Improving Text Classification Accuracy by Training Label Cleaning

ANDREA ESULI and FABRIZIO SEBASTIANI, Consiglio Nazionale delle Ricerche, Italy

In text classification (TC) and other tasks involving supervised learning, labelled data may be scarce or expensive to obtain. Semisupervised learning and active learning are two strategies whose aim is maximizing the effectiveness of the resulting classifiers for a given amount of training effort. Both strategies have been actively investigated for TC in recent years. Much less research has been devoted to a third such strategy, *training label cleaning* (TLC), which consists in devising ranking functions that sort the original training examples in terms of how likely it is that the human annotator has mislabelled them. This provides a convenient means for the human annotator to revise the training set so as to improve its quality. Working in the context of boosting-based learning methods for multilabel classification we present three different techniques for performing TLC and, on three widely used TC benchmarks, evaluate them by their capability of spotting training documents that, for experimental reasons only, we have purposefully mislabelled. We also evaluate the degradation in classification effectiveness that these mislabelled texts bring about, and to what extent training label cleaning can prevent this degradation.

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*

General Terms: Algorithms, Design, Experimentation, Measurement

Additional Key Words and Phrases: Text classification, supervised learning, training label cleaning, synthetic noise, training label noise

ACM Reference Format:
Esuli, A. and Sebastiani, F. 2013. Improving text classification accuracy by training label cleaning. *ACM Trans. Inf. Syst.* 31, 4, Article 19 (November 2013), 28 pages.
DOI: http://dx.doi.org/10.1145/2516889

1. INTRODUCTION

In many application contexts involving supervised learning, labelled data may be scarce or expensive to obtain. In such situations, once we have trained the classifier with the available training data, if we discover that its accuracy is insufficient we are left with the issue of how to further improve it, under the constraint that the amount of human effort available to perform additional labelling is limited. One solution is to apply *active learning* techniques (see, e.g., [Cohn et al. 1994; Yu et al. 2008]), which rank a set of unlabelled examples in terms of how useful they are expected to be, once manually labelled, for retraining a (hopefully) better classifier; this allows the human annotators to concentrate on the most promising examples

This article is a substantially revised and extended version of a paper presented at the 2nd International Conference on the Theory of Information Retrieval (ICTIR'09).
The order in which the authors are listed is alphabetical; each author has given an equally important contribution to this work.
Authors' address: A. Esuli and F. Sebastiani, Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, Via Giuseppe Moruzzi 1, 56124 Pisa, Italy; email: {andrea.esuli, fabrizio.sebastiani}@isti.cnr.it.

DOI: http://dx.doi.org/10.1145/2516889

only. A second solution, orthogonal to the previous one, is to apply *semisupervised learning* techniques (see, e.g., [Chapelle et al. 2006; Sindhwani and Keerthi 2006; Zhu and Goldberg 2009]), which instead attempt to improve the accuracy of the classifier by leveraging unlabelled data (so that *no* additional labelling is needed). 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.

Both semisupervised learning and active learning have been widely studied in the context of text classification (TC) and other IR tasks involving supervised learning. There is instead a third route to solving the given problem that has been studied much less, namely, (computer-assisted) *training label cleaning* (TLC). Similarly to active learning, TLC techniques attempt to minimize the additional effort required from human annotators. However, while in active learning the human annotator is asked to label new *unlabelled* examples, in TLC she is required to inspect the manually *labelled* examples, looking for possibly mislabelled ones.

In the same way as a good active learning technique top-ranks the unlabelled examples that, once labelled, would prove the most informative for the training process, a good TLC technique top-ranks the training examples with the highest likelihood of being mislabelled. This allows the human annotator to improve the quality of the training set by inspecting the labels attached to the training examples, starting with the ones most likely to be erroneous, and working down the ranked list as s/he sees fit.

In this article we present three different techniques for performing TLC in TC and test them using a boosting-based supervised learner that generates confidence-rated predictions. The reason we are using this device is that, as will be apparent in Sections 3 and 4, it has two features that allow us to exemplify our TLC techniques particularly well, that is, (i) it allows for a notion of confidence in the classifier's predictions; and (ii) the classifier it generates is actually a classifier committee. We run our tests on three widely used TC benchmarks (two of which are very large), on which we evaluate our TLC techniques by their capability of spotting texts that we have purposefully mislabelled, for experimental purposes only, in the training set. We also evaluate the degradation in classification effectiveness that these mislabelled texts bring about, and to what extent training label cleaning can prevent this degradation.

The rest of the article is organized as follows. In Section 2 we explore more deeply the motivation behind training label cleaning as arising from practical application scenarios. Section 3 gives a brief description of the supervised learner that we use in our experiments, focusing on the features that are important for understanding the three TLC techniques we study; these latter are presented in Section 4. In Section 5 we describe the results of our tests in which, using three popular TC benchmarks, we evaluate these techniques by their capability of spotting texts that we have purposefully mislabelled, for experimental purposes only, in the training set. In the same section we also evaluate the beneficial effects that performing TLC may have on classification accuracy, by measuring the deterioration in classification accuracy that the insertion of these mislabelled training examples brings about. Section 6 discusses additional experiments aimed at verifying if and how much the results presented in the previous section are learner-independent (Section 6.1), and at verifying whether mutual independence of a committee of classifiers may help one of the three techniques presented (Section 6.2). Section 7 describes related research efforts, comparing them with the research described in this article. Section 8 concludes, pointing at avenues for further research.

2. MOTIVATION

Training label cleaning has to do with the presence of mislabelled items in the training data (see Figure 1 for two concrete examples). Of course, defining what counts as a

Text	Customer Service	Network Service	Tariff or Value
I keep having constant harassment ie six calls a day to change package or upgrade this has been going on for over three months even tho i keep telling your staff i'm ok and to STOP calling me!	Yes	No	Yes (*)
Iv had nothing but trouble with your network. I was totally mislead in the shop. Iv had double amount of money taken out of my account your mistake! Cant wait till my contract runs out i wouldnt recommend you 2 anyone!	No (*)	No (*)	Yes

Fig. 1. Two (manually) mislabelled training documents. The 1st column lists the textual content of the documents, while the other columns indicate some among the classes that the human annotators were meant to assign. The context was a customer satisfaction survey by a telecommunications company to whom these authors provide text classification services; the goal of the classification is to spot reasons for dissatisfaction with the company. Labels marked with a "(*)" seem clear mislabellings on the part of the (junior) annotators who performed the annotation.

mislabelled example is itself tricky, because labelling a document is a subjective activity. Different annotators a_1 and a_2 might in good faith disagree as to whether class c_j should be attributed or not to document d_i, a phenomenon called *intercoder disagreement*. This problem is exacerbated when the meaning of the class is not clearcut: for instance, it is not always clear if a given product review should be classified as Positive or Negative, or whether a given news article fits or not into class Lifestyles.

For the purpose of this article we will simply assume that when annotator a_2 at an organization inspects a set of labelled documents owned by the organization with the purpose of determining the quality of the labelling, and detects a label (originally attributed by annotator a_1) she disagrees with, this counts as a mislabelled document. In other words, we are assuming that the judgment of the annotator who performs the quality check is more important that that of the annotator who had originally labelled the documents. There are several reasons for this assumption.

