell count estimates need to be smoothed, so as to avoid zero counts. In fact, if $\hat{TP}_j^{ML} = 0$, it would derive from Equation (4) that there is nothing to be gained by correcting a false positive, which is counterintuitive. Similarly, if $\hat{FP}_j^{ML} = 0$, the very notion of $F_1^{FP}(\hat{\Phi}_j)$ would be meaningless, since it does not make sense to speak of "removing a false positive" when there are no false positives; and the same goes for $\hat{FN}_j^{ML}$.

A second reason why $\hat{TP}_j^{ML}$, $\hat{FP}_j^{ML}$, $\hat{FN}_j^{ML}$ need to be smoothed is that, when $|Te|/|Tr| < 1$, they may give rise to negative values for $G(d_i, fp_j)$ and $G(d_i, fn_j)$, which is counterintuitive. To see this, note that $\hat{TP}_j^{ML}$, $\hat{FP}_j^{ML}$, $\hat{FN}_j^{ML}$ may not be integers (which is not bad per se, since the notions of precision, recall, and their harmonic mean intuitively make sense also when we allow the contingency cell counts to be nonnegative reals instead of the usual integers), and may be smaller than 1 (this happens when $|Te|/|Tr| < 1$). This latter fact is problematic, both in theory (since it is meaningless to speak of, say, removing a false positive from Te when "there are less than 1 false positives in Te") and in practice (since it is easy to verify that negative values for $G(d_i, fp_j)$ and $G(d_i, fn_j)$ may derive).

Smoothing has extensively been studied in language modelling for speech processing [Chen and Goodman 1996] and for ad hoc search in information retrieval (IR) [Zhai and Lafferty 2004]. However, the present context is slightly different, in that we need to smooth contingency tables, and not (as in the previous cases) language models. In particular, while the $\hat{TP}_j^{ML}$, $\hat{FP}_j^{ML}$, and $\hat{FN}_j^{ML}$ are the obvious counterparts of the document model resulting from maximum-likelihood estimation, there is no obvious counterpart to the "collection model", thus making the use of, for example, Jelinek–Mercer smoothing problematic. A further difference is that we here require the smoothed counts not only to be nonzero, but also to be ≥ 1 (a requirement not to be found in language modelling).

Smoothing has also been studied specifically for the purpose of smoothing contingency cell estimates [Burman 1987; Simonoff 1983]. However, these methods are inapplicable to our case, since they were originally conceived for contingency tables

[2]We will disregard the estimation of TN_j since it is unnecessary for our purposes, given that $F_1(\hat{\Phi}_j(Te))$ does not depend on TN_j.

[3]As in many other contexts, the assumption that the training set and the test set are independent and identically distributed may not be verified in practice; if it is not, in our case this leads to imprecise estimates of the contingency cell counts. While this may be suboptimal, there is practically nothing that we can do about it, since we do not know the real values of these counts; in other words, k-FCV is our "best possible shot" at estimating them in the absence of foreknowledge. As discussed in Section 6.5, we will exactly measure *how* suboptimal using k-FCV is, by running experiments in which an oracle feeds our utility-theoretic method with the true values of the contingency cells.

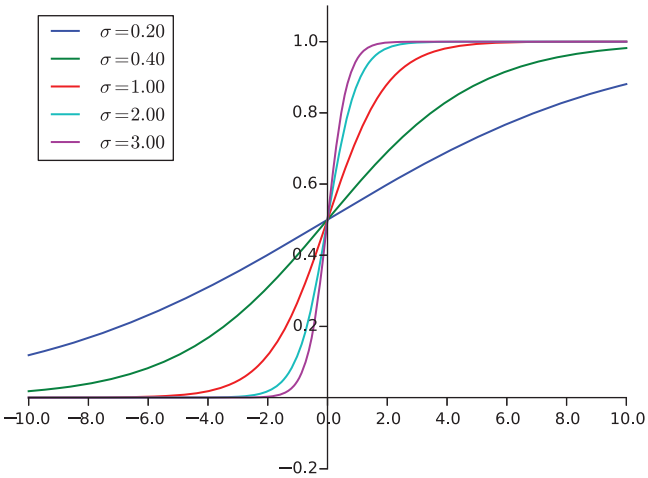

Fig. 1. The generalized logistic function.

characterized by a small (i.e., ≤ 1) ratio between the number of observations (which in our case is $|Te|$) and the number of cells (which in our case is 4); our case is quite the opposite. Additionally, these smoothing methods do not operate under the constraint that the smoothed counts should all be ≥ 1, which is a hard constraint for us.

For all these reasons, rather than adopting more sophisticated forms of smoothing, we adopt simple *additive smoothing* (also known as *Laplace smoothing*), a special case of Bayesian smoothing using Dirichlet priors [Zhai and Lafferty 2004] which is obtained by adding a fixed quantity to each of $\hat{TP}_j^{ML}$, $\hat{FP}_j^{ML}$, $\hat{FN}_j^{ML}$. As a fixed quantity we add 1, since it is the quantity that all our cell counts need to be greater than or equal to for Equations (4) and (5) to make sense. We denote the resulting estimates by $\hat{TP}_j^{La}$, $\hat{FP}_j^{La}$, $\hat{FN}_j^{La}$. As it will be clear in Section 6 and following, this simple form of smoothing proves almost optimal, which seems to indicate that there is not much to be gained by applying more sophisticated smoothing methods to our problem context.

Note that we apply smoothing in an "on-demand" fashion, that is, we check if the contingency table needs smoothing at all (i.e., if any of $\hat{TP}_j^{ML}$, $\hat{FP}_j^{ML}$, $\hat{FN}_j^{ML}$ is <1) and we smooth it only if this is the case. The reason why we adopt this "on-demand" policy will be especially apparent in Section 7.

4.4. Turning Confidence Scores into Probabilities

We derive the probabilities $P(\omega_k)$ in Equation (3) by assuming that the confidence scores C_{ij} generated by $\hat{\Phi}_j$ can be trusted (i.e., that the higher C_{ij}, the higher the probability that D_{ij} is correct), and by applying to C_{ij} a *generalized logistic function* $f(z) = e^{\sigma z}/(e^{\sigma z}+1)$. This results in

$$\begin{aligned} P(fp_j|D_{ij} = +1) &= 1 - \frac{e^{\sigma C_{ij}}}{e^{\sigma C_{ij}} + 1} \\ P(fn_j|D_{ij} = -1) &= 1 - \frac{e^{\sigma C_{ij}}}{e^{\sigma C_{ij}} + 1}. \end{aligned} \tag{6}$$

The generalized logistic function (see Figure 1) has the effect of monotonically converting scores ranging on $(-\infty, +\infty)$ into real values in the [0.0,1.0] range (therefore, the probabilities of Equation (6) range on [0.0,0.5]). When $C_{ij} = 0$ (this happens when $\hat{\Phi}_j$

has no confidence at all in its own decision D_{ij}), then

$$\begin{aligned} P(tp_j|D_{ij} = +1) = P(fp_j|D_{ij} = +1) = 0.5 \\ P(fn_j|D_{ij} = -1) = P(tn_j|D_{ij} = -1) = 0.5 \end{aligned} \tag{7}$$

that is, the probability of correct classification and the probability of misclassification are identical. Conversely, we have

$$\begin{aligned} \lim_{C_{ij}\to+\infty} P(fp_j|D_{ij} = +1) = 0 \\ \lim_{C_{ij}\to+\infty} P(fn_j|D_{ij} = -1) = 0 \end{aligned} \tag{8}$$

that is, when $\hat{\Phi}_j$ has a very high confidence in its own decision D_{ij}, the probability that D_{ij} is wrong is taken to be close to 0.

The reason why we use a *generalized* version of the logistic function instead of its nonparametric version (which corresponds to the case $\sigma = 1$) is that using this latter within Equation (6) would give rise to a very high number of zero probabilities of misclassification, since the nonparametric logistic function converts every positive number above a certain threshold ($\approx$36) to a number that standard implementations round up to 1 even by working in double precision. By tuning the σ parameter (the *growth rate*), we can tune the speed at which the right-hand side of the sigmoid asymptotically approaches 1, and we can thus tune how evenly Equation (6) distributes the confidence values across the [0.0,0.5] interval.

The process of optimizing σ within Equation (6) is usually called *probability calibration*. How we actually optimize σ is discussed in Section 6.1.

4.5. Ranking by Total Utility

Our function $U_j(d_i, \Omega)$ of Section 4.1 is thus obtained by plugging Equations (4) and (5) into Equation (3). Therefore, we are now in a position to compute, given an automatically classified document d_i and a class c_j, the utility, for the aims of increasing $F_1(\hat{\Phi}_j(Te))$, of manually validating the label D_{ij} attributed to d_i by $\hat{\Phi}_j$.

Now, let us recall from Section 3 that our goal is addressing not just the binary, but the multiclass multilabel TC case, in which binary classification must be accomplished simultaneously for $|\mathcal{C}| \geq 2$ different classes. It might seem sensible to propose ranking, for each $c_j \in \mathcal{C}$, all the automatically labelled documents in Te in decreasing order of their $U_j(d_i, \Omega)$ value. Unfortunately, this would generate $|\mathcal{C}|$ different rankings, and in an operational context it seems implausible to ask a human annotator to scan $|\mathcal{C}|$ different rankings of the same document set (this would mean reading the same document $|\mathcal{C}|$ times in order to validate its labels). As suggested in Esuli and Sebastiani [2009] for AL, it seems instead more plausible to generate a *single* ranking, according to a score $U(d_i, \Omega)$ that is a function of the $|\mathcal{C}|$ different $U_j(d_i, \Omega)$ scores. In such a way, the human annotator will scan this single ranking from the top, validating all the $|\mathcal{C}|$ different labels for d_i before moving on to another document. As the criterion for generating the overall utility score $U(d_i, \Omega)$ we use *total utility*, corresponding to the simple sum

$$U(d_i, \Omega) = \sum_{c_j \in \mathcal{C}} U_j(d_i, \Omega). \tag{9}$$

Our final ranking is thus generated by sorting the test documents in descending order of their $U(d_i, \Omega)$ score.

From the standpoint of computational cost, this technique is $O(|Te| \cdot (|\mathcal{C}| + \log |Te|))$, since the cost of sorting the test documents by their $U(\cdot, \Omega)$ score is $O(|Te| \log |Te|)$, and the cost of computing the $U(\cdot, \Omega)$ score for $|Te|$ documents and $|\mathcal{C}|$ classes is $O(|Te| \cdot |\mathcal{C}|)$.

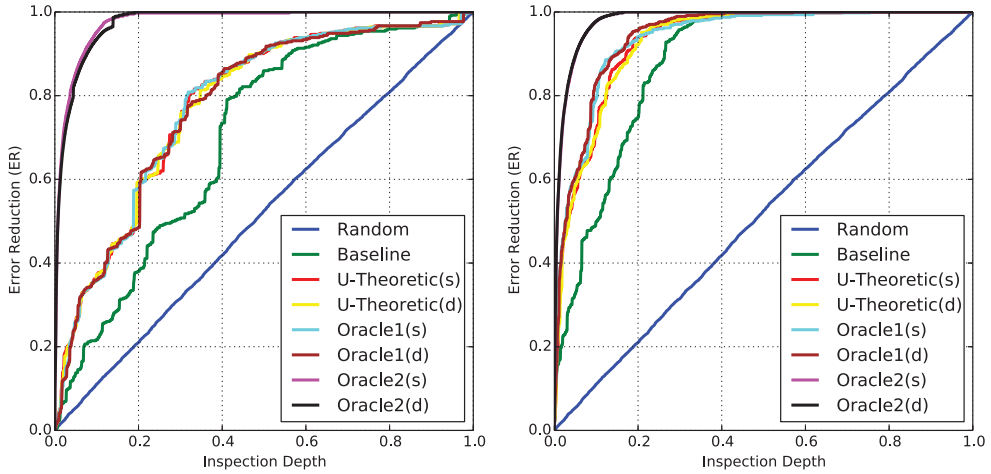

Fig. 2. Error reduction, measured as ER_ρ^M, as a function of validation depth. The dataset is REUTERS-21578, the learners are MP-BOOST (left) and SVMs (right). The Random curve indicates the results of our estimation of the expected ER of the random ranker via a Monte Carlo method with 100 random trials. Higher curves are better.

