A Utility-Theoretic Ranking Method for Semi-Automated Text Classification*

Giacomo Berardi, Andrea Esuli, and Fabrizio Sebastiani
Istituto di Scienza e Tecnologie dell'Informazione
Consiglio Nazionale delle Ricerche
56124 Pisa, Italy
[firstname.lastname]@isti.cnr.it

ABSTRACT

In *Semi-Automated Text Classification* (SATC) an automatic classifier $\hat{\Phi}$ labels a set of unlabelled documents $\mathcal{D}$, following which a human annotator inspects (and corrects when appropriate) the labels attributed by $\hat{\Phi}$ to a subset $\mathcal{D}'$ of $\mathcal{D}$, with the aim of improving the overall quality of the labelling.

An automated system can support this process by ranking the automatically labelled documents in a way that maximizes the expected increase in effectiveness that derives from inspecting $\mathcal{D}'$. An obvious strategy is to rank $\mathcal{D}$ so that the documents that $\hat{\Phi}$ has classified with the lowest confidence are top-ranked. In this work we show that this strategy is suboptimal. We develop a new utility-theoretic ranking method based on the notion of *inspection gain*, defined as the improvement in classification effectiveness that would derive by inspecting and correcting 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 inspecting a list generated by a given ranking method. We report the results of experiments showing that, with respect to the baseline method above, and according to the proposed measure, our ranking method can achieve substantially higher expected reductions in classification error.

Categories and Subject Descriptors

I.5.2 [**Pattern Recognition**]: Design Methodology—*Classifier design and evaluation*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering; Search process*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Text analysis*

Keywords

Text classification, supervised learning, semi-automated text classification, cost-sensitive learning, ranking

*In order to correctly interpret the plots in Figures 1 and 2, this document should be printed in colour.

SIGIR'12, August 12–16, 2012, Portland, Oregon, USA.

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 is not sufficient. In this case, the most plausible strategy to follow is to classify $\mathcal{D}$ by means of an automatic classifier $\hat{\Phi}$ (which we assume here to be generated by training a supervised learner on a training set Tr), and then to have a human editor inspect the results of the automatic classification, correcting misclassifications where appropriate[1]. The human annotator will obviously inspect only a subset $\mathcal{D}' \subset \mathcal{D}$ (since it would not otherwise make sense to have an initial automated classification phase), e.g., until she is confident that the overall level of accuracy of $\mathcal{D}$ is sufficient. We call this scenario *semi-automated text classification* (SATC).

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

One obvious strategy 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[2]. The rationale is that an increase in effectiveness can derive only by inspecting *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, when we deal with imbalanced TC problems (as most TC problems indeed

[1]In the rest of this paper we will simply write "inspect" to actually mean "inspect and correct where appropriate".
[2]We call this strategy "obvious" because of the evident similarities between SATC and active learning (see Section 6), where this strategy is an often-used baseline. However, to the best of our knowledge, the application of this ranking method (or of any other ranking method, for that matter) to SATC has never been discussed in the literature.

are) and, as a consequence, choose an evaluation measure – such as F_1 – that caters for this imbalance, the improvements in effectiveness that derive from correcting a false positive or a false negative may not be the same.

The contributions of this paper are the following. First, we develop a new utility-theoretic ranking method for SATC based on the notion of *inspection gain*, i.e., 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, and use it to evaluate our experiments on a standard dataset. The results show that, with respect to the confidence-based baseline method above, our ranking method is substantially more effective.

The rest of the paper is organized as follows. Section 2 sets out preliminary definitions and notation. Section 3 describes our utility-theoretic strategy for ranking the automatically labelled documents, while Section 4 describes the effectiveness measure we propose for this task. Section 5 reports the results of our experiments in which we compare the different ranking strategies by simulating the work of a human annotator that inspects variable portions of the classified test set. Section 6 reviews related work, while Section 7 concludes by charting avenues for future research.

2. PRELIMINARIES

This paper focuses on semi-automated (multi-class) *multi-label* TC. Given a set of textual documents $\mathcal{D}$ and a predefined set of classes $\mathcal{C} = \{c_1, \ldots, c_m\}$, multi-label TC is usually defined as the task of estimating an unknown *target function* $\Phi : \mathcal{D} \times \mathcal{C} \rightarrow \{-1, +1\}$, that describes how documents ought to be classified, by means of a function $\hat{\Phi} : \mathcal{D} \times \mathcal{C} \rightarrow \{-1, +1\}$ called the *classifier*[3]. Here, $+1$ and -1 represent membership and non-membership of the document in the class. Each document may thus belong to zero, one, or several classes at the same time. Multi-label 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 paper we will 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}, i.e., 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 $\hat{\Phi}_j : \mathcal{D} \rightarrow (-\infty, +\infty)$ 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 purposes of this paper we also assume that

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

(the well-known harmonic mean of precision and recall) is the chosen evaluation measure, 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, and TN_j indicate the numbers of true positives, false positives, false negatives, and true negatives in Te. 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.

We also use $TP(ij)$ as a shorthand to indicate that $\hat{\Phi}_j(d_i)$ is a true positive, and use $FP(ij)$, $FN(ij)$, and $TN(ij)$ with analogous meanings. In this paper 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.

[3]Consistently with most mathematical literature we use the caret symbol ($\hat{}$) to indicate estimation.

3. A RANKING METHOD FOR SATC BASED ON UTILITY THEORY

3.1 Ranking by utility

For the moment being, let us concentrate on the binary case, i.e., 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 a function $U_j(d_i)$ that estimates the utility, for the aims of increasing $F_1(\hat{\Phi}_j(Te))$, of manually inspecting the label D_{ij} attributed to d_i by $\hat{\Phi}_j$.