(1) In several organizations it is often the case that the original labelling is performed by annotators (usually called "coders") as a part of their daily routine. In this routine, throughput (i.e., number of annotated documents per unit of time) is an essential factor. As a result, the coders' labelling activity may be error-prone.
Coders usually report to a more senior, superordinate "information specialist" who, in case labelled data are to be used for training an automated classifier (thereby generating a durable asset for the organization), may decide to double-check the labels originally attached by her coders. In this double-checking activity the resulting label quality is essential, while throughput is much less so. As a result, the judgments of the information specialist override those of the coders, and may be taken to be "the correct ones".
(2) It is hardly the case that the coders are the originators of the classification scheme; as a result, they may have an imperfect understanding of the true meaning of the classes, which may further negatively affect the quality of the labels they attribute. On the contrary the information specialist, being more senior, may either be the originator of the classification scheme or may be its maintainer (i.e., the one that decides when and if it needs revision in order to better suit the changing needs

of the organization), which means that her understanding of the meaning of the classes is certainly higher than that of the coders. This is a further reason why the labels she decides to attribute are more reliable than those attributed by the coders.

(3) When coders perform the original labelling they tend to work in "routine mode", sometimes with less-than-total commitment; an example is the (increasingly frequent) case in which annotation is performed via crowdsourcing (e.g., Mechanical Turk), yet another context in which fast turnaround (rather than label quality) is the main goal of the annotators [Grady and Lease 2010; Snow et al. 2008]. When an information specialist sets out to revise the labels attributed by her coders, she is instead likely to work in "double-checking mode", which is obviously conducive to better labelling decisions.

(4) If the user interface coders work with displays up-front the titles of a list of documents to be labelled, and only shows the body of a document if the annotator double-clicks on them, some coders will be happy to work from the titles alone, and this might be sufficient to correctly label most documents. However, for some documents the resulting labelling will be incorrect because the coders have not inspected the actual body of the document. This is another potential source of error, and one the information specialist will not be prone to if, working in double-checking mode, she does indeed inspect the body of the document.

(5) If the actual task is multilabel classification (see Section 3 for details), coders might attribute one or two labels to a document and stop exploring the classification scheme for other potential classes that might apply, thus generating several false negatives. It is the experience of these authors that, when classifying texts for market research applications (see Esuli and Sebastiani [2010]), coders make a conscious effort to avoid false positives but a much smaller one to avoid false negatives. An information specialist double-checking the labelled documents would likely not incur in the same mistake.

For all these reasons we may confidently assume that, when a set of labelled data is double-checked with the purpose of correcting possible mislabellings and using it to train a classifier, the labels decided upon by the annotator who performs the revision are the "correct" ones. This assumption justifies the experimental protocol we will adopt in Section 5, and ultimately justifies the very endeavour of training label cleaning.

3. PRELIMINARIES

This work attempts to identify good TLC techniques for *text classification* (aka *text categorization* – TC), and for *multilabel* text classification (MLTC) in particular. Given a set of textual documents D and a predefined set of *classes* (aka *categories*) $C = \{c_1, \ldots, c_m\}$, MLTC can be defined as the task of estimating an unknown *target function* $\Phi : D \times C \rightarrow \{-1, +1\}$, that describes how documents ought to be classified, by means of a function $\hat{\Phi} : D \times C \rightarrow \{-1, +1\}$ called the *classifier*[1]; 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. As usual, we accomplish MLTC by generating m independent binary classifiers $\hat{\Phi}^j : D \rightarrow \{-1, +1\}$, one for each $c_j \in C$, entrusted with the task of deciding whether a document belongs to class c_j or not. Note that we here do not address the related problem of TLC for single-label, multiclass classification, where each document needs to be assigned one

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

and only one out of $m > 2$ candidate classes; we leave the investigation of this task to future work.

As the learner for generating our classifiers we use our in-house boosting-based learner, called MP-BOOST [Esuli et al. 2006]; classifiers obtained via boosting have consistently shown high accuracy in several learning tasks, while at the same time having strong justifications from computational learning theory [Schapire and Freund 2012]. MP-BOOST is a variant of ADABOOST.MH [Schapire and Singer 2000] explicitly optimized for multilabel settings, which has been shown in [Esuli et al. 2006] to obtain considerable effectiveness improvements with respect to ADABOOST.MH.

MP-BOOST works by iteratively generating, for each class c_j, a sequence $\hat{\Phi}_1^j, \ldots, \hat{\Phi}_S^j$ of classifiers (called *weak hypotheses*). A weak hypothesis is a function $\hat{\Phi}_s^j : D \rightarrow \mathbf{R}$, where D is the set of documents and $\mathbf{R}$ is the set of the reals. The sign of $\hat{\Phi}_s^j(d_i)$ (denoted by $sgn(\hat{\Phi}_s^j(d_i))$) represents the binary prediction of $\hat{\Phi}_s^j$ on whether d_i belongs to c_j, that is, $sgn(\hat{\Phi}_s^j(d_i)) = +1$ (resp., -1) means that d_i is predicted to belong (resp., not to belong) to c_j. The absolute value of $\hat{\Phi}_s^j(d_i)$ (denoted by $|\hat{\Phi}_s^j(d_i)|$) represents instead the confidence that $\hat{\Phi}_s^j$ has in this prediction, with higher values indicating higher confidence.

At each iteration s MP-BOOST tests the effectiveness of the most recently generated weak hypothesis $\hat{\Phi}_s^j$ on the training set and uses the results to update a distribution D_s^j of weights on the training examples. The initial distribution D_1^j is uniform. At each iteration s all the weights $D_s^j(d_i)$ are updated, yielding $D_{s+1}^j(d_i)$, so that the weight assigned to an example correctly (resp., incorrectly) classified by $\hat{\Phi}_s^j$ is decreased (resp., increased). The weight $D_{s+1}^j(d_i)$ is thus a measure of how ineffective $\hat{\Phi}_1^j, \ldots, \hat{\Phi}_s^j$ have been in predicting whether d_i belongs to c_j or not (denoted by $\Phi^j(d_i)$). By using this distribution, MP-BOOST generates a new weak hypothesis $\hat{\Phi}_{s+1}^j$ that concentrates on the examples with the highest weights, that is, those that had proven harder to classify for the previous weak hypotheses.

The overall prediction on whether d_i belongs to c_j is obtained as a sum $\hat{\Phi}^j(d_i) = \sum_{s=1}^{S} \hat{\Phi}_s^j(d_i)$ of the predictions of the weak hypotheses. The final classifier $\hat{\Phi}^j$ is thus a *committee* of S classifiers, each classifier casting a weighted vote, with the vote being the binary prediction $sgn(\hat{\Phi}_s^j(d_i))$ and the weight being the confidence $|\hat{\Phi}_s^j(d_i)|$ of this prediction. For the final classifier $\hat{\Phi}^j$ too, $sgn(\hat{\Phi}^j(d_i))$ represents the binary prediction as to whether d_i belongs to c_j, while $|\hat{\Phi}^j(d_i)|$ represents the confidence in this prediction.

See Esuli et al. [2006] for more details on these and other aspects of MP-BOOST.

4. THREE TECHNIQUES FOR TRAINING LABEL CLEANING

In the following, by a *TLC technique* ρ we will mean a technique that, given a training set Tr and a class c_j, produces a ranking $r_j^\rho(Tr)$ in which the elements of Tr are sorted in decreasing order of the likelihood that their manually assigned label for c_j is wrong. Different techniques correspond to different ways of estimating this likelihood.

Note that, given a set of classes $C = \{c_1, \ldots, c_m\}$, these techniques thus generate m different and independent rankings of Tr; no unified and/or merged ranking is generated. This evokes an application scenario in which the user cleans the training data on a class-per-class basis, working on the classes for which the effectiveness is still low and disregarding the ones for which the effectiveness is already high enough, or

inspecting the different class-specific lists down to different depths depending on how much a given class needs improvement.