5. EXPECTED NORMALIZED ERROR REDUCTION

No measures are known from literature for evaluating the effectiveness of a SATC-oriented ranking method ρ. We here propose such a measure, which we call *expected normalized error reduction* (denoted $ENER_\rho$). In this section, we will introduce $ENER_\rho$ in a stepwise fashion.

5.1. Error Reduction At Rank

Let us first introduce the notion of *residual error at rank n* (noted $E_\rho(n)$), defined as the error that is still present in the document set Te after the human annotator has validated the documents at the first n rank positions in the ranking generated by ρ. The value of $E_\rho(0)$ is the initial error generated by the automated classifier, and the value of $E_\rho(|Te|)$ is 0. We assume our measure of error to range on [0,1]; if so, $E_\rho(n)$ ranges on [0,1] too. We will hereafter call n the *validation depth* (or *inspection depth*).

We next define *error reduction at rank n* to be

$$ER_\rho(n) = \frac{E_\rho(0) - E_\rho(n)}{E_\rho(0)} \tag{10}$$

that is, a value in [0,1] that indicates the error reduction obtained by a human annotator who has validated the documents at the first n rank positions in the ranking generated by ρ; 0 stands for no reduction, 1 stands for total elimination of error.

Example plots of the $ER_\rho(n)$ measure are displayed in Figure 2, where different curves represent different ranking methods $\rho', \rho'', \ldots$, and where, for better convenience, the x axis indicates the fraction $n/|Te|$ of the test set that has been validated rather than the number n of validated documents. By definition, all curves start at the origin of the axes (i.e., if the annotator validates zero test documents, no error reduction is obtained) and end at the upper-right corner of the graph (i.e., if the annotator validates all the $|Te|$ test documents, a complete elimination of error is obtained). More convex (i.e., higher) curves represent better strategies, since they indicate that a higher error reduction is achieved for the same amount of manual validation effort.

The reason why we focus on error reduction, instead of the complementary concept of "increase in accuracy," is that error reduction has always the same upper bound

(i.e., 100% reduction), independently of the initial error. In contrast, the increase in accuracy that derives from validating the documents does *not* always have the same upper bound. For instance, if the initial accuracy is 0.5, if we assume that accuracy values range on [0,1] then an increase in accuracy of 100% is indeed possible, while this increase is not possible if the initial accuracy is 0.9. This makes the notion of "increase in accuracy" less immediately interpretable, since different datasets and/or different classifiers give rise to different initial levels of accuracy. So, using "error reduction" instead of "increase in accuracy" makes our curves more immediately interpretable, since error reduction has the same range (i.e., [0,1]) irrespectively of dataset used and/or initial classifier used.

Since (as stated in Section 3) we use F_1 for measuring effectiveness, as a measure of classification error we use $E_1 \equiv (1 - F_1)$, which indeed (as assumed at the beginning of this section) ranges on [0,1]. In order to measure the overall effectiveness of a ranking method across the entire set $\mathcal{C}$ of classes, we compute *macro-averaged* E_1 (noted E_1^M), obtained by computing the class-specific E_1 values and averaging them across the c_j's; from this it derives that $E_1^M = 1 - F_1^M$. By $ER_\rho^M(n)$, we will indicate macro-averaged $ER_\rho(n)$, also obtained by computing the class-specific $ER_\rho(n)$ values and averaging them across the c_j's.

5.2. Normalized Error Reduction At Rank ...

One problem with $ER_\rho(n)$, though, is that the expected $ER_\rho(n)$ value of the random ranker is fairly high[4], since it amounts to $\frac{n}{|Te|}$. The difference between the $ER_\rho(n)$ value of a genuinely engineered ranking method ρ and the expected $ER_\rho(n)$ value of the random ranker is particularly small for high values of n, and is null for $n = |Te|$. This means that it makes sense to factor out the random factor from $ER_\rho(n)$. This leads us to define the *normalized error reduction* of ranking method ρ as $NER_\rho(n) = ER_\rho(n) - \frac{n}{|Te|}$, with macro-averaged $NER_\rho(n)$ obtained as usual and denoted, as usual, by $NER_\rho^M(n)$.

5.3. . . . And Its Expected Value

However, $NER_\rho(n)$ is still unsatisfactory as a measure, since it depends on a specific value of n (which is undesirable, since our human annotator may decide to work down the ranked list as far as she deems suitable). Following Robertson [2008], we assume that the human annotator stops validating the ranked list at exactly rank n with probability $P_s(n)$ (the index s stands for "stoppage"). We can then define the *expected normalized error reduction* of ranking method ρ on a given document set Te as the expected value of $NER_\rho(n)$ according to probability distribution $P_s(n)$, that is,

$$ENER_\rho = \sum_{n=1}^{|Te|} P_s(n) NER_\rho(n) \tag{11}$$

with macro-averaged $ENER_\rho$ indicated, as usual, as $ENER_\rho^M$.

Different probability distributions $P_s(n)$ can be assumed. In order to base the definition of such a distribution on a plausible model of user behaviour, we here make the assumption (along with Moffat and Zobel [2008]) that a human annotator, after validating a document, goes on to validate the next document with probability (or *persistence* [Moffat and Zobel 2008]) p or stops validating with probability $(1 - p)$, so that

[4]That the expected $ER_\rho(n)$ value of the random ranker is $\frac{n}{|Te|}$ is something that we have not tried to formally prove. However, that this holds is supported by intuition *and* is unequivocally shown by Monte Carlo experiments we have run on our datasets; see Figures 2 to 4 for a graphical representation.

$$P_s(n) = \begin{cases} p^{n-1}(1-p) & \text{if } n \in \{1, \ldots, |Te| - 1\} \\ p^{n-1} & \text{if } n = |Te|. \end{cases} \quad (12)$$

It can be shown that, for a sufficiently large value of $|Te|$, $\sum_{n=1}^{|Te|} n \cdot P_s(n)$ (the expected number of documents that the human annotator will validate as a function of p) asymptotically tends to $\frac{1}{1-p}$. The value $\xi = \frac{1}{|Te|(1-p)}$ thus denotes the expected *fraction* of the test set that the human annotator will validate as a function of p.

Using this distribution in practice entails the need of determining a realistic value for p. A value $p = 0$ corresponds to a situation in which the human annotator only validates the top-ranked document, while $p = 1$ indicates a human annotator who validates each document in the ranked list. Unlike in ad hoc search, we think that in a SATC context it would be unrealistic to take a value for p as given irrespective of the size of Te. In fact, given a desired level of error reduction, when $|Te|$ is large the human annotators need to be more persistent (i.e., characterized by higher p) than when $|Te|$ is small. Therefore, instead of assuming a predetermined value of p we assume a predetermined value of ξ, and derive the value of p from the equation $\xi = \frac{1}{|Te|(1-p)}$. For example, in a certain application we might assume $\xi = 0.20$ (i.e., assume that the average human annotator validates 20% of the test set). In this case, if $|Te| = 1000$, then $p = 1 - \frac{1}{0.20 \times 1000} = 0.9950$, while if $|Te| = 10{,}000$, then $p = 1 - \frac{1}{0.20 \times 10000} = 0.9995$. In the experiments of Section 6, we will test all values of p corresponding to values of ξ in $\{0.05, 0.10, 0.20\}$.

Note that the values of $ENER_\rho$ are bound above by 1, but a value of 1 is not attainable. In fact, even the "perfect ranker" (i.e., the ranking method that top-ranks all misclassified documents, noted *Perf*) cannot attain an $ENER_\rho$ value of 1, since in order to achieve total error elimination all the misclassified documents need to be validated anyway, one by one, which means that the only condition in which $ENER_{Perf}$ might equal 1 is when there is just 1 misclassified document. We do not try to normalize $ENER_\rho$ by the value of $ENER_{Perf}$ since $ENER_{Perf}$ cannot be characterized analytically, and depends on the actual labels in the test set.

6. EXPERIMENTS

We have now fully specified (Section 4) a method for performing SATC-oriented ranking and (Section 5) a measure for evaluating the quality of the produced rankings, so we are now in a position to test the effectiveness of our proposed method. In Sections 6.1 to 6.5, we will describe our experimental setting, while in Section 6.6 we will report and discuss the actual results of these experiments.

6.1. Experimental Protocol

Let Ω be a dataset partitioned into a training set Tr and a test set Te. In each experiment reported in this article, we adopt the following experimental protocol:

(1) For each $c_j \in \mathcal{C}$
  (a) train classifier $\hat{\Phi}_j$ on Tr and classify Te by means of $\hat{\Phi}_j$;
  (b) run k-fold cross-validation on Tr, thereby
    i computing TP_j^{Tr}, FP_j^{Tr}, and FN_j^{Tr};
    ii optimizing the σ parameter of Equation (6) (see Section 6.2 next for the actual optimization method used).
(2) For every ranking policy ρ tested
  (a) rank Te according to ρ;
  (b) scan the ranked list from the top, correcting possible misclassifications and computing the resulting values of $ENER_\rho^M$ for different values of ξ.

For Step 1(b), we have used $k = 10$; we think this value guarantees a good tradeoff between the accuracy of the parameter estimates (which tends to increase with k) and the cost of computing these estimates (which also increases with k).

6.2. Probability Calibration

We optimize the σ parameter by picking the value of σ that minimizes the average (across the $c_j \in \mathcal{C}$) absolute value of the difference between Pos_j^{Tr}, the number of positive training examples of class c_j, and $\mathrm{E}[Pos_j^{Tr}]$, the *expected* number of such examples as resulting from the probabilities of membership in c_j computed in the k-fold crossvalidation. That is, we pool together all the training documents classified in the k-fold crossvalidation phase, and then we pick

$$\begin{aligned}
&\arg\min_{\sigma} \frac{1}{|\mathcal{C}|} \sum_{c_j \in \mathcal{C}} \left| Pos_j^{Tr} - \mathrm{E}\left[Pos_j^{Tr}\right] \right| \\
&= \arg\min_{\sigma} \frac{1}{|\mathcal{C}|} \sum_{c_j \in \mathcal{C}} \left| Pos_j^{Tr} - \sum_{d_i \in Tr} P(c_j|d_i) \right| \\
&= \arg\min_{\sigma} \frac{1}{|\mathcal{C}|} \sum_{c_j \in \mathcal{C}} \left| Pos_j^{Tr} - \sum_{d_i \in Tr} \frac{e^{\sigma \hat{\Phi}_j(d_i)}}{e^{\sigma \hat{\Phi}_j(d_i)} + 1} \right|.
\end{aligned} \tag{13}$$

This method is a much faster calibration method than the traditional method of picking the value of σ that has performed best in k-fold crossvalidation[5]. In fact, unlike the latter, it does not depend on the ranking method ρ. Therefore, this method spares us from the need of ranking the training set several times, that is, once for each combination of a tested value of σ and a ranking method ρ.

6.3. Learning Algorithms

As our first learning algorithm for generating our classifiers $\hat{\Phi}_j$, we use a boosting-based learner called MP-Boost [Esuli et al. 2006]. Boosting-based methods have shown very good performance across many learning tasks and, at the same time, have strong justifications from computational learning theory. MP-Boost is a variant of AdaBoost.MH [Schapire and Singer 2000] optimized for multilabel settings, which has been shown in Esuli et al. [2006] to obtain considerable effectiveness improvements with respect to AdaBoost.MH. In all our experiments, we set the S parameter of MP-Boost (representing the number of boosting iterations) to 1000.