Given a set Ω of mutually disjoint events, a *utility function* is defined as a sum $\sum_{\omega \in \Omega} P(\omega)G(\omega)$, where $P(\omega)$ represents the probability of occurrence of event ω and $G(\omega)$ represents the *gain* obtained if event ω indeed occurs.

Upon submitting document d_i to classifier $\hat{\Phi}_j$, a positive or a negative decision can be returned. If a positive decision is returned (i.e., $D_{ij} = +1$) then the mutually disjoint events $TP(ij)$ and $FP(ij)$ can occur, while if this decision is negative (i.e., $D_{ij} = -1$) then the mutually disjoint events $FN(ij)$ and $TN(ij)$ can occur. We thus naturally define the two utility functions

$$\begin{aligned} U_j^+(d_i) &= P(TP(ij)|D_{ij} = +1) \cdot G(TP(ij)) + \\ &\quad + P(FP(ij)|D_{ij} = +1) \cdot G(FP(ij)) \\ U_j^-(d_i) &= P(FN(ij)|D_{ij} = -1) \cdot G(FN(ij)) + \\ &\quad + P(TN(ij)|D_{ij} = -1) \cdot G(TN(ij)) \end{aligned} \qquad (2)$$

with $U_j^+(d_i)$ addressing the case of a positive decision and $U_j^-(d_i)$ the case of a negative decision. We also define

$$U_j(d_i) = \begin{cases} U_j^+(d_i) & \text{if } D_{ij} = +1 \\ U_j^-(d_i) & \text{if } D_{ij} = -1 \end{cases} \qquad (3)$$

as a function embracing both positive and negative decisions.

3.2 Inspection gains

We equate $G(FP(ij))$ in Equations 2 with the average increase in $F_1(\hat{\Phi}_j(Te))$ that would derive by manually inspecting the label attributed by $\hat{\Phi}_j$ to a document in FP_j. We call this the *inspection gain* of a member of FP_j. From now on we will write $G(FP_j)$ instead of $G(FP(ij))$ so as to reflect the fact that the inspection value is the same for all members of FP_j. Analogous arguments apply to $G(TP(ij))$, $G(FN(ij))$, and $G(TN(ij))$.

Quite evidently, $G(TP_j) = G(TN_j) = 0$, since when the human annotator inspects 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. This means that Equations 2 simplify to

$$\begin{aligned} U_j^+(d_i) &= P(FP(ij)|D_{ij} = +1) \cdot G(FP_j) \\ U_j^-(d_i) &= P(FN(ij)|D_{ij} = -1) \cdot G(FN_j) \end{aligned} \qquad (4)$$

$G(FP_j)$ (resp., $G(FN_j)$) evaluates instead to the average increase in $F_1(\hat{\Phi}_j(Te))$ obtained by correcting a false positive (resp., a false negative). It is easy to see that, in general, $G(FP_j) \neq G(FN_j)$. 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, i.e.,

$$\begin{aligned} G(FP_j) &= \frac{1}{FP_j}(F_1^{FP}(\hat{\Phi}_j) - F_1(\hat{\Phi}_j(Te))) \\ &= \frac{1}{FP_j}(\frac{2TP_j}{2TP_j + FN_j} \\ &\quad - \frac{2TP_j}{2TP_j + FP_j + FN_j}) \end{aligned} \tag{5}$$

where by $F_1^{FP}(\hat{\Phi}_j)$ we indicate the value of F_1 that would derive by correcting all false positives of $\hat{\Phi}_j(Te)$, i.e., turning 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, i.e.,

$$\begin{aligned} G(FN_j) &= \frac{1}{FN_j}(F_1^{FN}(\hat{\Phi}_j) - F_1(\hat{\Phi}_j(Te))) \\ &= \frac{1}{FN_j}(\frac{2(TP_j + FN_j)}{2(TP_j + FN_j) + FP_j} \\ &\quad - \frac{2TP_j}{2TP_j + FP_j + FN_j}) \end{aligned} \tag{6}$$

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.

3.3 Estimating the probabilities

We derive the probabilities $P(\cdot)$ in Equations 4 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

$$\begin{aligned} P(FP(ij)|D_{ij} = +1) &= 1 - \frac{e^{\sigma C_{ij}}}{e^{\sigma C_{ij}} + 1} \\ P(FN(ij)|D_{ij} = -1) &= 1 - \frac{e^{\sigma C_{ij}}}{e^{\sigma C_{ij}} + 1} \end{aligned} \tag{7}$$

The generalized logistic function has the effect of monotonically converting scores ranging on $(-\infty, +\infty)$ into real values in the [0.0,1.0] range. When $C_{ij} = 0$ (this happens when $\hat{\Phi}_j$ has no confidence at all in its own decision D_{ij}), then $P(TP(ij)|D_{ij} = +1) = P(FP(ij)|D_{ij} = +1) = 0.5$ and $P(FN(ij)|D_{ij} = -1) = P(TN(ij)|D_{ij} = -1) = 0.5$, i.e., the probability of correct classification and the probability of misclassification are identical. Conversely, we have

$$\begin{aligned} \lim_{C_{ij} \to +\infty} P(FP(ij)|D_{ij} = +1) &= 0 \\ \lim_{C_{ij} \to +\infty} P(FN(ij)|D_{ij} = -1) &= 0 \end{aligned}$$

This means that, 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 very low.