We now present three alternative TLC techniques.

4.1. The Confidence-Based Technique

For each $c_j \in C$, the first technique (that we dub *the confidence-based technique* – CONF, in short) consists in

(1) training a classifier $\hat{\Phi}^j$ on Tr;
(2) reclassifying the $d_i \in Tr$ by means of $\hat{\Phi}^j$;
(3) ranking the $d_i \in Tr$ in increasing order of their $\hat{\Phi}^j(d_i) \cdot \Phi^j(d_i)$ value.

The product $\hat{\Phi}^j(d_i) \cdot \Phi^j(d_i)$ is the *margin* of an example as computed by the final classifier. Note that, while $\Phi^j(d_i)$ is a value in $\{-1,+1\}$, $\hat{\Phi}^j(d_i)$ is a value in $(-\infty,+\infty)$, so $\hat{\Phi}^j(d_i) \cdot \Phi^j(d_i)$ is also in $(-\infty,+\infty)$; a positive (resp., negative) value of $\hat{\Phi}^j(d_i) \cdot \Phi^j(d_i)$ indicates a correct (resp., incorrect) binary prediction, while a high (resp., low) absolute value of The CONF technique thus corresponds to (a) top-ranking the examples $d_i \in Tr$ that $\hat{\Phi}^j$ has misclassified, in decreasing order of the confidence $|\hat{\Phi}^j(d_i)|$ with which $\hat{\Phi}^j$ has made its prediction, and (b) appending to this list the examples $d_i \in Tr$ that $\hat{\Phi}^j$ has correctly classified, in increasing order of the confidence $|\hat{\Phi}^j(d_i)|$. Obviously, different rankings are produced for the different $c_j \in C$. The rationale of this technique is that, if $\hat{\Phi}^j$ has misclassified a training example d_i with high confidence, this means that the label for c_j given to d_i by the human annotator is highly at odds with the labels for c_j that the human annotator has given to the other training examples, which indicates that the human annotator may well have mislabelled d_i for c_j.

4.2. The Nearest Neighbours Technique

For each $c_j \in C$, the second technique (that we dub *the nearest neighbours technique* – NN) consists in ranking the training examples in terms of how inconsistent their label for c_j is with the labels for c_j of their k nearest neighbours, for a predefined k. More formally, this technique consists in

(1) computing, for each $d_i \in Tr$, the value

$$\zeta(d_i, c_j) = \sum_{d_z \in Tr_k(d_i)} sim(d_i, d_z) \cdot \Phi^j(d_z), \tag{1}$$

where $sim(\cdot,\cdot)$ denotes a measure of similarity between documents and $Tr_k(d_i)$ denotes the k training examples most similar to d_i;
(2) ranking the $d_i \in Tr$ in increasing order of their $\zeta(d_i, c_j) \cdot \Phi^j(d_i)$ value.

For class c_j, the examples d_i with labels highly consistent with the labels of their neighbours will have high $\zeta(d_i, c_j) \cdot \Phi^j(d_i)$ values, which means that the ones with the lowest $\zeta(d_i, c_j) \cdot \Phi^j(d_i)$ values will be the ones with labels most dissimilar from those of their closest neighbours. Equation (1), of course, is that of the standard distance-weighted k-NN learner (see e.g., [Yang 1994, 1999]), the only difference being that, while in the standard case $\Phi^j(d_z)$ ranges on $\{0,1\}$, in our case it ranges on $\{-1,+1\}$, which means that neighbours with a negative label for c_j weigh negatively, instead of having no effect, on $\zeta(d_i, c_j)$. This variant of the k-NN learner is discussed in Galavotti et al. [2000].

The NN technique is similar to the CONF technique, and it might be seen as an instantiation of CONF where the Galavotti et al. [2000] variant of the k-NN classifier is used as the learning method. One difference between NN and CONF is that in NN the

sign of $\zeta(d_i, c_j)$ is, unlike $\hat{\Phi}^j(d_i)$ in CONF, not meant to represent the binary prediction of the classifier, since the decision threshold is not necessarily zero. A second, more significant difference is that in CONF the document d_i whose manually attributed label is being evaluated has also played the role of the training example in generating $\hat{\Phi}^j$, which is being used for the evaluation. This does not happen in the NN technique, since d_i is not a member of $Tr_k(d_i)$.

4.3. The Committee-Based Technique

For each $c_j \in C$, the third technique (that we dub *the committee-based technique* – COMM) consists in

(1) training a classifier $\hat{\Phi}^j$ on Tr;
(2) reclassifying Tr by means of $\hat{\Phi}^j$;
(3) ranking the $d_i \in Tr$ in increasing order of their

$$\Delta(\hat{\Phi}^j(d_i)) \cdot sgn(\hat{\Phi}^j(d_i)) \cdot \Phi^j(d_i)$$

value, where $\Delta(\hat{\Phi}^j(d_i))$ is a nonnegative real number that measures the *agreement* among the S members of $\hat{\Phi}^j$ on whether d_i belongs to c_j or not.

This technique is based on the intuition that the examples most in need of inspection are the ones which $\hat{\Phi}^j$ has misclassified (i.e., those such that $sgn(\hat{\Phi}^j(d_i)) \cdot \Phi^j(d_i) = -1$) *with the most widespread agreement* among its S members. In other words, if the information that a training example provides to the training process is so inconsistent with that provided by the other training data as to have the members of the generated classifier committee misclassify the example with widespread agreement, then it is likely that the example might be mislabelled. This technique will thus top-rank the training examples that the committee has misclassified and on which the S members of the committee agree most, mid-rank those on which there is disagreement, and bottom-rank those that the committee has classified correctly and on which the S members of the committee agree most.

The key difference between the first technique (CONF) and this technique is that here the confidence that a classifier committee has in a certain prediction is taken to coincide with the level of (weighted) agreement among its members, and not with the (weighted) sum of the individual opinions. As a measure of agreement among the S members of the committee we have chosen to use $\frac{1}{\sigma}$, where σ denotes *standard deviation*. This is a natural choice, given that the values $\hat{\Phi}^j_1(d_i), \ldots, \hat{\Phi}^j_S(d_i)$ are real numbers: standard deviation thus enables the measurement of (dis)agreement by taking into account not only the polarity $sgn(\hat{\Phi}^j_s(d_i))$ of each member's prediction, but also its confidence level $|\hat{\Phi}^j_s(d_i)|$, so that two members with views of different polarity are taken to disagree more if they are highly confident in their views, and less if they are not.[2]

4.4. The Distribution-Based Technique

Actually, there is a fourth technique (which we dub *the distribution-based technique* – DIS) that might come to mind [Abney et al. 1999, Section 5]. For each $c_j \in C$, this technique consists in (i) training the classifiers $\hat{\Phi}^j$ on Tr, and (ii) ranking the $d_i \in Tr$ in decreasing order of the $D^j_S(d_i)$ value that MP-BOOST has produced as a side effect of

[2]A previous version of this article [Esuli and Sebastiani 2009] contained a wrong, and ultimately unintuitive, version of this technique; the present article thus describes both a revised version of the technique and experiments run anew.

the learning process. The rationale of this technique is that, since the value $D_S^j(d_i)$ is a measure of how hard it has been, for the weak learners generated by the boosting iterations, to correctly reclassify d_i under c_j, the training examples that maximize $D_S^j(d_i)$ are the ones that have turned out the most difficult to make sense of during the boosting iterations. As a result, they are the ones whose label for c_j is most highly at odds with the label for c_j of the other training examples.[3]

The problem with the DIS technique is that it turns out to be equivalent to our first technique (CONF), in the sense that CONF and DIS always generate identical rankings, a fact that had never been noted in the literature.[4]

The only advantage that DIS provides over CONF is thus that there is no need to reclassify the training examples by means of $\hat{\Phi}^j$, since the information needed for ranking is already available after training has occurred.