As the second learning algorithm, we use SVMs. We use the implementation from the freely available LibSvm library[6], with a linear kernel and parameters at their default values.

In all the experiments discussed in this article, stop words have been removed, punctuation has been removed, all letters have been converted to lowercase, numbers have been removed, and stemming has been performed by means of Porter's stemmer. Word stems are thus our indexing units. Since MP-Boost requires binary input, only their presence/absence in the document is recorded, and no weighting is performed. Documents are instead weighted (by standard cosine-normalized $tfidf$) for the SVMs experiments.

[5]This method is sometimes called *Platt calibration* (see e.g., Niculescu-Mizil and Caruana [2005]), due to its use in Platt [2000]. However, the method was in use well before Platt's article (see e.g., Ittner et al. [1995, Section 2.3]).

[6]http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

Table I. Characteristics of the Test Collections Used

Dataset	$\|\mathcal{T}\|$	$\|Tr\|$	$\|Te\|$	$\|\mathcal{C}\|$	ACD	E_1^M		E_1^μ	
						MP-B	SVMs	MP-B	SVMs
REUTERS-21578	1	9603	3299	115	1.135	.392	.473	.152	.140
REUTERS-21578/10	10	9603	330	115	1.135	.194	.199	.151	.130
REUTERS-21578/100	100	9603	33	115	1.135	.050	.049	.149	.140
OHSUMED	1	183229	50216	97	0.132	.553	.577	.389	.324
OHSUMED-S	1	12358	3584	97	1.851	.520	.522	.286	.244

Notes: From left to right, we report the number of test sets $|\mathcal{T}|$ (Column 2) and, for each test set, the number of training documents $|Tr|$ (3), the number of test documents $|Te|$ (4), the number of classes $|\mathcal{C}|$ (5), and the average number of classes per test document ACD (6). Columns 7–10 report the initial error (both E_1^M and E_1^μ) generated by the MP-BOOST and SVMs classifiers.

6.4. Datasets

Our first dataset is the REUTERS-21578 corpus. It consists of a set of 12,902 news stories, partitioned (according to the standard "ModApté" split we have adopted) into a training set of 9603 documents and a test set of 3299 documents. The documents are labelled by 118 categories; the average number of categories per document is 1.08, ranging from a minimum of 0 to a maximum of 16; the number of positive examples per class ranges from a minimum of 1 to a maximum of 3964. In our experiments, we have restricted our attention to the 115 categories with at least one positive training example. This dataset is publicly available[7] and is probably the most widely used benchmark in TC research; this fact allows other researchers to easily replicate the results of our experiments.

Another dataset we have used is OHSUMED [Hersh et al. 1994], a test collection consisting of a set of 348,566 MEDLINE references spanning the years from 1987 to 1991. Each entry consists of summary information relative to a paper published on one of 270 medical journals. The available fields are title, abstract, MeSH indexing terms, author, source, and publication type. Not all the entries contain abstract and MeSH indexing terms. In our experiments, we have scrupulously followed the experimental setup presented in Lewis et al. [1996]. In particular, (i) we have used for our experiments only the 233,445 entries with both abstract and MeSH indexing terms; (ii) we have used the entries relative to years 1987 to 1990 (183,229 documents) as the training set and those relative to year 1991 (50,216 documents) as the test set; (iii) as the categories on which to perform our experiments we have used the *main heading* MeSH index terms assigned to the entries. Concerning the latter point, we have restricted our experiments to the 97 MeSH index terms that belong to the *Heart Disease* (HD) subtree of the MeSH tree, and that have at least one positive training example. This is the only point in which we deviate from Lewis et al. [1996], which experiments only on the 77 most frequent MeSH index terms of the HD subtree.

The main characteristics of our datasets, and of three variants (called REUTERS-21578/10, REUTERS-21578/100, and OHSUMED-S) that will be discussed in Section 6.6, are conveniently summarized in Table I.

6.5. Lower Bounds and Upper Bounds

As the baseline for our experiments, we use the confidence-based strategy discussed in Section 1, which corresponds to using our utility-theoretic method with both $G(fp)$ and $G(fn)$ set to 1.

While the confidence-based method will act as our lower bound, we have also run "oracle-based" methods aimed at identifying upper bounds for the effectiveness of our

[7]http://www.daviddlewis.com/resources/testcollections/~reuters21578/.

utility-theoretic method, that is, at assessing the effectiveness of "idealized" (albeit nonrealistic) systems at our task.

The first such method (dubbed Oracle1) works by "peeking" at the actual values of TP_j, FP_j, FN_j in Te, using them in the computation of $G(d_i, fp_j)$ and $G(d_i, fn_j)$, and applying our utility-theoretic method as usual. Oracle1 thus indicates how our method would behave were it able to "perfectly" estimate TP_j, FP_j, and FN_j. The difference in effectiveness between Oracle1 and our method will thus be due to (i) the performance of the method adopted for smoothing contingency tables, and (ii) possible differences between the distribution of the documents across the contingency table cells in the training and in the test set.

In the second such method (Oracle2), we instead peek *at the true labels* of the documents in Te, which means that we will be able to (a) use the actual values of TP_j, FP_j, FN_j in the computation of $G(d_i, fp_j)$ and $G(d_i, fn_j)$ (as in Oracle1), *and* (b) replace the probabilities in Equation (3) with the true binary values (i.e., replacing $P(x)$ with 1 if x is true and 0 if x is false), after which we apply our utility-based ranking method as usual. The difference in effectiveness between Oracle2 and our method will be due to factors (i) and (ii) already mentioned for Oracle1 *and* to our method's (obvious) inability to perfectly predict whether a document was classified correctly or not.

6.6. Results and Discussion

The results of our experiments are given in Table II, where we present the results of running, for each of two learners (MP-Boost and SVMs) and five datasets (Reuters-21578, OHSUMED, and three variants of them – called Reuters-21578/10, Reuters-21578/100, OHSUMED-S – that we will introduce in Sections 6.6.2, 6.6.3, 6.6.4), our utility-theoretic method against the three methods discussed in Section 6.5. In Table II our method, Oracle1 and Oracle2 are actually indicated as U-Theoretic(s), Oracle1(s), and Oracle2(s), to distinguish them from variants (indicated as U-Theoretic(d), Oracle1(d), and Oracle2(d)) that will be described in Section 7. Table II presents $ENER_\rho^M(\xi)$ values for three representative values of ξ, that is, 0.05, 0.10, and 0.20.

For each of two learners and five datasets, and for each pairwise combination of all the methods discussed (including those we will discuss in Section 7), we have run a paired t-test with $ENER_\rho^M(0.10)$ as the evaluation measure and 0.05 as the significance level, in order to determine whether the difference in performance between the two methods is statistically significant. The results of such tests are reported in Table III.

6.6.1. Mid-Sized Test Sets. Figure 2 plots the results, in terms of $ER_\rho^M(n)$, of our experiments with the MP-Boost and SVM learners on the Reuters-21578 dataset. The results of these experiments in terms of $ENER_\rho^M$ as a function of the chosen value of ξ are instead reported in Table II. The optimal value of σ returned by the k-fold crossvalidation phase is 0.554 for MP-Boost and 7.096 for SVMs; these values, sharply different from 1 and from each other, clearly show the advantage of converting confidence scores into probabilities via a *generalized* logistic function.

The first insight we can draw from these results is that our U-Theoretic(s) method outperforms Baseline in a very substantial way (the paired t-test – see Table III – indicates that this difference is statistically significant). This can be appreciated both from the plots of Figure 2, in which the red curve (corresponding to U-Theoretic(s)) is markedly higher than the green curve (corresponding to Baseline), and from Table II. In the latter, for $\xi = 0.10$ (corresponding to $p = 0.996$) our method obtains relative improvements over Baseline of +109% (MP-Boost) and +51% (SVMs); for $\xi = 0.20$ the improvements, while not as high as for $\xi = 0.10$, are still sizeable (+84% for MP-Boost and +34% for SVMs), while for $\xi = 0.05$ the improvements are even higher than for $\xi = 0.10$ (+128% for MP-Boost and +69% for SVMs).

Table II. Results of Various Ranking Methods, Applied to Two Learning Algorithms and Several Test Collections, in Terms of $ENER_\rho^M(\xi)$, for $\xi \in \{0.05, 0.10, 0.20\}$. Improvements Listed for the Various Methods are Relative to the Baseline

		MP-BOOST						SVMs					
		$\xi = 0.05$		$\xi = 0.10$		$\xi = 0.20$		$\xi = 0.05$		$\xi = 0.10$		$\xi = 0.20$	
REUTERS-21578	Baseline	0.071		0.108		0.152		0.262		0.352		0.420	
	U-Theoretic(s)	0.163	(+128%)	0.226	(+109%)	0.280	(+84%)	0.442	(+69%)	0.531	(+51%)	0.562	(+34%)
	U-Theoretic(d)	0.160	(+124%)	0.224	(+107%)	0.279	(+84%)	0.431	(+65%)	0.523	(+49%)	0.557	(+33%)
	Oracle1(s)	0.155	(+117%)	0.222	(+106%)	0.280	(+84%)	0.477	(+82%)	0.563	(+60%)	0.586	(+40%)
	Oracle1(d)	0.152	(+113%)	0.219	(+103%)	0.275	(+81%)	0.476	(+82%)	0.567	(+61%)	0.592	(+41%)
	Oracle2(s)	0.693	(+869%)	0.738	(+583%)	0.707	(+365%)	0.719	(+174%)	0.760	(+116%)	0.723	(+72%)
	Oracle2(d)	0.677	(+847%)	0.725	(+571%)	0.699	(+360%)	0.723	(+176%)	0.763	(+117%)	0.724	(+72%)
REUTERS-21578/10	Baseline	0.063		0.097		0.135		0.243		0.322		0.383	
	U-Theoretic(s)	0.145	(+131%)	0.203	(+110%)	0.245	(+81%)	0.330	(+36%)	0.415	(+29%)	0.465	(+21%)
	U-Theoretic(d)	0.139	(+121%)	0.198	(+105%)	0.239	(+77%)	0.335	(+38%)	0.420	(+30%)	0.470	(+23%)
	Oracle1(s)	0.159	(+153%)	0.205	(+112%)	0.243	(+80%)	0.392	(+61%)	0.482	(+50%)	0.522	(+36%)
	Oracle1(d)	0.158	(+152%)	0.212	(+119%)	0.255	(+89%)	0.394	(+62%)	0.488	(+52%)	0.531	(+39%)
	Oracle2(s)	0.555	(+784%)	0.643	(+566%)	0.648	(+380%)	0.596	(+145%)	0.676	(+110%)	0.672	(+75%)
	Oracle2(d)	0.558	(+789%)	0.648	(+571%)	0.654	(+384%)	0.599	(+147%)	0.679	(+111%)	0.675	(+76%)
REUTERS-21578/100	Baseline	0.069		0.121		0.164		0.226		0.302		0.364	
	U-Theoretic(s)	0.118	(+71%)	0.172	(+42%)	0.215	(+31%)	0.291	(+29%)	0.365	(+21%)	0.416	(+14%)
	U-Theoretic(d)	0.119	(+72%)	0.176	(+45%)	0.217	(+32%)	0.289	(+28%)	0.367	(+22%)	0.419	(+15%)
	Oracle1(s)	0.192	(+178%)	0.247	(+104%)	0.281	(+71%)	0.318	(+41%)	0.422	(+40%)	0.479	(+32%)
	Oracle1(d)	0.197	(+185%)	0.266	(+120%)	0.318	(+94%)	0.318	(+41%)	0.427	(+41%)	0.489	(+34%)
	Oracle2(s)	0.429	(+521%)	0.537	(+344%)	0.575	(+251%)	0.458	(+103%)	0.568	(+88%)	0.600	(+65%)
	Oracle2(d)	0.429	(+521%)	0.537	(+344%)	0.576	(+251%)	0.458	(+103%)	0.569	(+88%)	0.601	(+65%)
OHSUMED	Baseline	0.385		0.479		0.512		0.526		0.630		0.644	
	U-Theoretic(s)	0.442	(+15%)	0.529	(+10%)	0.549	(+7%)	0.623	(+18%)	0.685	(+9%)	0.666	(+3%)
	U-Theoretic(d)	0.443	(+15%)	0.531	(+11%)	0.550	(+7%)	0.618	(+17%)	0.676	(+7%)	0.655	(+2%)
	Oracle1(s)	0.445	(+16%)	0.530	(+11%)	0.549	(+7%)	0.639	(+21%)	0.687	(+9%)	0.657	(+2%)
	Oracle1(d)	0.449	(+17%)	0.532	(+11%)	0.550	(+7%)	0.617	(+17%)	0.659	(+5%)	0.636	(-1%)
	Oracle2(s)	0.838	(+118%)	0.839	(+75%)	0.769	(+50%)	0.864	(+64%)	0.854	(+36%)	0.778	(+21%)
	Oracle2(d)	0.758	(+97%)	0.762	(+59%)	0.700	(+37%)	0.795	(+51%)	0.787	(+25%)	0.721	(+12%)
OHSUMED-S	Baseline	0.021		0.025		0.026		0.075		0.124		0.164	
	U-Theoretic(s)	0.087	(+323%)	0.118	(+374%)	0.132	(+402%)	0.212	(+184%)	0.282	(+127%)	0.323	(+97%)
	U-Theoretic(d)	0.088	(+329%)	0.118	(+374%)	0.132	(+402%)	0.210	(+182%)	0.280	(+126%)	0.321	(+96%)
	Oracle1(s)	0.091	(+343%)	0.117	(+370%)	0.125	(+375%)	0.272	(+265%)	0.334	(+169%)	0.352	(+115%)
	Oracle1(d)	0.094	(+358%)	0.119	(+378%)	0.128	(+387%)	0.301	(+303%)	0.363	(+193%)	0.380	(+132%)
	Oracle2(s)	0.481	(+2246%)	0.554	(+2125%)	0.572	(+2075%)	0.511	(+585%)	0.589	(+375%)	0.603	(+268%)
	Oracle2(d)	0.450	(+2095%)	0.498	(+1900%)	0.496	(+1786%)	0.487	(+553%)	0.540	(+335%)	0.536	(+227%)