The reason why we use a *generalized* version of the logistic function instead of the standard version (which corresponds to the case $\sigma = 1$) is that using this latter within Equations 7 would give rise to a very high number of zero probabilities of misclassification, since the standard logistic function converts every positive number above a certain threshold (≈ 36) to a number that standard implementations round 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 Equations 7 distribute the confidence values across the [0.0,0.5] interval. How we optimize the σ parameter is discussed in Section 5.1.

3.4 Smoothing contingency cell estimates for computing $G(FP_j)$ and $G(FN_j)$

One problem that needs to be tackled in order to compute $G(FP_j)$ and $G(FN_j)$ is that the contingency cell counts TP_j, FP_j, and FN_j are not known, and thus need to be estimated[4]. In order to estimate $\alpha_j \in \{TP_j, FP_j, FN_j\}$ we make the assumption that the training set and the test set are independent and identically distributed. We then perform a k-fold cross-validation 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 cross-validation on Tr, the maximum-likelihood estimate of TP_j is $\hat{TP}_j^{ML} = TP_j^{Tr} \cdot |Te|/|Tr|$; the same holds for $\hat{FP}_j^{ML}$ and $\hat{FN}_j^{ML}$.

However, these maximum-likelihood cell count estimates (noted $\hat{\alpha}_j^{ML}$) need to be smoothed, so as to avoid zero counts. In fact, if $\hat{TP}_j^{ML} = 0$, it would derive from Equation 5 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; the same goes for $\hat{FN}_j^{ML}$.

A second reason why the $\hat{\alpha}_j^{ML}$ need to be smoothed is that, when $|Te|/|Tr| < 1$, they may give rise to negative values for $G(FP_j)$ and $G(FN_j)$, which is obviously counterintuitive. To see this, note that the $\hat{\alpha}_j^{ML}$ 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 $G(FP_j)$ and $G(FN_j)$ may derive).

Smoothing has extensively been studied in language modelling for speech processing [3] and for ad hoc search in IR [24]. However, the present context is slightly different, in that we need to smooth contingency tables, and not (as in the cases above) language models. In particular, while the $\hat{\alpha}_j^{ML}$ are the obvious counterpart of the document model resulting from maximum-likelihood estimation, there is no obvious counterpart to the "collection model", thus making the use of, e.g., 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 pur-

[4]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.

pose of smoothing contingency cell estimates [1, 21]. However, these methods are inapplicable to our case, since they were originally conceived for contingency tables 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 [24] which is obtained by adding a fixed quantity to all the $\hat{\alpha}_j^{ML}$. As a fixed quantity we add 1, since it is the quantity that all our cell counts need to be greater or equal to for Equations 5 and 6 to make sense. We thus leave the study of more sophisticated smoothing methods to future work.

However, it should be noted that we apply smoothing in an "on demand" fashion, i.e., we check if the contingency table needs smoothing at all (i.e, if any of the $\hat{\alpha}_j^{ML}$ is < 1) and we smooth it only if this is the case.

3.5 Ranking by total utility

Our function $U_j(d_i)$ of Section 3.1 is thus obtained by plugging Equations 5 and 6 into Equations 4.

At this point, it would seem sensible to propose ranking, for each $c_j \in \mathcal{C}$, all the automatically labelled documents in Te in decreasing order of their $U_j(d_i)$ 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 documents (this might mean reading the same document $|\mathcal{C}|$ times in order to validate its labels). As suggested in [6] for active learning, it seems instead more plausible to generate a *single* ranking, according to a score $U(d_i)$ that is a function of the $|\mathcal{C}|$ different $U_j(d_i)$ 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 $U(d_i)$ we use *total utility*, corresponding to the simple sum

$$U(d_i) = \sum_{c_j \in \mathcal{C}} U_j(d_i) \tag{8}$$

Our final ranking is thus generated by sorting the test documents in descending order of their $U(d_i)$ 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 $U(\cdot)$ score is $O(|Te| \log |Te|)$, and the cost of computing the $U(\cdot)$ score for $|Te|$ documents and $|\mathcal{C}|$ classes is $O(|Te| \cdot |\mathcal{C}|)$.

4. 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* (noted $ENER_\rho$).

4.1 Error reduction at rank

Let us first introduce the notion of *residual error at rank n* (noted $E_\rho(n)$, which we assume to range on [0,1]), defined as the error that is still present in the document set Te after the human annotator has inspected 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 will hereafter call n the *inspection length*.

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

$$ER_\rho(n) = \frac{E_\rho(0) - E_\rho(n)}{E_\rho(0)} \tag{9}$$

i.e., a value in [0,1] that indicates the error reduction obtained by a human annotator who has inspected 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 Figures 1 and 2, where different curves represent different ranking methods $\rho', \rho'', ...$, and where, for better convenience, the x axis indicates the fraction $n/|Te|$ of the test set that has been inspected rather than the number n of inspected documents. By definition all curves start at the origin of the axes and end at the upper right corner of the graph. Higher curves represent better strategies, since they indicate that a higher error reduction is achieved for the same amount of manual inspecting 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 inspecting the documents does *not* always have the same upper bound. For instance, if the initial accuracy is 0.5 (with accuracy values ranging 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 problematic, since different datasets and/or different classifiers give rise to different initial levels of accuracy. So, using error reduction instead of increase in accuracy "normalizes" our curves, i.e., allows a meaningful comparison of curves obtained on different datasets and after different classifiers have been used.