4.5. A Note about Generality

Before discussing the experiments it is worthwhile noting that, although we have described these techniques in the context provided by a boosting-based learner which generates confidence-rated predictions, all of these techniques can be used also in connection with other learners. More specifically, CONF only needs the classifier to return a score of confidence in its own prediction, NN has no specific requirements, and COMM only require the classifier to consist of a committee of classifiers. Moreover, the discussed equivalence between CONF and DIS has the practical consequence of making available a technique equivalent to DIS to learners not based on boosting.

5. EXPERIMENTS

5.1. Experimental Protocol

In order to test our TLC techniques we use a standard MLTC dataset $\Omega = \langle Tr, Te \rangle$ split into a training set Tr and a test set Te. We assume that Tr and Te contain no mislabelled examples, and simulate the presence of mislabelled training examples by artificially "corrupting" a small number t of training examples; we call $\Pi = \frac{t}{|Tr|}$ the *corruption ratio*. In what follows, "corrupting a training example d_i for class c_j" means changing its label for c_j from positive to negative (in this case we call d_i a *fake negative for* c_j) or from negative to positive (a *fake positive*); by $\widehat{Tr}$ we denote the training set after corruption, and by FN_j and FP_j the sets of fake negatives for c_j and fake positives for c_j, respectively. We use the term "fake" instead of "false" in order to avoid overloading the latter term. We will also use the term "genuine" as the opposite of "fake."

We test two different corruption techniques, which we call *random corruption* (RC) and *targeted corruption* (TC). As the name implies, in RC the training examples to

[3]A similar technique would also be applicable when using SVMs as the learner, since SVMs assign, as a side-effect of training, a weight α_i to each training example that reflects how hard it has been for the generated classifier to reclassify it.

[4]We discovered this fact experimentally in the course of this work. A conversation with Robert Schapire, the inventor of boosting, later revealed that, while this phenomenon had never been observed before, an *a posteriori* justification can be found in the theory that underlies the ADABOOST.MH algorithm, of which MP-BOOST is a variant. Specifically, the reason is to be found in the fact that (as shown in the proof of Theorem 1 of [Schapire and Singer 1999]) $D_{S+1}^j(d_i) \propto \exp(-\Phi^j(d_i) \cdot \hat{\Phi}^j(d_i))$. Since CONF ranks the $d_i \in Tr$ in increasing order of $\Phi^j(d_i) \cdot \hat{\Phi}^j(d_i)$ value, since DIS ranks them in decreasing order of $D_{S+1}^j(d_i)$ value, and since $\exp(x)$ is a monotonically increasing function of its argument, it follows that the two rankings are the same. This property applies not only to ADABOOST.MH but also, straightforwardly, to MP-BOOST (see [Esuli et al. 2006, Section 3]).

Table I. Percentage p_{fn} of Corrupted Documents That Are Fake Negatives as a Function of the Corruption Ratio Π

	REUTERS-21578		RCV1-v2		OHSUMED	
Π	RC	TC	RC	TC	RC	TC
.001	0.7%	46.1%	2.8%	65.4%	0.2%	31.0%
.010	0.9%	19.3%	3.1%	43.9%	0.1%	8.4%
.050	0.8%	7.3%	3.1%	22.8%	0.1%	2.3%
.100	0.8%	4.3%	3.1%	14.9%	0.1%	1.2%

corrupt are picked at random from Tr. For simplicity, the same t training examples are corrupted for all classes $c_j \in C$. (This is absolutely equivalent to corrupting different training examples for the different classes, since our methods work on each of the classes independently.) TC is instead obtained by

(1) training the classifiers $\hat{\Phi}^j$ on Tr;
(2) reclassifying the $d_i \in Tr$ by means of them;
(3) ranking, for each $c_j \in C$, the reclassified examples in increasing order of the confidence $|\hat{\Phi}^j(d_i)|$ that $\hat{\Phi}^j$ had in classifying them;
(4) corrupting the t top-ranked ones.

The rationale of this technique is that the training examples that $\hat{\Phi}^j$ classifies with low confidence are more likely to be "borderline" examples for c_j; as a result these examples, should they be manually labelled, would have a high likelihood of being mislabelled (either due to lack of experience or to lack of adequate time) by a human annotator. In other words, while RC simulates the corruption of a training set that might derive from, say, lack of commitment on the part of the human annotators (e.g., in crowdsourced annotation), TC simulates the corruption that might derive from incomplete or imperfect understanding of the semantics of the classes. While it is true that what counts as a borderline example to a human annotator might not count as borderline to a text classification system (and vice versa), targeted corruption makes at least a substantive step in the direction of identifying examples that are more likely to get mislabelled by annotators.

Unlike in RC, in TC we allow different training examples to be corrupted for different classes $c_j \in C$, since the same document might be controversial, or "borderline", for one class but not for others.

Table I illustrates, for each of the datasets we use in this article (see Section 5.3 for a detailed description of them), for each corruption technique and for each corruption ratio, the percentage $p_{fn} = \frac{\sum_{j=1}^{m} |FN_j|}{\sum_{j=1}^{m} |FN_j + FP_j|} \cdot 100\%$ of corrupted documents that are fake negatives; obviously, $p_{fp} = 100\% - p_{fn}$. For random corruption this percentage tends to be fairly constant across the different corruption ratios (although different across datasets), which is obvious since it tends to coincide with the average class frequency of the entire dataset.

Table I also shows that, for a given corruption ratio, p_{fn} is always higher (and usually much higher) for TC than for RC; for example, for REUTERS-21578 and $\Pi = .001$ the value of p_{fn} is 0.7% for RC and 46.1% for TC. The reason is that TC corrupts not random but "borderline" examples, and these proportionally include many more positives than random examples do.

The same table also shows that in targeted corruption, while fake positives tend to outnumber fake negatives, this tendency is increasingly marked as the corruption rate increases; for example, for REUTERS-21578 the value of p_{fn} is 46.1% for $\Pi = .001$ but only 4.3% for $\Pi = .100$. This is due to the fact that, as in most text classification

datasets, the number of genuinely positive examples is much smaller than the number of genuinely negative examples. As a result, as the number of documents to corrupt increases the number of positive documents that can be corrupted cannot increase proportionally.

5.2. Effectiveness Measures

In order to determine which among the three TLC techniques of Section 4 is the best we will measure how good each technique is at ranking the $d_i \in \widehat{Tr}$ in such a way that the corrupted training examples are placed at the top of the ranking. To this end, it seems natural to adopt one of the measures routinely used for evaluating ad-hoc (ranked) retrieval. Of course, ad-hoc retrieval is all about ranking the "good" (i.e., relevant to the information need) examples higher than the bad ones, while TLC aims at ranking the "bad" (i.e., mislabelled) examples higher than the good ones; but this is obviously inessential.