A second insight is that, surprisingly, our method hardly differs in terms of performance from Oracle1(s). The two curves can be barely distinguished in Figure 2, and in terms of $ENER_\rho^M$ Oracle1(s) is even slightly outperformed, in the MP-BOOST experiments, by U-Theoretic(s) (e.g., 0.226 vs. .222 for $\xi = 0.10$); the paired t-test (see Table III) indicates that the difference between the two methods is not statistically significant. This shows that (at least judging from these experiments) Laplace smoothing is nearly optimal, and there is likely not much we can gain from applying alternative, more sophisticated smoothing methods. This is sharply different from what happens in language modelling, where Laplace smoothing has been shown to be an underperformer [Gale and Church 1994]. The fact that with MP-BOOST our method slightly

Table III. Statistical Significance Results Obtained for the Two Learners (MP-Boost and SVMs) via a Paired t-test with $ENER_\rho^M(0.10)$ as the Evaluation Measure and 0.05 as the Significance Level

		Baseline	U-Theoretic(s)	U-Theoretic(d)	Oracle1(s)	Oracle1(d)	Oracle2(s)	Oracle2(d)
MP-Boost	Baseline	– – – –	YYYYY	YYYYY	YYYYY	YYYYY	YYYYY	YYYYY
	U-Theoretic(s)	YYYYY	– – – –	NYNNN	NNYNN	YNYNN	YYYYY	YYYYY
	U-Theoretic(d)	YYYYY	NYNNN	– – – –	NNYNN	NNYNN	YYYYY	YYYYY
	Oracle1(s)	YYYYY	NNYNN	NNYNN	– – – –	NNYNN	YYYYY	YYYYY
	Oracle1(d)	YYYYY	YNYNN	NNYNN	NNYNN	– – – –	YYYYY	YYYYY
	Oracle2(s)	YYYYY	YYYYY	YYYYY	YYYYY	YYYYY	– – – –	NNNYN
	Oracle2(d)	YYYYY	YYYYY	YYYYY	YYYYY	YYYYY	NNNYN	– – – –
SVMs	Baseline	– – – –	YYYYY	YYYYY	YYYYY	YYYYY	YYYYY	YYYYY
	U-Theoretic(s)	YYYYY	– – – –	YNNYN	YYYNY	YYYNY	YYYYY	YYYYY
	U-Theoretic(d)	YYYYY	YNNYN	– – – –	YYYNY	YYYNY	YYYYY	YYYYY
	Oracle1(s)	YYYYY	YYYNY	YYYNY	– – – –	NNYNY	YYYYY	YYYYY
	Oracle1(d)	YYYYY	YYYNY	YYYNY	NNYNY	– – – –	YYYYY	YYYYY
	Oracle2(s)	YYYYY	YYYYY	YYYYY	YYYYY	YYYYY	– – – –	YYNNN
	Oracle2(d)	YYYYY	YYYYY	YYYYY	YYYYY	YYYYY	YYNNN	– – – –

Notes: "Y" means that there is a statistically significant difference between the two methods, while "N" means there is not; each 5-tuple of Y's and N's indicates this for the five datasets studied in this article (REUTERS-21578, REUTERS-21578/10, REUTERS-21578/100, OHSUMED, OHSUMED-S, in this order).

(and strangely) outperforms Oracle1(s) is probably due to accidental, "serendipitous" interactions between the probability estimation component (Equation 6) and the contingency cell estimation component of Section 4.3; in fact, the paired t-test indicates (see Table III) that this difference is not statistically significant.

A third interesting fact is that error reduction is markedly better in the SVM experiments than in the MP-BOOST experiments. This is evident from the fact that the Figure 2 curves for SVMs are much more convex (i.e., are higher) and are closer to the optimum (i.e., closer to the Oracle2(s) curve) than the corresponding Figure 2 curves for MP-BOOST. This fact is also evident from the numerical results reported in Table II where, with U-Theoretic(s), SVMs obtain $ENER_\rho^M(0.10) = 0.531$, which is +134% better than the $ENER_\rho^M(0.10) = 0.226$ result obtained by MP-BOOST (similar improvements can be observed for the other methods and for the other values of ξ). This provides a striking contrast with the *classification accuracy* results reported in Figure 1 where, on the same dataset, MP-BOOST ($E_1^M = 0.392$)substantially outperformed SVMs ($E_1^M = 0.473$). It is easy to conjecture that, even if MP-BOOST yields higher classification accuracy, it generates less reliable (calibrated) confidence scores, that is, it generates confidence scores that correlate with the ground truth worse than the SVM-generated scores.

The rates of improvement of U-Theoretic(s) over the baseline are instead much higher for MP-BOOST than for SVMs (e.g., for $\xi = 0.10$ these are +109% and +51%, respectively). (The same goes for the improvements of Oracle1(s) over the baseline.) This is likely due to the fact that, as observed earlier, the absolute values of $ENER_\rho^M(\xi)$ obtained by the baseline are much higher for SVMs than for MP-BOOST for all methods, so the margins of improvement with respect to the baseline are smaller for SVMs than for MP-BOOST.

6.6.2. Small Test Sets. We have also run a batch of experiments aimed at assessing how the methods fare when ranking test sets much smaller than REUTERS-21578. This may

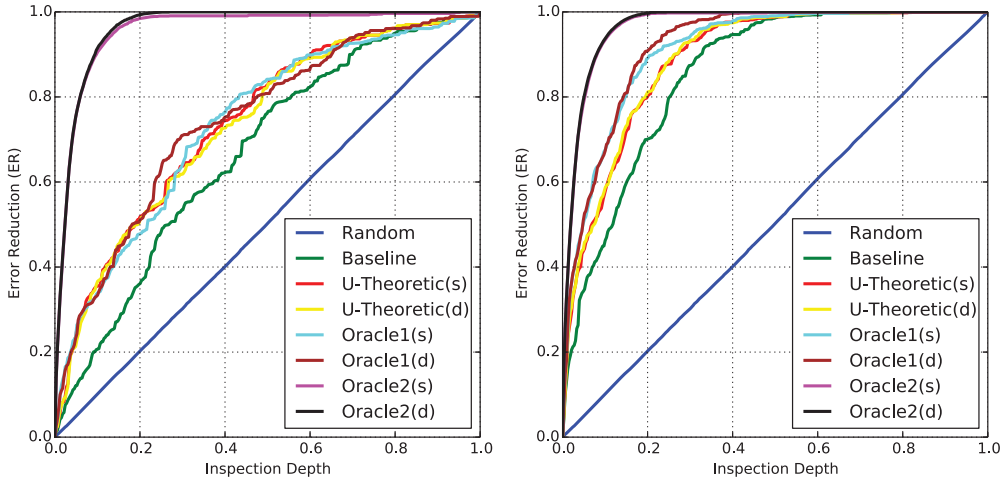

Fig. 3. Results obtained by (a) splitting the REUTERS-21578 test set into 10 random, equally-sized parts, (b) running the analogous experiments of Figure 2 independently on each part, and (c) averaging the results across the 10 parts. The learners used are MP-BOOST (left) and SVMs (right).

be *more* challenging than ranking larger sets since, when the test set is small, Laplace smoothing (i) can seriously perturb the relative proportions among the cell counts, which can generate poor estimates of $G(d_i, fp_j)$ and $G(d_i, fn_j)$, and (ii) is performed for more classes, since (as discussed at the end of Section 4.3) we smooth "on demand" only, and since the likelihood that $\hat{TP}_j^{ML}, \hat{FP}_j^{ML}, \hat{FN}_j^{ML}$ are smaller than 1 is higher with small test sets. This is also a realistic setting since, if a set of unlabelled documents is small, it is likely that validating a portion of it that can lead to sizeable enough effectiveness improvements is feasible from an economic point of view.

Rather than choosing a completely different dataset, we generate 10 new test sets by randomly splitting the REUTERS-21578 test set in 10 equally sized parts (about 330 documents each). In our experiments, we run each ranking method on each such part individually and average the results across the 10 parts. We call this experimental scenario REUTERS-21578/10. This allows us to study the effects of test set size on our methods in a more controlled way than if we had picked a completely different dataset, since test set size is the only difference with respect to the previous REUTERS-21578 experiments.

The results displayed in Figure 3 allow us to visually appreciate that U-Theoretic(s) substantially outperforms Baseline also in this context. This can be seen also from Table II: for $\xi = 0.10$, the relative improvement over Baseline is $+110\%$ for MP-BOOST and $+30\%$ for SVMs, and similarly substantial improvements are obtained for the two other values of ξ tested.

Incidentally, note that the REUTERS-21578/10 experiments model an application scenario in which a set of automatically labelled documents is split (e.g., to achieve faster throughput) among 10 human annotators, each one entrusted with validating a part of the set. In this case, each annotator is presented with a ranking of her own document subset, and works exclusively on it[8].

[8]Actually, if we did have k annotators available, the best strategy would be to generate the k rankings in a "round robin" fashion, that is, by allotting to annotator i the documents ranked (in the global ranking) at the positions r such that $(r \bmod k) = i$. This splitting method would guarantee that only the most promising documents are validated by the annotators.