Since (as stated in Section 2) we use F_1 for measuring effectiveness, as a measure of classification error we use $E_1 \equiv (1 - F_1)$. In order to measure the overall effectiveness of a ranking method across the entire set $\mathcal{C}$ of categories, we compute two versions of $ER_\rho(n)$, one based on *microaveraged* E_1 (denoted by E_1^μ) and one based on *macroaveraged* E_1 (E_1^M). E_1^μ is obtained by (i) computing the class-specific values TP_j, FP_j and FN_j, (ii) obtaining TP as the sum of the TP_j's (same for FP and FN), and then (iii) applying the formula $E_1 = 1 - \frac{2TP}{2TP+FP+FN} = \frac{FP+FN}{2TP+FP+FN}$. E_1^M is instead obtained by computing the class-specific E_1 values and averaging them across the c_j's. The two versions of $ER_\rho(n)$ will be indicated as $ER_\rho^\mu(n)$ and $ER_\rho^M(n)$.

4.2 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[5], since it

[5]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

amounts (for both $ER^{\mu}_{\rho}(n)$ and $ER^{M}_{\rho}(n)$) 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 the two versions noted as $NER^{\mu}_{\rho}(n)$ and $NER^{M}_{\rho}(n)$.

4.3 ... 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 [18] we assume that the human annotator stops inspecting the ranked list at exactly rank n with probability $P_s(n)$. We can then define the *expected normalized error reduction* of ranking method ρ on a given document set Te as

$$ENER_{\rho} = \sum_{n=1}^{|Te|} P_s(n) NER_{\rho}(n) \tag{10}$$

with the two versions indicated as $ENER^{\mu}_{\rho}$ and $ENER^{M}_{\rho}$.

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 [15]) that a human annotator, after inspecting a document, goes on to inspect the next document with probability p (also called *persistence* in [15]) or stops inspecting with probability $(1-p)$, so that

$$P_s(n) = \begin{cases} p^{n-1}(1-p) & \text{if } n \in \{1, \ldots, |Te|-1\} \\ p^{n-1} & \text{if } n = |Te| \end{cases} \tag{11}$$

It can be shown that, for a sufficiently large value of $|Te|$, the expected number of documents that the human annotator will inspect 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 inspect as a function of p.

Using this distribution entails the need of determining a realistic value for p. A value $p = 0$ corresponds to a situation in which the human annotator only inspects the top-ranked document, while $p = 1$ indicates a human annotator who inspects 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 $\xi = .2$ (i.e., that the average human annotator inspects 20% of the test set). In this case, if $|Te| = 1,000$, then $p = 1 - \frac{1}{.2 \cdot 1000} = .9950$, while if $|Te| = 10,000$, then $p = 1 - \frac{1}{.2 \cdot 10000} = .9995$. In the experiments of Section 5 we will test all values of p corresponding to values of ξ in $\{.05, .10, .20\}$.

run on our datasets; see Figures 1 and 2 for a graphical representation.

Note that the values of $ENER_{\rho}$ are bounded 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 inspected anyway, 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.

5. EXPERIMENTS

5.1 Experimental protocol

Let Ω be a dataset partitioned into a training set Tr and a test set Te. In each experiment reported in this paper 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^{Tr}_j, FP^{Tr}_j, and FN^{Tr}_j;
      ii. optimizing the σ parameter of Equations 7;
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 $ENER^{\mu}_{\rho}$ and $ENER^{M}_{\rho}$ for different values of ξ.

For Step 1b we have used $k = 10$.

The optimization method we use for Step 1(b)ii consists in picking the value of σ that minimizes the average (across the $c_j \in \mathcal{C}$) absolute value of the difference between Pos^{Tr}_j, the number of positive training examples of class c_j, and $E[Pos^{Tr}_j]$, the *expected* number of such examples as resulting from the probabilities of membership in c_j computed in the k-fold cross-validation. That is, we pool together all the training documents classified in the k-fold cross-validation, and then we pick

$$\arg\min_{\sigma} \frac{1}{|\mathcal{C}|} \sum_{c_j \in \mathcal{C}} |Pos^{Tr}_j - E[Pos^{Tr}_j]| \quad =$$

$$\arg\min_{\sigma} \frac{1}{|\mathcal{C}|} \sum_{c_j \in \mathcal{C}} |Pos^{Tr}_j - \sum_{d_i \in Tr} \frac{e^{\sigma \hat{\Phi}_j(d_i)}}{e^{\sigma \hat{\Phi}_j(d_i)} + 1}|$$

This is a much faster parameter optimization method than the traditional method of picking the value that has performed best in k-fold cross-validation since, 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, i.e., for each combination of a tested value of σ and a ranking method ρ.

As the learner for generating our classifiers $\hat{\Phi}_j$ we use a boosting-based learner called MP-Boost [5]. Boosting-based methods have showed very good performance across many learning tasks and, at the same time, have strong justifications from computational learning theory. MP-Boost