As a measure of ranking quality we will choose *mean average precision* (MAP), which in our context is defined as follows. Let $r_j^\rho(\widehat{Tr})$ be the ranking for class c_j realized according to TLC technique ρ, of the corrupted training set $\widehat{Tr}$, and let $[r_j^\rho(\widehat{Tr})]_k$ be a binary predicate that returns 1 if the example at the k-th position in $r_j^\rho(\widehat{Tr})$ is corrupted for class c_j, and 0 otherwise. We define the *precision at n of* $r_j^\rho(\widehat{Tr})$ as

$$P_n(r_j^\rho(\widehat{Tr})) = \frac{1}{n}\sum_{k=1}^{n}[r_j^\rho(\widehat{Tr})]_k\,. \tag{2}$$

We then define the *average precision of* $r_j^\rho(\widehat{Tr})$ as

$$AP(r_j^\rho(\widehat{Tr})) = \frac{\sum_{k=1}^{|\widehat{Tr}|} P_k(r_j^\rho(\widehat{Tr}))\cdot[r_j^\rho(\widehat{Tr})]_k}{\sum_{k=1}^{|\widehat{Tr}|}[r_j^\rho(\widehat{Tr})]_k}. \tag{3}$$

The *mean average precision* (MAP) of TLC technique r on $\widehat{Tr}$ is finally defined as

$$MAP(r(\widehat{Tr})) = \frac{1}{|C|}\sum_{c_j\in C} AP(r_j^\rho(\widehat{Tr})). \tag{4}$$

Aside from a measure of TLC effectiveness we will also need a measure of MLTC effectiveness, so as to determine which effectiveness gains in classification can be obtained if TLC is performed. As a MLTC effectiveness measure that combines the contributions of *precision* (π) and *recall* (ρ) we have used the well-known F_1 function, defined as $F_1 = \frac{2\pi\rho}{\pi+\rho} = \frac{2TP}{2TP+FP+FN}$, where TP, FP, and FN stand for the numbers of true positives, false positives, and false negatives, respectively. Note that F_1 is undefined when $TP = FP = FN = 0$; in this case we take F_1 to equal 1, since the classifier has correctly classified all documents as negative examples. We compute both microaveraged F_1 (denoted by F_1^μ) and macroaveraged F_1 (F_1^M). F_1^μ is obtained by (i) computing the category-specific values TP_i, FP_i and FN_i, (ii) obtaining TP as the sum of the TP_i's (same for FP and FN), and then (iii) applying the $F_1 = \frac{2TP}{2TP+FP+FN}$ formula. F_1^M is obtained by first computing the category-specific F_1 values and then averaging them across the c_j's.

Section 5.4 reports the results of our experiments with the three TLC techniques of Section 4, each tested under two different corruption techniques, four different corruption ratios, and three different datasets.

5.3. The Datasets

In our experiments we have used the REUTERS-21578, RCV1-V2, and OHSUMED datasets.

REUTERS-21578 is probably still the most widely used benchmark in MLTC research.[5] 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; in our experiments we have restricted our attention to the 115 categories with at least one positive training example. The average number of categories per training document is 1.005, the number of positive training examples per category ranges from a minimum of 1 to a maximum of 2877, and the average balance ratio[6] in the training set is $B = .017$.

REUTERS CORPUS VOLUME 1 version 2 (RCV1-V2)[7] is a more recent MLTC benchmark made available by Reuters and consisting of 804,414 news stories produced by Reuters from 20 Aug 1996 to 19 Aug 1997. In our experiments we have used the "LYRL2004" split, defined in Lewis et al. [2004], in which the (chronologically) first 23,149 documents are used for training and the other 781,265 are used for testing. Of the 103 "Topic" categories, in our experiments we have restricted our attention to the 101 categories with at least one positive training example. Consistently with the evaluation presented in [Lewis et al. 2004], (i) also categories placed at internal nodes in the hierarchy are considered in the evaluation, and (ii) as positive training examples of these categories we use the union of the positive examples of their subordinate nodes, plus their "own" positive examples. The average number of categories per training document is thus 3.184, the number of positive training examples per category ranges from a minimum of 2 to a maximum of 10786, and the average balance ratio in the training set is $B = .063$.

The OHSUMED test collection [Hersh et al. 1994] consists 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; 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" part of the MeSH index terms assigned to the entries.[8] Concerning this latter point, we have restricted our experiments to the

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

[6] We define the *average balance ratio* in the training set as the value

$$B = (1 - \frac{1}{|C|} \sum_{c_i \in C} |\frac{|Tr_i^+| - |Tr_i^-|}{|Tr|}|),$$

where $|Tr_i^+|$ (resp., $|Tr_i^-|$) is the number of positive (resp., negative) training examples for class c_i. The average balance ratio is $B = 1$ only if all classes are perfectly balanced (i.e., they have an equal number of positive and negative training examples) and is 0 if all classes are perfectly imbalanced (i.e., each of them has either no positive training examples or – uncharacteristically – no negative training examples).

[7] http://trec.nist.gov/data/reuters/reuters.html

[8] MeSH index terms consist of a main heading optionally qualified with subheadings and/or importance markers. For example, in the MeSH index term Oxytocin/*AA/GE, the main heading is Oxytocin. Several MeSH index terms may be assigned to the same entry, which means this is a multilabel TC task.

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 average number of categories per training document is 0.130 (many training documents are unlabelled, and just serve as negative training examples for all classes), the number of positive training examples per category ranges from a minimum of 1 to a maximum of 4075, and the average balance ratio in the training set is $B = .003$.

There are three main reasons why we have chosen exactly these datasets:

(1) All these datasets are publicly available and very widely used in text classification research, which allows other researchers to easily replicate the results of our experiments.
(2) RCV1-v2 and OHSUMED are among the largest datasets used to date in text classification research, which lends robustness to our results.
(3) For at least two (REUTERS-21578 and RCV1-v2) of our chosen datasets, the assumption that the uncorrupted training sets do not contain mislabelled training examples (see Section 5.1) is probably more justified than for any other text classification datasets available in research, since the document labelling of these datasets has undergone a lot of quality checking from Reuters editors and text classification researchers alike [Lewis 2004; Lewis et al. 2004].

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 as far as MP-BOOST is concerned. Documents are instead weighted (by standard cosine-normalized *tfidf*) (i) for the sake of computing the interdocument similarity values required by the *NN* technique of Section 4.2, and (ii) for the further experiments with the SVM learner that we will later describe in Section 6.

5.4. Results and Discussion

5.4.1. Evaluating the Quality of Training Label Cleaning. Table II reports MAP values obtained by ranking the corrupted training sets by means of the three TLC techniques (CONF, NN, COMM); the meaning of the fourth column (labelled BAG) will be made clear in Section 6.2. For each tested corpus, (a) we report results for the full set of classes, and (b) from these results we single out those concerning the 30 most infrequent classes and report them separately. (The meaning of the rows labelled "OHSUMED-S" will be clarified in Section 5.4.2.) The reason we pay special attention to the most *infrequent* classes (unlike many researchers who often report results only for the most frequent classes of a collection) is that they are usually the classes for which standard supervised learning techniques produce the lowest classification effectiveness. This means that they are the classes most in need of effectiveness improvements, by TLC or other techniques: a user might typically engage in TLC for these highly problematic classes, and disregard the classes for which high enough accuracy has already been achieved.

In all the experiments MP-BOOST has been run with a number S of iterations fixed to 1,000. For the NN technique, as the $sim(\cdot, \cdot)$ measure of inter-document similarity we have used the inner product of the cosine-normalized *tfidf* vectors of the two documents. For the same technique we have used the value $k = 45$, since in using

Table II.

Mean average precision (MAP) of the four TLC techniques (CONF, NN, COMM, BAG) on the full set of classes (top 4 rows) and on the 30 most infrequent classes (bottom 4 rows) of REUTERS-21578, RCV1-v2, OHSUMED, and OHSUMED-S. **Boldface** indicates a statistically significant (two-tailed paired t-test on average precision value over categories, $P < 0.01$) best performer for a given combination of corruption ratio (Π), corruption method, and dataset.