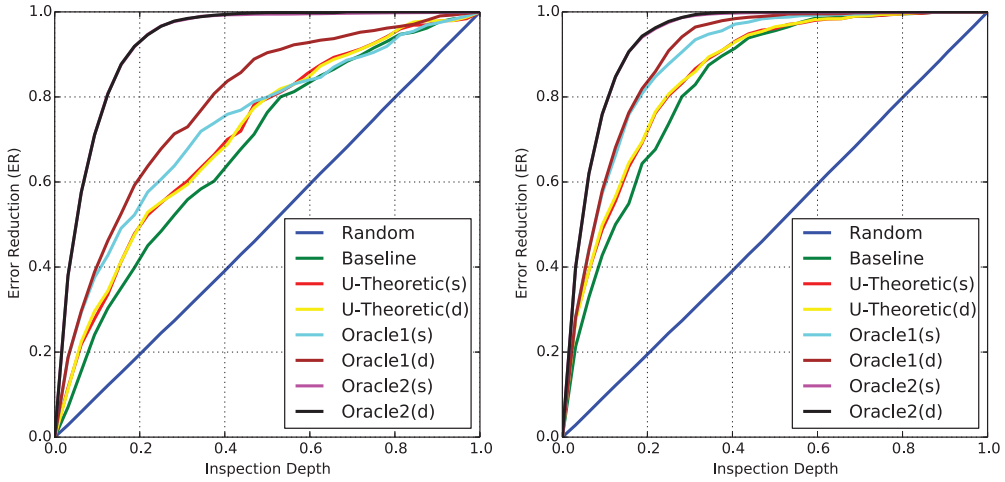

Fig. 4. Same as Figure 3 but with REUTERS-21578/100 in place of REUTERS-21578/10. The learners used are MP-BOOST (left) and SVMs (right).

6.6.3. Tiny Test Sets. In further experiments that we have run, we have split the REUTERS-21578 test set even further, that is, into 100 equally sized parts of about 33 documents each, so as to test the performance of Laplace smoothing methods in even more challenging conditions. We call this experimental scenario REUTERS-21578/100. From an application point of view, this is a less interesting scenario than the two previously discussed ones, since applying a ranking method to a set of 33 documents only is of debatable utility, given that a human annotator confronted with the task of validating just 33 documents can arguably check them all without any need for ranking. The goal of these experiments is thus checking whether our method can perform well even in extreme, albeit scarcely realistic, conditions.

The detailed $ER_\rho^M(n)$ plots for this REUTERS-21578/100 scenario are presented in Figure 4, while the $ENER_\rho^M$ results are reported in Table II[9]. U-Theoretic(s) still outperforms Baseline, with a relative improvement of +42% with MP-BOOST and +21% with SVMs with $\xi = 0.10$, corresponding to $p = 0.696$; qualitatively similar improvements are obtained with the other tested values of ξ.

Note that in these experiments, unlike in those performed on the full REUTERS-21578, the Oracle1(s) method proves to be markedly superior to U-Theoretic(s) (e.g., 0.247 vs. 0.172 in terms of $ENER_\rho^M(0.10)$ with MP-BOOST, and similarly for other values of ξ and for the SVM learner); unlike in the previous two datasets, the difference between the two methods turns out to be statistically significant. The reason is that, for a smaller test set, (a) distribution drift is higher, (b) "smoothing on demand" is invoked more frequently (because the likelihood that contingency table cells have a value ≤ 1 is higher), and (c) when smoothing is indeed applied, the distribution across the cells of the contingency table is perturbed more strongly.

Note also that the $ER_\rho^M(n)$ curves are smoother than the analogous curves for the full REUTERS-21578 and, although to a lesser extent, those for REUTERS-21578/10. This is due to the fact that the curves in Figure 4 result from averages across 100 different

[9]From the next experiments onwards, for reasons of space we will not include the full plots in the style of Figures 2 to 4, and will only report $ENER_\rho^M$ results.

experiments, and the increase brought about at rank n is actually the average of the increases brought about at rank n in the 100 experiments.

6.6.4. Large Test Sets. While in the previous sections we have discussed experiments on mid-sized to small (or very small) datasets, we now look at larger datasets such as OHSUMED. The OHSUMED results in Table II confirm the quality of U-Theoretic(s), which outperforms the purely confidence-based baseline by +10% (MP-BOOST) and +9% (SVMs) in terms of $ENER_{\rho}^{M}(0.10)$; qualitatively similar improvements are obtained for the other two values of ξ studied.

The OHSUMED collection is characterized by the presence of an unusually large number (93.1% of the entire lot) of unlabelled documents (i.e., documents, that are negative examples for all $c_j \in \mathcal{C}$) that originally belonged to other subtrees of the MeSH tree. Since such a large percentage is unnatural, we have generated (and also used in our experiments) a variant of OHSUMED (called OHSUMED-S) by removing all the unlabelled documents from both the training set and the test set.

As illustrated in Table II, on OHSUMED-S U-Theoretic(s) outperforms the confidence-based baseline by a very large margin (+374% with MP-BOOST and +127% with SVMs for $\xi = 0.10$, with qualitatively similar results for the other two tested values of ξ).

6.6.5. Discussion. In sum, the results discussed from Section 6.6.1 to the present one have unequivocally shown that U-Theoretic(s) outperforms the confidence-based baseline, usually by a large or very large margin, for all the five tested datasets and for both tested learners.

Note that, for all five datasets and for both learners, the improvements of the utility-theoretic methods over Baseline are larger for smaller values of ξ. This indicates that the difference between the two methods is larger for smaller validation depths, that is, where using the utility-theoretic method pays off the most is at the very top of the ranking. This is an important feature of this method, since it means that all human annotators, be they persistent or not (i.e., independently of the depth at which they validate), are going to benefit from this approach.

7. AN IMPROVED, "DYNAMIC" RANKING FUNCTION FOR SATC

The utility-theoretic method discussed in Section 4 is reasonable but, in principle, suboptimal, and its suboptimality derives from its "static" nature. To see this, assume that the system has ranked the test documents according to the strategy mentioned earlier, that the human annotator has started from the top of the list and validated the labels of document d_i, that she has found out that its label assignment for class c_j is a false negative, and that she has corrected it, thus bringing about an increase in F_1 equivalent to

$$\frac{2(TP_j + 1)}{2(TP_j + 1) + FP_j + (FN_j - 1)} - \frac{2TP_j}{2TP_j + FP_j + FN_j}. \tag{14}$$

Following this correction, the value of FN_j is decreased by 1 and the value of TP_j is increased by 1. This means that, when another false negative for c_j is found and corrected, the value of (14) has changed. In other words, the improvement in F_1 due to the validation of a false negative is not constant through the validation process. Of course, similar considerations apply for false positives.

This suggests redefining the validation gains defined in Equations (4) and (5) as

$$G(d_i, fp_j) = \frac{2TP_j}{2TP_j + (FP_j - 1) + FN_j} - \frac{2TP_j}{2TP_j + FP_j + FN_j}$$
$$G(d_i, fn_j) = \frac{2(TP_j + 1)}{2(TP_j + 1) + FP_j + (FN_j - 1)} - \frac{2TP_j}{2TP_j + FP_j + FN_j}. \quad (15)$$

To see the novelty introduced with respect to Equation (15), in the following we will discuss the case of false negatives; the case of false positives is completely analogous. The difference between Equations (5) and (15) is that the former equates $G(d_i, fn_j)$ with the increase in $F_1(\hat{\Phi}_j(Te))$ that would derive by correcting *all of the documents in* FN_j *divided by their number*, while the latter equates $G(d_i, fn_j)$ with the increase in $F_1(\hat{\Phi}_j(Te))$ that would derive by correcting *the next document in* FN_j. In other words, we might say that Equation (5) enforces the notion of *average gain*, while Equation (15) enforces the notion of *pointwise gain*[10]. The two versions return different values of $G(d_i, fn_j)$: as the following example shows, it is immediate to verify that if FN_j contains more than one document, the validation gains $G(d_i, fn_j)$ that derive by correcting different documents are the same (by definition) if we use Equation (5) but are not the same if we use Equation (15).

Example 7.1. Suppose we have classified a set of 100 documents according to class c_j, and that the classification is such that $TP_j = 10$, $FN_j = 20$, $FP_j = 30$, and $TN_j = 40$. According to Equation (5), $G(d_i, fn_j)$ evaluates to $\approx$0.0190 for each false negative corrected. Instead, according to Equation (15), $G(d_i, fn_j)$ evaluates to $\approx$0.0241 for the first false negative corrected, $\approx$0.0235 for the second, $\approx$0.0228 for the third, ..., down to $\approx$0.0147 for the twentieth.

Given this new definition, we may implement a dynamic strategy in which, instead of plainly sorting the test documents in descending order of their $U(d_i, \Omega)$ score, after each correction is made we update $\hat{TP}_j^{La}$, $\hat{FP}_j^{La}$, $\hat{FN}_j^{La}$ by adding and subtracting 1 where appropriate, we recompute $G(d_i, fp_j)$, $G(d_i, fn_j)$ and $U(d_i, \Omega)$, and we use the newly computed $U(d_i, \Omega)$ values when selecting the document that should be presented next to the human annotator. In detail, the following steps are iteratively performed:

(1) for all classes $c_j \in \mathcal{C}$, compute $G(d_i, fp_j)$ and/or $G(d_i, fn_j)$ using Equations (15);
(2) if the human annotator does not want to stop validating documents, then identify the document $d_{max} \equiv \arg\max_{d_i \in Te} U(d_i, \Omega)$ for which total utility is maximised;
(3) remove d_{max} from Te;
(4) for all $c_j \in \mathcal{C}$, have the human annotator check the label attached by $\hat{\Phi}_j$ to d_{max}; if all these labels are correct go to Step 2; else, for all classes $c_j \in \mathcal{C}$ for which the label attached by $\hat{\Phi}_j$ to d_{max} is incorrect:
   (a) have the human annotator correct the label;
   (b) if d_{max} was a false positive for c_j, decrease $\hat{FP}_j^{La}$ by 1; if it was a false negative for c_j, increase $\hat{TP}_j^{La}$ by 1 and decrease $\hat{FN}_j^{La}$ by 1;

[10]Equations (15) might have also been formulated in a continuous way, that is, as partial derivatives of F_1 in the two variables TP_j and TN_j (in other words, Equations (15) would thus represent the gradient of F_1). We have preferred to stick to a discrete formulation, since (a) Equations (4) and (5) are instead *not* naturally formulated as derivatives (exactly because they represent average – rather than pointwise – gains), and since (b) having Equations (4), (5) and (15) all formulated in a common notation allows an easier comparison among them.

(c) resmooth $\hat{TP}_j^{La}$, $\hat{FP}_j^{La}$, $\hat{FN}_j^{La}$ if needed;
(d) recompute $G(d_i, fp_j)$ and/or $G(d_i, fn_j)$ and go back to Step 2.

This might also be dubbed an *incremental* ranking strategy, in the sense pioneered in Aalbersberg [1992] for relevance feedback in ad hoc search, in the sense that the values of $G(d_i, fp_j)$ and $G(d_i, fn_j)$ are incrementally updated so that the $U(d_i, \Omega)$ function reflects the fact that part of Te has indeed been corrected. In keeping with Brandt et al. [2011], we prefer to call it a *dynamic* strategy, and to call the one of Section 4 a *static* one.

Note that in Step 2 we simply compute the maximum element (according to $U(d_i, \Omega)$) of Te instead of sorting the entire set, since we can perform this step in $O(|Te|)$ instead of $O(|Te| \log |Te|)$[11]. Furthermore, note that in this algorithm the recomputation of $U_j(d_i, \Omega)$ does *not* entail the recomputation of the probabilities $P(fp_j)$ and/or $P(fn_j)$ of Equation (3), since these probabilities are computed (i.e., calibrated) once for all, immediately after the training phase.

Note also that computing validation gains via Equations (4) and (5) is the only possibility within the static method (since the values of $G(d_i, fp_j)$ and $G(d_i, fn_j)$ produced must be used unchanged throughout the process), but is clearly inadequate in a dynamic context, in which validation gains are always supposed to be up-to-date reflections of the current situation.