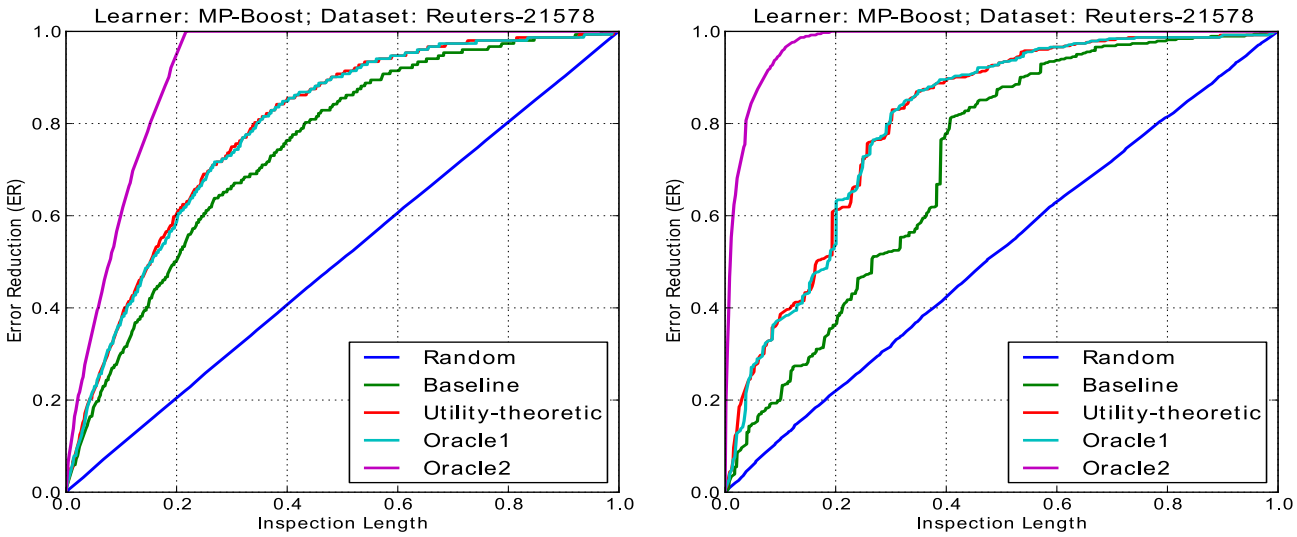

Figure 1: Error reduction as a function of the fraction of the test set that the human annotator has inspected. The dataset is the Reuters-21578 collection. Error reduction is measured as ER_ρ^μ (left) and ER_ρ^M (right). The Random curve indicates the results of our estimation of the expected performance of the random ranker via a Monte Carlo method with 50 random trials. Higher curves are better.

is a variant of AdaBoost.MH [19] optimized for multi-label settings, which has been shown in [5] to obtain considerable effectiveness improvements with respect to AdaBoost.MH. MP-Boost generates a classifier $\hat{\Phi}_j$ where $sgn(\hat{\Phi}_j(d_i))$ represents the binary decision as to whether d_i belongs to c_j and $|\hat{\Phi}_j(d_i)|$ represents the confidence in this decision. In all our experiments we set the S parameter of MP-Boost (representing the number of boosting iterations) to 1000.

As dataset we have used the Reuters-21578 corpus. It consists of a set of 12,902 news stories, partitioned (according to the "ModApté" split we have adopted) into a training set of 9,603 documents and a test set of 3,299 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[6] and is probably the most widely used benchmark in text classification research, which allows other researchers to easily replicate the results of our experiments.

In all the experiments discussed in this paper 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.

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

5.2 Results and discussion

5.2.1 *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. As discussed in Footnote 2, while this strategy has not explicitly been proposed before, it seems a reasonable, common-sense strategy anyway.

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 utility-theoretic method, i.e., at assessing the effectiveness of "idealized" (albeit non-realistic) systems at our task.

The first such method (dubbed Oracle1) consists of "peeking" at the actual values of TP_j, FP_j, and FN_j in Te, using them in the computation of $G(FP_j)$ and $G(FN_j)$, and running 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 smoothing method adopted, 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, and FN_j in the computation of $G(FP_j)$ and $G(FN_j)$ (as in Oracle1), *and* (b) replace the probabilities in Equations 4 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 run 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

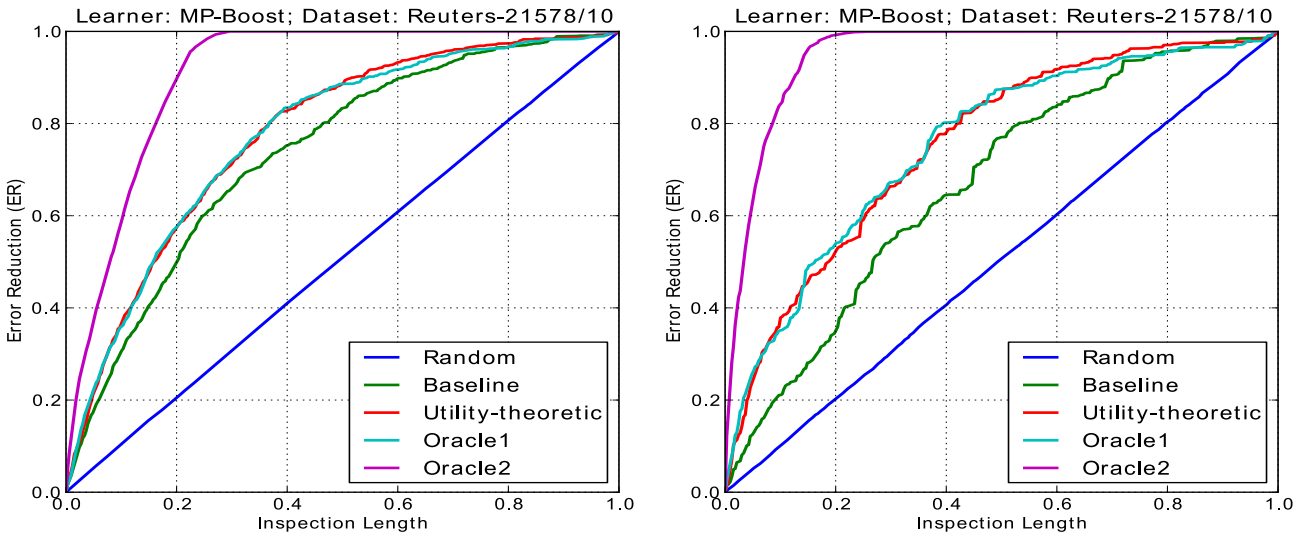

Figure 2: Results obtained by (a) splitting the Reuters-21578 test set into 10 random, equally-sized parts, (b) running the analogous experiments of Figure 1 independently on each part, and (c) averaging the results across the 10 parts.