			Random corruption				Targeted corruption			
		Π	CONF	NN	COMM	BAG	CONF	NN	COMM	BAG
REUTERS-21578	FULL SET	.001	**.596**	.458	.110	.213	**.510**	.369	.107	.230
		.010	.653	**.771**	.367	.427	**.608**	.525	.245	.291
		.050	**.968**	.907	.841	.790	**.677**	.621	.320	.340
		.100	.973	.961	.900	.850	**.665**	.634	.422	.457
	30 INFR	.001	.748	**.790**	.231	.423	.648	**.681**	.091	.235
		.010	.674	**.966**	.490	.531	.581	**.670**	.181	.287
		.050	.982	.992	.842	.803	.647	**.701**	.281	.361
		.100	.981	.985	.903	.861	.673	.651	.431	.439
RCV1-v2	FULL SET	.001	.232	.238	.135	.130	**.357**	.082	.039	.278
		.010	**.752**	.542	.418	.380	**.519**	.376	.100	.384
		.050	**.927**	.777	.790	.689	**.672**	.512	.321	.430
		.100	**.945**	.865	.849	.803	**.658**	.593	.369	.461
	30 INFR	.001	.222	.225	.112	.108	**.323**	.101	.046	.181
		.010	**.702**	.476	.480	.391	**.435**	.375	.201	.297
		.050	**.896**	.716	.750	.650	**.608**	.427	.374	.381
		.100	**.919**	.845	.789	.720	**.613**	.523	.401	.413
OHSUMED	FULL SET	.001	**.474**	.422	.241	.230	**.438**	.308	.405	.392
		.010	**.370**	.291	.194	.221	**.767**	.609	.572	.432
		.050	**.331**	.264	.170	.197	**.758**	.620	.550	.467
		.100	**.270**	.232	.176	.199	**.695**	.635	.419	.403
	30 INFR	.001	**.490**	.418	.268	.218	**.667**	.343	.404	.383
		.010	**.387**	.310	.249	.211	**.790**	.653	.586	.443
		.050	**.362**	.290	.234	.171	**.773**	.674	.547	.466
		.100	**.291**	.242	.229	.169	**.754**	.680	.429	.411
OHSUMED-S	FULL SET	.001	.461	.475	.328	.301	**.403**	.365	.177	.104
		.010	.667	.688	.566	.610	**.576**	.549	.413	.401
		.050	**.917**	.870	.856	.803	.651	.642	.521	.507
		.100	**.948**	.898	.893	.841	**.669**	.650	.527	.509
	30 INFR	.001	.544	.555	.419	.423	**.564**	.526	.224	.209
		.010	.832	**.854**	.738	.740	.631	.643	.351	.339
		.050	**.953**	.915	.883	.831	.671	.669	.485	.458
		.100	**.969**	.933	.921	.873	.693	.683	.526	.513

k-NN as a learner for TC Yang [1999], using REUTERS-21578, has found this value to yield the best effectiveness (and has found negligible differences among values of $k \in [30, 65]$).[9]

[9]In operational conditions, if one had to pick the optimal value of k for the NN technique, one might well classify all the training documents via the k-NN classifier (using each training document as test and the other training documents as training), compute the resulting classification accuracy, do all this for various values of k, pick the value of k that has given the best classification accuracy, and use this value of k for performing the cleaning, on the assumption that what works best for classification also works best for training data cleaning. This means that, despite appearances to the contrary, the given protocol of choosing a value of k that has proven optimal in classification experiments on the very same dataset we use is legitimate.

A "trivial" baseline to the results of Table II is the expected MAP value of the *random ranker* (RR), that is, the algorithm which generates random document rankings. As proven in Resta [2012], the expected AP value of the RR is equal to

$$AP(RR(\Omega)) = \frac{t_j - 1}{n - 1} + \frac{(n - t_j)H_n}{n(n-1)}. \quad (5)$$

In our setting, t_j corresponds to the number of documents which are mislabelled for c_j and n to the number of documents that need to be ranked (i.e., $n = |Tr|$); H_n denotes the n-th harmonic number (i.e., $H_n = \sum_{k=1}^{n} \frac{1}{k}$). In the hypothesis (which is indeed always assumed true in our experiments) that the number t_j of mislabelled documents is the same for all classes $c_j \in C$, this is obviously also the expected *mean* AP value of the RR. Actual computation of this formula shows that $MAP(RR(\Omega))$ is approximated by $\frac{t}{n}$ (and in an especially accurate way for large values of n), which in our case coincides with the corruption ratio Π. Since for all of our datasets and corruption ratios approximating Equation (5) to the third decimal digit exactly yields Π, the first column of Table II also *de facto* indicates the trivial baseline for the experiments in the corresponding row.

There are several insights that can be gained from observing the results of Table II. The first observation is that, since picking training examples at random is the only method one can adopt when wanting to perform TLC, unless equipped with a specific TLC technique such as CONF, NN or COMM, the improvement that the three TLC techniques display in Table II over the baseline of Column 1 is considerable.

A second observation is that, with few exceptions and all other things being equal, each technique performs better for random corruption than for targeted corruption. This is intuitive, since mislabelled training examples inserted at random in the training set tend to be easier to spot, since their labels tend to be more blatantly wrong; conversely, targeted corruption alters the label of examples which are borderline anyway, and their altered label is thus much more difficult to recognize as such for *any* technique. By averaging all the figures contained in Table II we obtain a MAP value of .554 for random corruption and a value of .487 for targeted corruption.[10] (We will informally call the values resulting from such averages the "TII-average MAP values.")

The third observation is that, among the three competing TLC techniques, CONF is a clear winner and COMM is a clear loser. In the vast majority of testing situations, CONF is either superior (in a statistically significant sense, two-tailed paired t-test on average precision value over categories, $P < 0.01$) to both other techniques, or is not inferior (also in a statistically significant sense) to any of them. The COMM technique obtains instead, in almost all situations, results inferior (and often radically so) to CONF and NN. The TII-average MAP values are .625 for CONF, .549 for NN, and .388 for COMM. The CONF technique tends to be the better one on the RCV1-V2 and OHSUMED datasets, while the situation is less clearcut on REUTERS-21578. All in all, both techniques turn out to be respectable contenders, often achieving (sometimes surprisingly) high MAP values in absolute terms. We conjecture that the reason for the bad performance of COMM may be found in the fact that MP-BOOST generates a committee of classifiers that are *not* independent of each other. Indeed, each member $\hat{\Phi}_s^j$ of the committee strongly depends on the previously generated member $\hat{\Phi}_{s-1}^j$, since the former is generated according to the distribution resulting from applying $\hat{\Phi}_{s-1}^j$ to Tr. As a consequence, agreement is probably not something one could reasonably

[10] In the computation of these averages, and of other similar ones that will be discussed in the rest of this article, we disregard the values from the rows marked OHSUMED-S since, as will be apparent from Section 5.4.2, they would duplicate values from the rows marked OHSUMED and would thus bias the final results.

expect from the members of *this* kind of committee, since sharp disagreement may derive from reasons different from a bad label, such as the different emphasis that the different members place, by construction, on a given training example.