The dynamic nature of this method makes it clear why, as specified at the end of Section 4.3, we smooth the cell count estimates only "on demand" (see also Step 4(c) of the previous algorithm), that is, only if any of $\hat{TP}_j^{ML}$, $\hat{FP}_j^{ML}$, $\hat{FN}_j^{ML}$ is < 1. To see this, suppose that we smooth $\hat{TP}_j^{ML}$, $\hat{FP}_j^{ML}$, $\hat{FN}_j^{ML}$ at each iteration, even when not strictly needed. Adding a count of one to each of them at each iteration means that, after k iterations, k counts have been added to each of them; this means that, after many iterations, the counts added to the cells have completely disrupted the relative proportions among the cells that result from the maximum-likelihood estimation. This would likely make the dynamic method underperform the static method, which does not suffer from this problem since the maximum-likelihood estimates are smoothed only once. As a result, we smooth a contingency table only when strictly needed, that is, when one of $\hat{TP}_j^{ML}$, $\hat{FP}_j^{ML}$, $\hat{FN}_j^{ML}$ is < 1.

By solving the inequality $G(d_i, fn_j) > G(d_i, fp_j)$, we may find out under which conditions correcting a false negative yields a higher gain than correcting a false positive. It turns out that, when validation gains are defined according to Equation (15), $G(d_i, fn_j) > G(d_i, fp_j)$ whenever $FN + FP > 1$, that is, practically always. Of course, this need not be the case for evaluation functions different from F_1, and in particular for instances of F_β with $\beta \neq 1$.

From the standpoint of total computational cost, our dynamic technique is $O(|Te| \cdot (|\mathcal{C}| + |Te|))$, since (i) computing the $U(d_i, \Omega)$ score for $|Te|$ documents and computing their maximum according to the computed $U(d_i, \Omega)$ score can be done in $O(|Te| \cdot |\mathcal{C}|)$ steps, and (ii) this step must be repeated $O(|Te|)$ times. This policy is thus, as expected, computationally more expensive than the previous one.

[11]When computing this maximum element returns repeatedly a document whose labels are all correct, the lack of a sorting step entails the need of computing the maximum element several times in a row with the values of $G(d_i, fp_j)$ and $G(d_i, fn_j)$ unchanged. In these cases, the presence of a sorting step would thus have been advantageous. However, the likelihood that this situation occurs tends to be small, especially when $|\mathcal{C}|$ is large, thus making the computation of the maximum element preferable to sorting.

7.1. Experiments

The results of the experiments with the dynamic version of our utility-theoretic method and of our two oracle-based methods are reported in Figures 2 to 4 and in Table II, where they are indicated as U-Theoretic(d), Oracle1(d), and Oracle2(d). Of course, there exists no dynamic version of the baseline method, since this latter does not involve validation gains.

The first observation that can be drawn from these results is the fact that U-Theoretic(d) is not superior to U-Theoretic(s), as could instead have been expected. In fact, in Figures 2 to 4 the curves corresponding to the former are barely distinguishable from those corresponding to the latter, and the numeric results reported in Table II show no substantial difference either; as reported in Table III, in 7 out of 10 cases (2 learners $\times$ 5 datasets) the difference is not statistically significant. Note that there are extremely small differences also between Oracle1(s) and Oracle1(d); again, in 7 out of 10 cases no statistically significant difference can be detected. This shows that the lack of any substantial difference between static and dynamic is not due to a possible suboptimality of the method for estimating contingency table cells (including the method adopted for smoothing the estimates). Analogously, note also the extremely small differences between Oracle2(s) and Oracle2(d) (again, no statistically significant difference in 7 out of 10 cases), which indicates that the culprit is not the method for estimating the probabilities of misclassification.

This substantial equivalence between the static and the dynamic methods is somehow surprising, since on a purely intuitive basis the dynamic method seems definitely superior to the static one. We think that the reason for this apparently counterintuitive results is that, when validation gains are recomputed in Step 4(d) of the algorithm, the magnitude of the update (i.e., the difference between validation gains before and after the update) is too small to make an impact. This is especially true for large test sets, where incrementing or decrementing by 1 the value of a contingency cell makes too tiny a difference, since that value is very large.

Actually, the part of Figure 2 relative to MP-BOOST displays an apparently strange phenomenon, that is, the fact that for some values of ξ the Oracle2(s) method outperforms Oracle2(d). A similar phenomenon can be noticed in some of the cells of Table II, where the static version of either Oracle1 or Oracle2 outperforms, even if by a small margin, the dynamic version. This seems especially strange for Oracle2(d), which is the theoretically optimal method (since it is a method that operates with perfect foreknowledge), and as such should be impossible to beat. The reason for this apparently counterintuitive behaviour lies not in the ranking methods, but in a counterintuitive property of F_1, that is, the fact that, when $TP = FN = 0$ (i.e., there are no positives in the gold standard – and 25 out of 115 classes in the dataset used in Figure 2 have this property), its value is 0 when $FP > 0$ but 1 when $FP = 0$ (so, $TP = FP = FN = 0$ is a "point of discontinuity" for F_1). This essentially means that, when $TP = FN = 0$ and $FP > 0$, $G(d_i, fn_j)$ is $1/|FP|$ for the static method and 0 for the dynamic method; that is, in this case the dynamic method does not provide any incentive for correcting a false positive, while the static method does. As a result, the static method can speed up the correction of false positives more than the dynamic method does. As mentioned earlier, this phenomenon exposes a suboptimality not of the dynamic method, but of the F_1 function.

In Table IV, we report the actual computation times incurred by both U-Theoretic(s) and U-Theoretic(d) on our five datasets[12]. These figures confirm that the dynamic method is (as already discussed earlier) substantially more expensive to run than

[12]The times reported are relative to an experiment in which the entire test set is validated; this is because, in a simulated experiment, the entire test set must be validated in order to compute the $ER_\rho^M(n)$ values reported

Table IV. Comparison Between the Actual Computation Times (in seconds) of the U-Theoretic(s) and U-Theoretic(d) Methods on our Five Datasets

Dataset	Method	MP-BOOST	SVMs
REUTERS-21578	U-Theoretic(s)	0.426	0.452
	U-Theoretic(d)	3.128	3.021
REUTERS-21578/10	U-Theoretic(s)	0.166	0.153
	U-Theoretic(d)	0.195	0.198
REUTERS-21578/100	U-Theoretic(s)	0.033	0.033
	U-Theoretic(d)	0.046	0.044
OHSUMED	U-Theoretic(s)	10.282	11.251
	U-Theoretic(d)	500.047	577.864
OHSUMED-S	U-Theoretic(s)	0.418	0.424
	U-Theoretic(d)	4.731	4.195

the static method; in particular, the magnitude of this difference, together with the marginal (if any) accuracy improvements brought about by the dynamic method over the static one, shows that the static method is much more cost effective than the dynamic one. In other words, the bad news is that the dynamic method brings about no improvement; the good news is that the computationally cheaper static method is hard to beat.

8. A "MICRO-ORIENTED" RANKING FUNCTION FOR SATC

In Section 3, we have assumed that the evaluation of classification algorithms across the $|\mathcal{C}|$ classes of interest is performed by *macro-averaging* the F_1 results obtained for the individual classes $c_j \in \mathcal{C}$. Consistently with this view, in Section 5 we have introduced macro-averaged versions of E_1, ER_ρ, NER_ρ, and $ENER_\rho$. Macro-averaging across the classes in $|\mathcal{C}|$ essentially means paying equal attention to all of them, irrespective of their frequency or other such characteristics.

However, there is an alternative, equally important way to evaluate effectiveness when a set of $|\mathcal{C}|$ classes is involved, namely, *micro-averaged* effectiveness. While macro-averaged measures are computed by first computing the measure of interest individually on each class-specific contingency table and then averaging the results, micro-averaged measures are computed by merging the $|\mathcal{C}|$ contingency tables into a single one (via summing the values of the corresponding cells) and then computing the measure of interest on the resulting table. For instance, micro-averaged F_1 (noted F_1^μ) is obtained by (i) computing the category-specific values TP_j, FP_j and FN_j for all $c_j \in \mathcal{C}$, (ii) obtaining TP as the sum of the TP_j's (same for FP and FN), and then (iii) applying Equation (1). Measures such as E_1^μ, ER_ρ^μ, NER_ρ^μ, and $ENER_\rho^\mu$ are defined in the obvious way. The net effect of using a single, global contingency table is that micro-averaged measures pay more attention to more frequent classes, that is, the more the members of a class c_j in the test set, the more the measure is influenced by c_j.

Neither macro- nor micro-averaging are the "right" way to average in evaluating multilabel multiclass classification; it is instead the case that in some applications we may want to pay equal attention to all the classes (in which case macro-averaging would be our evaluation method of choice), while in some other applications we may want to pay more attention to the most frequent classes (in which case we should opt for micro-averaging).

in Figures 2 to 4. In a realistic setting in which only a portion of the ranked list is validated, the difference between U-Theoretic(s) and U-Theoretic(d) is smaller, since the cost of recomputing validation gains is roughly proportional to the validation depth, and since this cost affects U-Theoretic(d) but not U-Theoretic(s).

While we have not explicitly discussed this, the method of Section 4 was devised with macro-averaged effectiveness in mind. To see this, note that the $U(d_i, \Omega)$ function of Equation (9) is based on an unweighted sum of the class-specific $U_j(d_i, \Omega)$ scores, that is, it pays equal importance to all classes in $\mathcal{C}$. This means that Equation (9) is optimized for metrics that also pay equal attention to all classes, as all macro-averaged measures do. We now describe a way to modify the method of Section 4 in such a way that it is instead optimal when our effectiveness measure of choice (e.g., $ENER_\rho$) is micro-averaged. To do this, we do away with Equation (9) and (similarly to what happens for F_1^μ and E_1^μ) compute instead $U(d_i, \Omega)$ directly on a single, global contingency table obtained by the cell-wise sum of the class-specific contingency tables. That is, we redefine $U(d_i, \Omega)$ as

$$U(d_i, \Omega) = \sum_{c_j \in \mathcal{C}} \sum_{\omega_k \in \{tp_j, fp_j, fn_j, tn_j\}} P(\omega_k) G(d_i, \omega_k), \tag{16}$$

where

$$\begin{aligned} G(d_i, fp_j) &= \frac{1}{FP}\left(F_1^{FP}(\hat{\Phi}(Te)) - F_1(\hat{\Phi}(Te))\right) \\ &= \frac{1}{FP}\left(\frac{2TP}{2TP+FN} - \frac{2TP}{2TP+FP+FN}\right) \\ G(d_i, fn_j) &= \frac{1}{FN}\left(F_1^{FN}(\hat{\Phi}(Te)) - F_1(\hat{\Phi}(Te))\right) \\ &= \frac{1}{FN}\left(\frac{2(TP+FN)}{2(TP+FN)+FP} - \frac{2TP}{2TP+FP+FN}\right). \end{aligned} \tag{17}$$

Equations (17) are the same as Equation (4) and (5), but for the fact that the latter are class specific (as indicated by the index j) while the former are global. This is due to the fact that, when using micro-averaging, there is a single contingency table, and the gain obtained by correcting, say, a false positive for c_x is equal to the gain obtained by correcting a false positive for c_y, for any $c_x, c_y \in \mathcal{C}$. Of course, Equations (17) are to be applied when the *static* method of Section 4 needs to be optimized for micro-averaging; when we instead want to do the same optimization for the *dynamic* method of Section 7, we need instead to apply, in the obvious way, "global" versions of Equations (15).