(obvious) inability to perfectly predict whether a document was classified correctly or not.

5.2.2 *Large test sets*

Figure 1 illustrates the results of our experiments on the REUTERS-21578 dataset in terms of $ER^{\mu}_{\rho}(n)$ and $ER^{M}_{\rho}(n)$; the results of the same experiments in terms of $ENER^{\mu}_{\rho}$ and $ENER^{M}_{\rho}$ as a function of the chosen value of ξ are instead reported in Table 1. The initial error generated by the automatic classifier is $E^{\mu}_{1} = .152$ and $E^{M}_{1} = .383$. The optimal value of σ returned by the k-fold cross-validation phase is .420.

The first insight we can draw from these results is that our utility-theoretic method outperforms the baseline in a very substantial way. For instance, for $\xi = .10$ (corresponding to $p = .996$) it obtains a relative improvement over the baseline of +30% in terms of $ENER^{\mu}_{\rho}$ and of +119% in terms of $ENER^{M}_{\rho}$. Improvements obtained for the two other tested values of ξ are qualitatively similar.

A second insight is that, surprisingly, our method hardly differs in terms of performance from Oracle1. The two curves can be barely distinguished in Figure 1, and in terms of $ENER_{\rho}$ Oracle1 is even outperformed by our utility-theoretic method, albeit by a narrow margin (.221 vs. .217 for $ENER^{\mu}_{\rho}$ and .233 vs. .224 for $ENER^{M}_{\rho}$, both for $\xi = .10$; the other tested values for ξ give similar results). This shows that (at least judging from this experiment) 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 [9]. The fact that our method slightly (and strangely) outperforms Oracle1 is probably due to accidental, "serendipitous" interactions between the probability estimation component (Equation 7) and the contingency cell estimation component of Section 3.4.

Note that in Figure 1 the ER^{μ}_{ρ} curves (left) are smoother than the ER^{M}_{ρ} curves (right). This is due to the fact that E^{μ}_{1} is evaluated on a single, global contingency table, so that correcting an individual document always has a small effect on E^{μ}_{1}. By contrast, even correcting a single document may have a major effect on a class-specific value of E_1 (especially if the class is infrequent), and this may bring about a relatively major effect on E^{M}_{1} too.

5.2.3 *Small (and tiny) test sets*

We have run a second batch of experiments in which we randomly split the REUTERS-21578 test set in 10 equally-sized parts (about 330 documents each), we run each ranking method on each such part individually, and we average the results across the 10 parts. We call this scenario REUTERS-21578/10. The relative results in terms of $ER^{\mu}_{\rho}(n)$ and $ER^{M}_{\rho}(n)$ are shown in Figure 2, while the results of the same experiments in terms of $ENER^{\mu}_{\rho}$ and $ENER^{M}_{\rho}$ are reported in Table 2.

The rationale of these experiments is checking how the methods fare when ranking test sets much smaller that the REUTERS-21578 test set. This is *more* challenging than ranking larger sets, since in this case Laplace smoothing (i) can seriously perturb the relative proportions among the cell counts, which can generate poor estimates of $G(FP_j)$ and $G(FN_j)$, and (ii) is performed for more classes, since we smooth "on demand" only and since the likelihood that the $\hat{\alpha}_j$ are smaller than 1 is higher with small test sets.

Figure 2 confirms that our utility-theoretic method substantially outperforms the baseline also in this context. Note that the E^{M}_{1} curves (left) are smoother than the analogous curves of the full REUTERS-21578. This is due to the fact that the curves in Figure 2 result from averages across 10 different experiments, and the increase brought about at rank n is actually the average of the increases brought about at rank n in the 10 experiments.

That our utility-theoretic method substantially outperforms the baseline also in this experiment can be seen also

Table 1: Results of various ranking methods on Reuters-21578 in terms of EER_ρ; the notation $EER^x_\rho(y)$ indicates EER_ρ values obtained by using x as an averaging method (micro- or macro-averaging) for E_1 and y as a value for ξ. Improvements listed for the various methods are relative to the baseline.

	$ENER^\mu_\rho(0.05)$		$ENER^\mu_\rho(0.10)$		$ENER^\mu_\rho(0.20)$		$ENER^M_\rho(0.05)$		$ENER^M_\rho(0.10)$		$ENER^M_\rho(0.20)$	
Baseline	.109		.169		.224		.070		.106		.155	
Utility-theoretic	.145	(+32%)	.221	(+30%)	.285	(+27%)	.162	(+133%)	.233	(+119%)	.298	(+92%)
Oracle1	.142	(+29%)	.217	(+28%)	.281	(+25%)	.151	(+116%)	.224	(+111%)	.292	(+88%)
Oracle2	.277	(+153%)	.401	(+137%)	.480	(+113%)	.671	(+864%)	.725	(+582%)	.701	(+351%)

Table 2: As in Table 1, but with Reuters-21578/10 instead of Reuters-21578.