A fourth insight we can gain by looking at Table II is that MAP tends to increase with the corruption ratio Π, and may reach extremely high values for high values of Π. The TII-average MAP values are 0.294 for $\Pi = .001$ (i.e., 0.1% of the documents are corrupted), 0.427 for $\Pi = .010$, 0.534 for $\Pi = .050$, and 0.552 for $\Pi = .100$. These high values of MAP are not a trivial result since, although higher corruption ratio means that there are many mislabelled examples, this does not make them easier to spot: possibly quite the contrary, since the ratio between correctly labelled and mislabelled documents decreases, which means that the mislabelled documents are less inconsistent with the rest. High MAP values for high corruption ratio is very good news, since this means that if we have reasons to believe that our training set is extremely low-quality, we know that our time in cleaning it will not be wasted, since these techniques will place many of the bad examples near the top of the ranking.[11] Note that, when the corruption ratio is high and the class is infrequent, the number of corrupted documents may well exceed the number of positive instances of the class. (For instance, the 30 most infrequent classes of REUTERS-21578 have at most 2 positive training examples each (out of the total 9,603 training examples), which means that this problem occurs even for a modest corruption ratio such as $\Pi = .001$.) As a result, in the corrupted training set the number of fake positives may well exceed the number of genuine positives. In this case, the good MAP results are due to the discriminating power of the (genuine) negative examples; for instance, the NN technique spots many fake positives since each of them lies, in the space of examples, close to many negative examples, which means that its ζ score (see Equation (1)) is extremely low. Similar arguments apply to the CONF and COMM techniques. We can also observe that there is no radical or systematic difference between the way our techniques work on the full set of classes and the way they work on the 30 most infrequent classes. While substantial differences are observed for some specific combinations (e.g., CONF on REUTERS-21578 corrupted via random corruption with $\Pi = .001$), these differences are not systematic. To witness, the TII-average MAP values are .441 for the full set and .463 for the 30 most infrequent classes.

5.4.2. Strange News from Planet OHSUMED. As can be noticed by looking at Table II, when perturbed via random corruption the OHSUMED collection displays a qualitatively different behaviour from the other two collections; in fact, while MAP tends to increase with Π in all other cases (i.e., when targeted corruption is used, or when the other two collections are involved), it tends to decrease when random corruption is applied to OHSUMED.

We conjecture that this strange phenomenon might be due to the fact that OHSUMED exhibits a much smaller average balance ratio ($B = .003$) than the other two collections ($B = .017$ for REUTERS-21578, $B = .063$ for RCV1-V2). This depends on the fact that its training set contains a huge amount of documents (more than 93% of the entire training set) that do not belong to any class, and that originally belonged to other subtrees of the MeSH tree.

In order to test this conjecture we have run additional experiments on a collection (that we here call OHSUMED-S) obtained from OHSUMED by retaining only the

[11]Note that a higher corruption ratio means higher *a priori* probability that MAP is high, as witnessed by the fact that the expected MAP of the random ranker grows linearly with the corruption ratio. But this factor alone does not justify the very high MAP values we reach for high corruption ratios, as shown by the fact that the MAP of our techniques grows with the corruption ratio much faster than the expected MAP of the random ranker.

documents with at least one label in the HD subtree. The OHSUMED-S training set thus contains 12,358 documents, and its average balance ratio in the training set is $B = .020$, much higher than the one of the full OHSUMED ($B = .003$) and similar to the one of the REUTERS-21578 collection ($B = .017$).

The results of these additional experiments, displayed in the last eight rows of Table II, essentially confirm our hypothesis, since they are qualitatively similar to those observed for REUTERS-21578 and RCV1-v2, and sharply different from those for the entire OHSUMED. The same similarity among REUTERS-21578, RCV1-v2, and OHSUMED-S (and their dissimilarity from OHSUMED) will be observed from Tables III and IV, to be discussed in the following sections

5.4.3. Evaluating the Effects of Noise. Table III reports instead the micro- and macro-averaged F_1 values obtained by the classifiers generated via MP-BOOST before and after corruption, that is, after training either on the uncorrupted or on the corrupted training sets. This is an indication of the improvement in classification effectiveness one might obtain by *fully* cleaning the original training set when it contains noise at the corruption ratios indicated. Results are reported for the full set of classes and for the 30 most infrequent classes of our two datasets.

One insight that this table enables is that random corruption is usually more damaging to effectiveness than targeted corruption, and this fact tends to become evident as the corruption rate increases. That targeted corruption may have less disruptive effects can be explained by the fact that TC introduces mislabellings on documents that are likely borderline examples anyway, that is, documents that two human annotators might legitimately label in different ways. Mislabelling them may hurt classification accuracy in the thin region of document space close to the surface that separates the positives from the negatives, but is not likely to affect accuracy elsewhere. Conversely, random corruption may have effects anywhere in document space, and may seriously mislead the classifiers even on cases that would be clearcut otherwise.

A second fact that immediately jumps to the eye is that the decrease in effectiveness deriving from corruption is considerable even for very modest corruption rates (e.g., $\Pi = .001$, i.e., 0.1%), and already becomes disastrous for slightly less modest ones (e.g., $\Pi = .010$). For instance, for a $\Pi = .001$ targeted corruption rate (which corresponds to roughly 10 mislabelled training documents in a training set of more than 9,600 documents), removing the mislabellings from the REUTERS-21578 training set makes F_1^{μ} jump from .821 to .852 for the full set of classes. This is a 3% relative improvement, that in the '90s has taken years of improvement in TC technology to achieve. This shows that one mislabelled document in a thousand can single-handedly defy the efforts of many TC researchers at improving effectiveness.

While the percentages of deterioration are high throughout the table, there seems to be a correlation between deterioration and average balance ratio of the training set. In fact, recall from Section 5.3 that this ratio is $B = .003$ for OHSUMED, $B = .017$ for REUTERS-21578, and $B = .063$ for RCV1-v2. The three datasets are in the same order when it comes to deterioration; for example, the deterioration in F_1^M at $\Pi = .001$ (full set of classes, targeted corruption) is -46.3% for OHSUMED, -26.1% for REUTERS-21578, and -16.3% for RCV1-v2. The fact that, for all three datasets, the deterioration radically increases when we move to the set of the 30 most infrequent categories, reinforces the point. This may be explained by the fact that learning a classifier in the presence of strong imbalance (i.e., few positive training examples) is hard, and even a moderate corruption ratio can be disruptive on the effectiveness of the classifier when the positive training examples are, relatively to the entire training set, few.

The third insight that Table III suggests is that the deterioration in effectiveness resulting from corruption is larger for the more infrequent classes. For instance, in

Table III.

Micro- and macro-averaged F_1 values of the classifiers generated by MP-BOOST for the full set of classes (5 top rows) and for the 30 most infrequent classes (5 bottom rows) of REUTERS-21578, RCV1-v2, OHSUMED, and OHSUMED-S after corruption, as a function of the corruption ratio Π. Percentages indicate the relative deterioration in effectiveness with respect to the uncorrupted training set, which corresponds to the $\Pi = .000$ (no corruption) rows. The values in the third column are also a (trivial) baseline for the experiments in the corresponding row.