Actually, a second aspect in the method of Section 4 that we need to change in order for it to be optimized for micro-averaging is the probability calibration method discussed in Section 6.2. In fact, Equation (13) is clearly devised with macro-averaging in mind, since it minimizes *the average across the* $c_j \in \mathcal{C}$ of the difference between the number Pos_j^{Tr} and the expected number $\mathrm{E}[Pos_j^{Tr}]$ of positive training examples of class c_j. Again, all classes are given equal attention. For our micro-averaging-oriented method, we thus replace Equation (13) with

$$\begin{aligned} &\arg\min_\sigma |Pos^{Tr} - \mathrm{E}[Pos^{Tr}]| \\ &= \arg\min_\sigma \left| \sum_{c_j \in \mathcal{C}} Pos_j^{Tr} - \sum_{c_j \in \mathcal{C}} \mathrm{E}\left[Pos_j^{Tr}\right] \right| \\ &= \arg\min_\sigma \left| \sum_{c_j \in \mathcal{C}} Pos_j^{Tr} - \sum_{c_j \in \mathcal{C}} \sum_{d_i \in Tr} P(c_j|d_i) \right| \\ &= \arg\min_\sigma \left| \sum_{c_j \in \mathcal{C}} Pos_j^{Tr} - \sum_{c_j \in \mathcal{C}} \sum_{d_i \in Tr} \frac{e^{\sigma \hat{\Phi}_j(d_i)}}{e^{\sigma \hat{\Phi}_j(d_i)} + 1} \right|, \end{aligned} \tag{18}$$

where the difference between the number and the expected number of training examples *in the global contingency table* is minimized. It is easy to verify that the two methods may return different values of σ, as the following example shows.

Example 8.1. Suppose that $\mathcal{C} = \{c_1, c_2\}$, that $Pos_1^{Tr} = 20$ and that $Pos_2^{Tr} = 10$. Suppose that when $\sigma = a$ then $\mathrm{E}[Pos_1^{Tr}] = 18$ and $\mathrm{E}[Pos_2^{Tr}] = 8$, while when $\sigma = b$ then $\mathrm{E}[Pos_1^{Tr}] = 17$ and $\mathrm{E}[Pos_2^{Tr}] = 13$. According to Equation (13), value a is better than b (since $\frac{1}{|\mathcal{C}|}\sum_{c_j \in \mathcal{C}} |Pos_j^{Tr} - \mathrm{E}[Pos_j^{Tr}]|$ is equal to 2 for $\sigma = a$ and to 3 for $\sigma = b$), but according to Equation (18) value b is better than a (since $|Pos^{Tr} - \mathrm{E}[Pos^{Tr}]|$ is equal to 4 for $\sigma = a$ and to 0 for $\sigma = b$).

The same smoothing methods as discussed in Section 4.3 can instead be used; however, note that smoothing is likely to be needed much less frequently (if at all) here since, given that we now have a single global contingency table, it is much less likely that any of its cells have values <1.

8.1. Experiments

The experiments with our "micro-oriented" methods are reported in Table V. Note that, since the method we use as baseline corresponds (as noted in Section 6.5) to using U-Theoretic(s) with all validation gains set to 1, the baseline we use here is different from the baseline we had used in Section 6.6, since the latter was optimized for macro-averaging while the one we use here is optimized for micro-averaging. This guarantees that, in both cases, our baselines are strong ones.

The results show that utility-theoretic methods bring about a much slighter improvement with respect to the baseline, compared to what we have seen for the macro-oriented methods. For instance, for the SVM learner, REUTERS-21578 dataset, and validation depth $\xi = 0.10$, the improvement of our (static) micro-oriented utility-theoretic method with respect to the baseline is just $+2\%$, while the improvement was $+51\%$ for the equivalent macro-oriented method. Across the two ranking methods (static and dynamic), five datasets, two learners, and three values of inspection depth studied, improvements range from -1% (i.e., in a few peculiar cases we even have a small deterioration) to $+14\%$, much smaller than in the macro-oriented case in which the improvements ranged between $+2\%$ and $+402\%$.

The main reason for these much smaller improvements lies in the combined action of two factors. The first factor is that the validation gains of Equations (17) are computed on the global contingency table, whose cells contain very large numbers, $|\mathcal{C}|$ times larger than the values in the local contingency tables of the macro-oriented method. This means that, since the values of the validation gains are very small (given that an increase or a decrease by 1 of very large values brings about little difference), the difference between $G(d_i, fp_j)$ and $G(d_i, fn_j)$ is even smaller. This makes the difference between the utility-theoretic methods and the baseline smaller. The second factor is that the utility function of Equation (16), by collapsing all the class-specific utility values for a document into a single value, tends to dwarf the differences between the documents.

It should also be noted that, in the micro-oriented method, improvements are small also *because the margins of improvement are small*. To witness, the improvements brought about by Oracle2(d) (our theoretical upper bound) with respect to the baseline are smaller than for the macro-oriented method. For instance, for the MP-BOOST learner, REUTERS-21578 dataset, and validation depth $\xi = 0.10$, this improvement is $+168\%$, while it was $+571\%$ for the macro-oriented method. So, improving over the baseline is more difficult for the micro-oriented method than for the macro-oriented one. The reason why the margins of improvement are smaller is that, when accuracy is evaluated

Table V. As Table II, but with $ENER^{\mu}_{\rho}(\xi)$ in Place of $ENER^{M}_{\rho}(\xi)$

		MP-Boost						SVMs					
		$\xi = 0.05$		$\xi = 0.10$		$\xi = 0.20$		$\xi = 0.05$		$\xi = 0.10$		$\xi = 0.20$	
Reuters-21578	Baseline	0.107		0.167		0.222		0.240		0.325		0.389	
	U-Theoretic(s)	0.107	(+0%)	0.168	(+1%)	0.224	(+1%)	0.246	(+3%)	0.332	(+2%)	0.395	(+2%)
	U-Theoretic(d)	0.107	(+0%)	0.167	(+0%)	0.224	(+1%)	0.246	(+3%)	0.331	(+2%)	0.394	(+1%)
	Oracle1(s)	0.107	(+0%)	0.168	(+1%)	0.224	(+1%)	0.246	(+3%)	0.332	(+2%)	0.395	(+2%)
	Oracle1(d)	0.107	(+0%)	0.167	(+0%)	0.224	(+1%)	0.246	(+3%)	0.331	(+2%)	0.395	(+2%)
	Oracle2(s)	0.333	(+211%)	0.448	(+168%)	0.512	(+131%)	0.394	(+64%)	0.506	(+56%)	0.556	(+43%)
	Oracle2(d)	0.333	(+211%)	0.448	(+168%)	0.512	(+131%)	0.394	(+64%)	0.506	(+56%)	0.556	(+43%)
Reuters-21578/10	Baseline	0.110		0.169		0.222		0.232		0.317		0.380	
	U-Theoretic(s)	0.112	(+2%)	0.171	(+1%)	0.224	(+1%)	0.237	(+2%)	0.323	(+2%)	0.386	(+2%)
	U-Theoretic(d)	0.113	(+3%)	0.171	(+1%)	0.224	(+1%)	0.238	(+3%)	0.322	(+2%)	0.383	(+1%)
	Oracle1(s)	0.112	(+2%)	0.171	(+1%)	0.224	(+1%)	0.237	(+2%)	0.324	(+2%)	0.386	(+2%)
	Oracle1(d)	0.113	(+3%)	0.171	(+1%)	0.224	(+1%)	0.238	(+3%)	0.324	(+2%)	0.386	(+2%)
	Oracle2(s)	0.325	(+195%)	0.438	(+159%)	0.502	(+126%)	0.385	(+66%)	0.496	(+56%)	0.547	(+44%)
	Oracle2(d)	0.325	(+195%)	0.438	(+159%)	0.502	(+126%)	0.385	(+66%)	0.496	(+56%)	0.547	(+44%)
Reuters-21578/100	Baseline	0.102		0.158		0.208		0.223		0.301		0.361	
	U-Theoretic(s)	0.107	(+5%)	0.163	(+3%)	0.212	(+2%)	0.224	(+0%)	0.305	(+1%)	0.366	(+1%)
	U-Theoretic(d)	0.106	(+4%)	0.162	(+3%)	0.211	(+1%)	0.226	(+1%)	0.304	(+1%)	0.363	(+1%)
	Oracle1(s)	0.115	(+13%)	0.170	(+8%)	0.216	(+4%)	0.232	(+4%)	0.317	(+5%)	0.377	(+4%)
	Oracle1(d)	0.116	(+14%)	0.170	(+8%)	0.217	(+4%)	0.235	(+5%)	0.322	(+7%)	0.383	(+6%)
	Oracle2(s)	0.318	(+212%)	0.429	(+172%)	0.492	(+137%)	0.367	(+65%)	0.481	(+60%)	0.534	(+48%)
	Oracle2(d)	0.318	(+212%)	0.429	(+172%)	0.492	(+137%)	0.367	(+65%)	0.481	(+60%)	0.534	(+48%)
OHSUMED	Baseline	0.442		0.552		0.583		0.492		0.600		0.620	
	U-Theoretic(s)	0.440	(+0%)	0.549	(-1%)	0.580	(-1%)	0.496	(+1%)	0.602	(+0%)	0.621	(+0%)
	U-Theoretic(d)	0.442	(+0%)	0.552	(+0%)	0.582	(+0%)	0.496	(+1%)	0.602	(+0%)	0.621	(+0%)
	Oracle1(s)	0.439	(-1%)	0.549	(-1%)	0.580	(-1%)	0.497	(+1%)	0.602	(+0%)	0.621	(+0%)
	Oracle1(d)	0.441	(+0%)	0.551	(+0%)	0.582	(+0%)	0.497	(+1%)	0.603	(+1%)	0.621	(+0%)
	Oracle2(s)	0.660	(+49%)	0.733	(+33%)	0.711	(+22%)	0.704	(+43%)	0.761	(+27%)	0.727	(+17%)
	Oracle2(d)	0.660	(+49%)	0.733	(+33%)	0.711	(+22%)	0.704	(+43%)	0.761	(+27%)	0.727	(+17%)
OHSUMED-S	Baseline	0.044		0.068		0.094		0.058		0.096		0.136	
	U-Theoretic(s)	0.044	(+1%)	0.069	(+3%)	0.096	(+2%)	0.063	(+10%)	0.102	(+7%)	0.143	(+5%)
	U-Theoretic(d)	0.044	(+1%)	0.070	(+3%)	0.097	(+3%)	0.066	(+14%)	0.104	(+9%)	0.144	(+6%)
	Oracle1(s)	0.044	(+1%)	0.069	(+3%)	0.096	(+2%)	0.064	(+10%)	0.103	(+8%)	0.143	(+5%)
	Oracle1(d)	0.044	(+1%)	0.070	(+3%)	0.097	(+3%)	0.066	(+15%)	0.105	(+10%)	0.144	(+6%)
	Oracle2(s)	0.149	(+242%)	0.221	(+227%)	0.287	(+205%)	0.175	(+203%)	0.259	(+171%)	0.330	(+143%)
	Oracle2(d)	0.149	(+242%)	0.221	(+227%)	0.287	(+205%)	0.175	(+203%)	0.259	(+171%)	0.330	(+143%)

at the macro level, the infrequent classes play a bigger role than when evaluating at the micro level. Infrequent classes are such that a large reduction in error can be achieved even by validating a few documents of the right type (i.e., false negatives). As a consequence, for the infrequent classes a ranking method that pays attention to validation gains has the potential to obtain sizeable improvements in accuracy right from the beginning; and a method that favours the infrequent classes tends to shine when evaluated at the macro level.

9. CONCLUSIONS

We have presented a range of methods, all based on utility theory, for ranking the documents labelled by an automatic classifier. The documents are ranked in such a way as to maximize the expected reduction in classification error brought about by a

human annotator who validates a top-ranked subset of the ranked list. We have also proposed an evaluation measure for such ranking methods, based on the expectation of the (normalized) reduction in error brought about by the human annotator's validation activity. This "semiautomated document classification" task is different from "soft (document-ranking) classification", since in the latter case it is the documents with the highest probability of being members of the class (and not the ones which bring about the highest expected utility if validated) that are top ranked.