	$ENER^\mu_\rho(0.05)$		$ENER^\mu_\rho(0.10)$		$ENER^\mu_\rho(0.20)$		$ENER^M_\rho(0.05)$		$ENER^M_\rho(0.10)$		$ENER^M_\rho(0.20)$	
Baseline	.110		.169		.222		.063		.097		.136	
Utility-theoretic	.141	(+27%)	.212	(+25%)	.272	(+22%)	.150	(+138%)	.210	(+116%)	.257	(+88%)
Oracle1	.144	(+30%)	.215	(+26%)	.273	(+23%)	.163	(+158%)	.220	(+126%)	.264	(+94%)
Oracle2	.288	(+161%)	.403	(+138%)	.477	(+114%)	.480	(+662%)	.585	(+503%)	.612	(+349%)

from Table 2. For $\xi = .10$ (corresponding to $p = .969$) the relative improvement over the baseline is +25% for $ENER^\mu_\rho$ and +116% for $ENER^M_\rho$. Similarly substantial improvements are obtained for the two other values of ξ tested.

Note also that in this case our method underperforms Oracle1 (.212 vs. .215 for $ENER^\mu_\rho$, .210 vs. .220 for $ENER^M_\rho$), and this points to a possible, small suboptimality of the smoothing method adopted. We leave the investigation of more sophisticated smoothing methods to a future paper.

In further experiments that we have run, we have split the Reuters-21578 test set even further, i.e., 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. The $ENER^\mu_\rho$ and $ENER^M_\rho$ results for this Reuters-21578/100 scenario are reported in Figure 3; we do not report the detailed $ER^\mu_\rho(n)$ and $ER^M_\rho(n)$ plots for reasons of space. Our utility-theoretic model still outperforms the baseline, with a relative improvement of +18% on $ENER^\mu_\rho$ and +48% on $ENER^M_\rho$ with $\xi = .10$, corresponding to $p = .696$; qualitatively similar improvements are obtained with the other tested values of ξ.

However, we consider the Reuters-21578/100 scenario less interesting 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 inspecting just 33 documents can arguably check them all without any need for ranking.

Incidentally, note that the Reuters-21578/10 and Reuters-21578/100 experiments model an application scenario in which a set of automatically labelled documents is split (e.g., to achieve faster throughput) among k human annotators, each one entrusted with inspecting 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.

5.2.4 A note on (micro- or macro-) averaging

It is important to note that both the baseline and our utility-theoretic method are *explicitly optimized* for $ENER^M_\rho$, and not for $ENER^\mu_\rho$. To see this, note that the $U(d_i)$ function of Equation 8 is based on an unweighted sum of the class-specific $U_j(d_i)$ scores, i.e., it pays equal importance to all classes, irrespective of frequency considerations. This means that it is optimized for metrics that also pay equal attention to all classes, as all macroaveraged measures such as $ENER^M_\rho$ do.

By contrast, $ENER^\mu_\rho$ pays attention to classes proportionally to their frequency. Therefore, a ranking method that optimizes for $ENER^\mu_\rho$ should instead do away with class-specific utility functions and use a utility function that (similarly to what happens for E^μ_1) is directly computed on a single, global contingency table obtained by the cell-wise sum of the class-specific contingency tables. In such a method, $G(FP)$ and $G(FN)$ would be global to $\mathcal{C}$, i.e., they would be the same for all $c_j \in \mathcal{C}$. We leave the investigation of ranking methods optimized for $ENER^\mu_\rho$ to a future paper.

All this shows that the measure according to which both the baseline and our utility-theoretic method should be evaluated is $ENER^M_\rho$, and not $ENER^\mu_\rho$, since it is $ENER^M_\rho$ that these methods were designed for[7]. However, we have also reported results measured according to E^μ_1 for completeness, and in order to show that, while our methods were not meant for use with E^μ_1 as the yardstick, they still perform well even in this context.

It should thus come as no surprise that the improvements displayed by our method (and the oracles) over the baseline are always higher, or much higher, for $ENER^M_\rho$ than for $ENER^\mu_\rho$, as is apparent from all the figures and tables in this paper.

6. RELATED WORK

Many researchers have tackled the problem of how to use automated TC technologies in application contexts in which the required accuracy levels are unattainable by the generated automatic classifiers.

A standard response to this problem is to adopt *active learning* (AL – see e.g., [11, 22]), i.e., use algorithms that optimize the informativeness of additional training examples provided by a human annotator. Still, providing additional training examples, no matter how carefully chosen, may be insufficient, since in many applicative contexts high

[7]The baseline we have used is, as specified in Section 5.2.1, our utility-theoretic method with $G(FP_j)$ and $G(FN_j)$ set to 1; it is thus also optimized for E^M_1. A baseline method optimized for E^μ_1 would be the method outlined in the last paragraph with $G(FP)$ and $G(FN)$ set to 1.

Table 3: As in Table 1, but with Reuters-21578/100 instead of Reuters-21578.

	$ENER^{\mu}_{\rho}(0.05)$		$ENER^{\mu}_{\rho}(0.10)$		$ENER^{\mu}_{\rho}(0.20)$		$ENER^{M}_{\rho}(0.05)$		$ENER^{M}_{\rho}(0.10)$		$ENER^{M}_{\rho}(0.20)$	
Baseline	.102		.158		.207		.073		.117		.158	
Utility-theoretic	.119	(+16%)	.187	(+18%)	.244	(+17%)	.115	(+56%)	.173	(+48%)	.221	(+39%)
Oracle1	.170	(+66%)	.241	(+52%)	.292	(+41%)	.153	(+108%)	.208	(+77%)	.249	(+57%)
Oracle2	.306	(+198%)	.417	(+163%)	.482	(+132%)	.362	(+395%)	.473	(+304%)	.527	(+233%)

enough accuracy levels cannot be attained, irrespectively of the quantity and quality of the available training data. Similar considerations apply when active learning is carried out at the term level, rather than at the document level [10, 17].