Experiments carried out on standard datasets and variants thereof show that the intuition of using utility theory is correct. In particular, of four methods studied, we have found that two methods optimized for micro-averaged effectiveness bring about only limited improvements, while the two methods optimized for macro-averaged effectiveness deliver drastically improved performance with respect to the baseline. We have also found that the two "static" methods, while seemingly inferior to the "dynamic" ones on a purely intuitive basis, perform as well as the dynamic ones at a fraction of the computational cost.

It should be remarked that the very fact of using a utility function, that is, a function in which different events are characterized by different gains, makes sense here since we have adopted an evaluation function, such as F_1, in which correcting a false positive or a false negative brings about different benefits to the final effectiveness score. If we instead adopted standard *accuracy* (i.e., the percentage of binary classification decisions that are correct) as the evaluation measure, utility would default to the probability of misclassification, and our method would coincide with the baseline, since correcting a false positive or a false negative would bring about the same benefit. The methods we have presented are justified by the fact that, in TC and in other classification contexts in which imbalance is the rule, F_1 is the standard evaluation function, while standard accuracy is a deprecated measure because of its lack of robustness to class imbalance (see e.g., Sebastiani [2002, Section 7.1.2] for a discussion of this point).

The methods we have proposed are valid also when a different instantiation of the F_β function (i.e., with $\beta \neq 1$) is used as the evaluation function. This may be the case, for example, when classification is to be applied to a recall-oriented task (such as e-discovery [Oard et al. 2010; Oard and Webber 2013]), in which case values $\beta > 1$ are appropriate. In these cases, our utility-theoretic method can be used once the appropriate instance of F_β is plugged, in place of F_1, into the equations defining the validation gains (and into the equations that lead to the definition of $ENER_\rho(\xi)$). The same trivially holds for any other evaluation function, even different from F_β and even multivariate and nonlinear, provided it can be computed from a contingency table. It is easy to foresee that, the higher the difference between the roles that false positives and false negatives play into the chosen function, the bigger the improvements brought about by the utility-theoretic methods with respect to the baseline are going to be. (For instance, it is easy to foresee that these improvements would be higher for F_2 than for F_1.)

We also remark that this technique is not limited to TC, but can be useful in any classification context in which class imbalance [He and Garcia 2009], or cost-sensitivity in general [Elkan 2001], suggest using a measure (such as F_β) that caters for these characteristics.

Note that, by using our methods, it is also easy to provide the human annotator with an estimate of how accurate the labels of the test set are as a result of her validation activity. In fact, if the contingency cell maximum-likelihood estimates $\hat{TP}_j^{ML}$, $\hat{FP}_j^{ML}$, and $\hat{FN}_j^{ML}$ (see Section 4.3) are updated (adding and subtracting 1 where appropriate) after each correction by the human annotator, at any point in the validation activity these are up-to-date estimates of how well the test set is now classified, and from these estimates F_1 (or other) can be computed as usual.

In the future, we would like to try applying an SATC method after a transductive learner (e.g., Transductive SVMs [Joachims 1999]) has been used to generate the base classifier in place of the standard inductive learners we have used in this work. A transductive method, rather than attempting to generate a model that minimizes the expected risk on *any* test set, attempts to minimize misclassifications on a *specific* test set. When the focus of one's application is squeezing the highest possible accuracy from a specific test set, as is the case when using SATC, it would thus make sense to use a transductive instead of an inductive learning method.

ACKNOWLEDGMENTS

We would like to thank David Lewis and Diego Marcheggiani for many interesting discussions on the topics of this article.

REFERENCES

IJsbrand J. Aalbersberg. 1992. Incremental relevance feedback. In *Proceedings of the 15th ACM International Conference on Research and Development in Information Retrieval (SIGIR 1992)*. Copenhagen, DK, 11–22.

Paul Anand. 1993. *Foundations of Rational Choice Under Risk*. Oxford University Press, Oxford, UK.

Giacomo Berardi, Andrea Esuli, and Fabrizio Sebastiani. 2012. A Utility-theoretic ranking method for semi-automated text classification. In *Proceedings of the 35th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2012)*. Portland, US, 961–970.

Giacomo Berardi, Andrea Esuli, and Fabrizio Sebastiani. 2014. Optimising human inspection work in automated verbatim coding. *International Journal of Market Research* 56, 4 (2014), 489–512.

Christina Brandt, Thorsten Joachims, Yisong Yue, and Jacob Bank. 2011. Dynamic ranked retrieval. In *Proceedings of the 4th International Conference on Web Search and Web Data Mining (WSDM 2011)*. Hong Kong, CN, 247–256.

Carla E. Brodley and Mark A. Friedl. 1999. Identifying mislabeled training data. *Journal of Artificial Intelligence Research* 11 (1999), 131–167.

Prabir Burman. 1987. Smoothing sparse contingency tables. *The Indian Journal of Statistics* 49, 1 (1987), 24–36.

Olivier Chapelle, Bernard Schölkopf, and Alexander Zien (Eds.). 2006. *Semi-Supervised Learning*. The MIT Press, Cambridge, US.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics (ACL 1996)*. Santa Cruz, US, 310–318.

Charles Elkan. 2001. The foundations of cost-sensitive learning. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI 2001)*. Seattle, US, 973–978.

Andrea Esuli, Tiziano Fagni, and Fabrizio Sebastiani. 2006. MP-Boost: A multiple-pivot boosting algorithm and its application to text categorization. In *Proceedings of the 13th International Symposium on String Processing and Information Retrieval (SPIRE 2006)*. Glasgow, UK, 1–12.

Andrea Esuli and Fabrizio Sebastiani. 2009. Active learning strategies for multi-label text classification. In *Proceedings of the 31st European Conference on Information Retrieval (ECIR 2009)*. Toulouse, FR, 102–113.

Andrea Esuli and Fabrizio Sebastiani. 2013. Training data cleaning for text classification. *ACM Transactions on Information Systems* 31, 4 (2013).

Fumiyo Fukumoto and Yoshimi Suzuki. 2004. Correcting category errors in text classification. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*. Geneva, CH, 868–874.

William A. Gale and Kenneth W. Church. 1994. What's wrong with adding one? In *Corpus-Based Research into Language: In Honour of Jan Aarts*, N. Oostdijk and P. de Haan (Eds.). Rodopi, Amsterdam, NL, 189–200.

Shantanu Godbole, Abhay Harpale, Sunita Sarawagi, and Soumen Chakrabarti. 2004. Document classification through interactive supervision of document and term labels. In *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2004)*. Pisa, IT, 185–196.

Haibo He and Edwardo A. Garcia. 2009. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering* 21, 9 (2009), 1263–1284.

William Hersh, Christopher Buckley, T. J. Leone, and David Hickman. 1994. OHSUMED: An interactive retrieval evaluation and new large text collection for research. In *Proceedings of the 17th ACM International Conference on Research and Development in Information Retrieval (SIGIR 1994)*. Dublin, IE, 192–201.

Steven C. Hoi, Rong Jin, and Michael R. Lyu. 2006. Large-scale text categorization by batch mode active learning. In *Proceedings of the 15th International Conference on World Wide Web (WWW 2006)*. Edinburgh, UK, 633–642.

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

Thorsten Joachims. 1999. Transductive inference for text classification using support vector machines. In *Proceedings of the 16th International Conference on Machine Learning (ICML 1999)*. Bled, SL, 200–209.

Ashish Kapoor, Eric Horvitz, and Sumit Basu. 2007. Selective supervision: Guiding supervised learning with decision-theoretic active learning. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence (IJCAI 2007)*. San Francisco, US, 877–882.

Leah S. Larkey and W. Bruce Croft. 1996. Combining classifiers in text categorization. In *Proceedings of the 19th ACM International Conference on Research and Development in Information Retrieval (SIGIR 1996)*. Zürich, CH, 289–297.

David D. Lewis and Jason Catlett. 1994. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of 11th International Conference on Machine Learning (ICML 1994)*. New Brunswick, US, 148–156.

David D. Lewis, Robert E. Schapire, James P. Callan, and Ron Papka. 1996. Training algorithms for linear text classifiers. In *Proceedings of the 19th ACM International Conference on Research and Development in Information Retrieval (SIGIR 1996)*. Zürich, CH, 298–306.

Miguel Martinez-Alvarez, Alejandro Bellogin, and Thomas Roelleke. 2013. Document difficulty framework for semi-automatic text classification. In *Proceedings of the 15th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2013)*. Prague, CZ, 110–121.

Miguel Martinez-Alvarez, Sirvan Yahyaei, and Thomas Roelleke. 2012. Semi-automatic document classification: exploiting document difficulty. In *Proceedings of the 34th European Conference on Information Retrieval (ECIR 2012)*. Barcelona, ES, 468–471.

Andrew K. McCallum and Kamal Nigam. 1998. Employing EM in pool-based active learning for text classification. In *Proceedings of the 15th International Conference on Machine Learning (ICML 1998)*. Madison, US, 350–358.

Alistair Moffat and Justin Zobel. 2008. Rank-biased precision for measurement of retrieval effectiveness. *ACM Transactions on Information Systems* 27, 1 (2008).

Alexandru Niculescu-Mizil and Rich Caruana. 2005. Obtaining calibrated probabilities from boosting. In *Proceedings of the 21st Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI 2005)*. Arlington, US, 413–420.

Douglas W. Oard, Jason R. Baron, Bruce Hedin, David D. Lewis, and Stephen Tomlinson. 2010. Evaluation of information retrieval for E-discovery. *Artificial Intelligence and Law* 18, 4 (2010), 347–386.

Douglas W. Oard and William Webber. 2013. Information retrieval for e-discovery. *Foundations and Trends in Information Retrieval* 7, 2/3 (2013), 99–237.

John C. Platt. 2000. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In *Advances in Large Margin Classifiers*, Alexander Smola, Peter Bartlett, Bernard Schölkopf, and Dale Schuurmans (Eds.). The MIT Press, Cambridge, MA, 61–74.

Hema Raghavan, Omid Madani, and Rosie Jones. 2006. Active learning with feedback on features and instances. *Journal of Machine Learning Research* 7 (2006), 1655–1686.

Stephen E. Robertson. 2008. A new interpretation of average precision. In *Proceedings of the 31st ACM International Conference on Research and Development in Information Retrieval (SIGIR 2008)*. Singapore, SN, 689–690.

Robert E. Schapire and Yoram Singer. 2000. BoosTexter: A boosting-based system for text categorization. *Machine Learning* 39, 2/3 (2000), 135–168.

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *Comput. Surveys* 34, 1 (2002), 1–47.

Burr Settles. 2012. *Active learning*. Morgan & Claypool Publishers, San Rafael, US.

Jeffrey S. Simonoff. 1983. A penalty function approach to smoothing large sparse contingency tables. *The Annals of Statistics* 11, 1 (1983), 208–218.

Simon Tong and Daphne Koller. 2001. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research* 2 (2001), 45–66.

Sudheendra Vijayanarasimhan and Kristen Grauman. 2009. What's it going to cost you?: Predicting effort vs. informativeness for multi-label image annotations. In *Proceedings of the 15th IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2009)*. Miami, US, 2262–2269.

John von Neumann and Oskar Morgenstern. 1944. *Theory of games and economic behavior*. Princeton University Press, Princeton, US.

Yiming Yang and Xin Liu. 1999. A re-examination of text categorization methods. In *Proceedings of the 22nd ACM International Conference on Research and Development in Information Retrieval (SIGIR 1999)*. Berkeley, US, 42–49.

ChengXiang Zhai and John Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems* 22, 2 (2004), 179–214.

Xiaojin Zhu and Andrew B. Goldberg. 2009. *Introduction to Semi-supervised Learning*. Morgan and Claypool, San Rafael, US.

Received July 2014; revised February 2015; accepted March 2015