A related response to the same problem is to adopt *training data cleaning* (TDC – see e.g., [7, 8]), i.e., use algorithms that optimize the human annotator's efforts at correcting possible labelling mistakes in the training set. Similarly to the case of AL, in many applicative contexts high enough accuracy levels cannot be attained even at the price of carefully inspecting the entire training set for labelling mistakes.

Both AL and TDC are different from the task we deal with, since we are not concerned with improving the quality of the *training* set. We are instead concerned with improving the quality of the automatically classified *test* set, typically after all attempts at injecting quality in the automatic classifier have proved insufficient; in particular, no retraining / reclassification phase is involved in SATC.

Active learning. As remarked above, SATC certainly bears strong relations with active learning. In both SATC and in the *selective sampling* – also known as *pool-based* – approach to AL [13, 14], 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 above, the goals of the two tasks are different. For instance, in active learning 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 inspected, would 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.

Semi-automated TC. While AL (and, to a much lesser degree, TDC) have been investigated extensively in a TC context, semi-automated TC has been completely neglected by the research community. While a number of papers (e.g., [12, 20, 23]) have evoked the existence of this scenario, we are not aware of any 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 [12] 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 inspecting activity brings about, or methods aimed at optimizing the cost-effectiveness of this activity.

7. CONCLUSIONS AND FUTURE WORK

We have presented a method 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 inspects a subset of the ranked list and corrects the labels when appropriate. We have also proposed an evaluation measure for such ranking methods, based on the notion of expected normalized error reduction. Experiments carried out on a standard dataset show that our method substantially outperforms a state-of-the-art baseline method. To the best of our knowledge, this is the first paper in the literature that addresses semi-automated text classification as a task in its own right, and which presents methods explicitly devised for optimizing it; this is obviously the reason of the very substantive improvements obtained by our method with respect to the baseline.

It should be remarked that the very fact of using a utility function, i.e., 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 indeed brings about different benefits to the final effectiveness score. If we instead adopted *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 method we have presented is justified by the fact that F_1 is the standard evaluation function for text classification, while accuracy is a deprecated measure in a text classification context since it is not robust to class imbalance. See e.g., [20, Section 7.1.2] for a discussion of this point.

The method we have proposed is obviously 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, e.g., when classification is to be applied to a recall-oriented task (such as e-discovery [16]), 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 used in Equations 5 and 6 in place of F_1. The same trivially holds for any other evaluation function, even different from F_β and even multivariate and non-linear, provided it can be computed from a contingency table. We also remark that this technique is obviously not limited to *text* classification, but can be useful in any classification context in which class imbalance [2], or cost-sensitivity in general [4], suggest using a measure (such as F_β) that caters for these characteristics.

Note that, by using our method, 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 inspecting all the documents until rank n. In fact, if the contingency cell estimates $\hat{TP}_j$, $\hat{FP}_j$, and $\hat{FN}_j$ (see Section 3.4) are up-

dated (adding and subtracting 1 where appropriate) after each correction made by the human annotator, at any point in the inspection 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 next future we plan to carry our more experiments, using additional datasets, learners, and (when the test sets are small) smoothing methods. We are also currently testing an improved ranking method that we have recently designed. Essentially, this "dynamic" method is based on the observation that inspecting a document misclassified for c_j brings about changes in at least one of TP_j, FP_j, and FN_j (and in $G(FP_j)$ and/or $G(FN_j)$ and $U_j(\cdot)$ as a consequence). This dynamic version of our ranking strategy consists of updating (after each correction has been performed) $\hat{TP}_j$, $\hat{FP}_j$, and $\hat{FN}_j$ by adding and subtracting 1 where appropriate, re-estimating $G_j(FP)$, $G_j(FN)$ and $U_j(\cdot)$, and bringing to bear these new estimates when selecting the document that should be presented next to the human annotator.

8. REFERENCES

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

[2] N. V. Chawla, N. Japkowicz, and A. Kolcz. Editorial: Special issue on learning from imbalanced data sets. *ACM SIGKDD Explorations*, 6(1):1–6, 2004.

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

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

[5] A. Esuli, T. Fagni, and F. Sebastiani. 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)*, pages 1–12, Glasgow, UK, 2006.

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

[7] A. Esuli and F. Sebastiani. Training data cleaning for text classification. In *Proceedings of the 2nd International Conference on the Theory of Information Retrieval (ICTIR 2009)*, pages 29–41, Cambridge, UK, 2009.

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

[9] W. Gale and K. Church. What's wrong with adding one? In N. Oostdijk and P. de Haan, editors, *Corpus-Based Research into Language: In honour of Jan Aarts*, pages 189–200. Rodopi, Amsterdam, NL, 1994.

[10] S. Godbole, A. Harpale, S. Sarawagi, and S. Chakrabarti. 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)*, pages 185–196, Pisa, IT, 2004.

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

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

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

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

[15] A. Moffat and J. Zobel. Rank-biased precision for measurement of retrieval effectiveness. *ACM Transactions on Information Systems*, 27(1), 2008.

[16] D. W. Oard, J. R. Baron, B. Hedin, D. D. Lewis, and S. Tomlinson. Evaluation of information retrieval for E-discovery. *Artificial Intelligence and Law*, 18(4):347—386, 2010.

[17] H. Raghavan, O. Madani, and R. Jones. Active learning with feedback on features and instances. *Journal of Machine Learning Research*, 7:1655–1686, 2006.

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

[19] R. E. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[20] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

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

[22] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2001.

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

[24] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 22(2):179—214, 2004.