			Random corruption				Targeted corruption			
			F_1^μ		F_1^M		F_1^μ		F_1^M	
REUTERS-21578	FULL SET	.000	.852	(0.0%)	.606	(0.0%)	.852	(0.0%)	.606	(0.0%)
		.001	.822	(−3.5%)	.356	(−41.3%)	.821	(−3.6%)	.448	(−26.1%)
		.010	.583	(−31.6%)	.227	(−62.5%)	.632	(−25.8%)	.254	(−58.1%)
		.050	.138	(−83.8%)	.074	(−87.8%)	.209	(−75.5%)	.094	(−84.5%)
		.100	.064	(−92.5%)	.047	(−92.2%)	.116	(−86.4%)	.061	(−89.9%)
	30 INFR	.000	.373	(0.0%)	.245	(0.0%)	.373	(0.0%)	.245	(0.0%)
		.001	.190	(−49.1%)	.114	(−53.5%)	.139	(−62.7%)	.137	(−44.1%)
		.010	.038	(−89.8%)	.036	(−85.3%)	.056	(−85.0%)	.052	(−78.8%)
		.050	.004	(−98.9%)	.004	(−98.4%)	.011	(−97.1%)	.011	(−95.5%)
		.100	.002	(−99.5%)	.002	(−99.2%)	.006	(−98.4%)	.005	(−98.0%)
RCV1-v2	FULL SET	.000	.572	(0.0%)	.423	(0.0%)	.572	(0.0%)	.423	(0.0%)
		.001	.557	(−2.6%)	.368	(−13.0%)	.558	(−2.4%)	.354	(−16.3%)
		.010	.348	(−39.2%)	.224	(−47.0%)	.441	(−22.9%)	.324	(−23.4%)
		.050	.105	(−81.6%)	.096	(−77.3%)	.211	(−63.1%)	.160	(−62.2%)
		.100	.050	(−91.3%)	.064	(−84.9%)	.137	(−76.0%)	.107	(−74.7%)
	30 INFR	.000	.164	(0.0%)	.062	(0.0%)	.164	(0.0%)	.062	(0.0%)
		.001	.102	(−37.8%)	.044	(−29.0%)	.038	(−76.8%)	.035	(−43.5%)
		.010	.025	(−84.8%)	.024	(−61.3%)	.063	(−61.6%)	.039	(−37.1%)
		.050	.006	(−96.3%)	.005	(−91.9%)	.015	(−90.9%)	.014	(−77.4%)
		.100	.005	(−97.0%)	.003	(−95.2%)	.010	(−93.9%)	.008	(−87.1%)
OHSUMED	FULL SET	.000	.624	(0.0%)	.508	(0.0%)	.624	(0.0%)	.508	(0.0%)
		.001	.340	(−45.5%)	.235	(−53.7%)	.403	(−35.4%)	.273	(−46.3%)
		.010	.129	(−79.3%)	.072	(−85.8%)	.129	(−79.3%)	.110	(−78.3%)
		.050	.010	(−98.4%)	.007	(−98.6%)	.070	(−88.8%)	.063	(−87.6%)
		.100	.002	(−99.7%)	.001	(−99.8%)	.021	(−96.6%)	.019	(−96.3%)
	30 INFR	.000	.465	(0.0%)	.327	(0.0%)	.465	(0.0%)	.327	(0.0%)
		.001	.118	(−74.6%)	.080	(−75.7%)	.134	(−71.1%)	.139	(−57.6%)
		.010	.061	(−86.9%)	.037	(−88.7%)	.017	(−96.4%)	.029	(−91.0%)
		.050	.003	(−99.4%)	.002	(−99.4%)	.008	(−98.3%)	.007	(−97.9%)
		.100	.001	(−99.8%)	.001	(−99.7%)	.001	(−99.8%)	.001	(−99.7%)
OHSUMED-S	FULL SET	.000	.707	(0.0%)	.478	(0.0%)	.707	(0.0%)	.478	(0.0%)
		.001	.539	−(23.8%)	.422	−(11.6%)	.526	−(25.6%)	.396	−(17.1%)
		.010	.459	−(35.1%)	.279	−(41.7%)	.431	−(39.0%)	.257	−(46.3%)
		.050	.215	−(69.6%)	.141	−(70.6%)	.171	−(75.8%)	.147	−(69.3%)
		.100	.177	−(75.0%)	.119	−(75.2%)	.093	−(86.8%)	.095	−(80.1%)
	30 INFR	.000	.320	(0.0%)	.314	(0.0%)	.320	(0.0%)	.314	(0.0%)
		.001	.093	−(70.9%)	.284	−(9.6%)	.107	−(66.6%)	.294	−(6.5%)
		.010	.022	−(93.2%)	.045	−(85.6%)	.032	−(90.0%)	.071	−(77.5%)
		.050	.008	−(97.4%)	.008	−(97.4%)	.015	−(95.4%)	.013	−(95.8%)
		.100	.002	−(99.4%)	.002	−(99.3%)	.006	−(98.2%)	.005	−(98.3%)

the REUTERS-21578 case discussed earlier ($\Pi = .001$), while the deterioration in F_1^μ brought about by targeted corruption for the full set of classes is from .852 to .821 (-3.7%), for the 30 most infrequent classes the deterioration is from .373 to .139 (-62.8%)! The same effect may be observed by looking at the F_1^M results (instead of F_1^μ) across the entire table: the improvements resulting from performing TLC are much larger for F_1^M than for F_1^μ, due to the fact that F_1^μ is not much influenced by the results on the infrequent classes, while F_1^M is. It is not hard to see why the effect of even a few mislabelled training examples on the classification accuracy for infrequent classes can be so large. Given a class with very few positive training examples, mislabelling even one or a handful negatives as positives can severely alter the set of positive training examples, while mislabelling even one or a handful of positives as negatives has the double effect of depleting the already slim set of positive examples and confusing the learner, by presenting it with negative training documents that are very similar to the remaining positive ones. It is also likely that, given a class with few positive training examples, the presence of corrupted training examples close to the separating surface generates so much uncertainty in the classifier that it may often decide to vote negative so as to maximize accuracy. This may often be detrimental to F_1, since zero recall means $F_1 = 0$.

Similar observations also hold for random corruption and for the other two datasets. For reasons of space we do not separately report the results on the $(|C| - 30)$ most frequent classes of our two datasets. In a nutshell, on these classes the decrease in F_1^μ is very similar to the decrease on the full set of classes (since F_1^μ is mostly influenced by the behaviour on the most frequent classes), while the decrease in F_1^M is smaller than the decrease in the full set of classes (since F_1^M is equally influenced by all the classes in C).

5.4.4. Evaluating the Effects of Cleaning. Note that Table III only gives us a picture of the improvement that might be obtained by cleaning the *entire* training set. Aside from probably being too expensive in many real-world situations, this is something that would defy the purpose of the TLC techniques we have presented. A study should thus be performed that, for any combination of TLC technique, corruption method, corruption ratio, and dataset, plots the effectiveness of the classifiers generated after TLC has been performed, as a function of K, the number of top-ranked training examples that the human annotator has inspected for misclassifications. This is obviously a daunting experimentation, since for each such combination and each value of K the classifiers should be retrained from scratch and the test examples should be relabelled anew. More modestly, in Table IV we provide a sample such experiment, in which for the two different corruption methods, four corruption ratios, and all our three datasets, we test the effectiveness values resulting from

(1) ranking the training documents via the CONF technique;
(2) "uncorrupting" the corrupted documents found at the top $K = \frac{|Tr|}{100}$ positions (i.e., 1% of the total) in the ranking;
(3) training the classifiers on this partially cleaned training set;
(4) classifying the test set via the classifiers thus generated.

For instance, on REUTERS-21578 with targeted corruption and $\Pi = .001$, the MAP value of .510 that CONF obtains (see Table II) guarantees that F_1^μ, which corruption had reduced from .852 to .821, jumps back to .850, and that F_1^M, which corruption had reduced from .606 to .448, jumps back to .498. What we may also observe from Table IV is that, unsurprisingly, high values of P_K (precision at K) lead to higher

Table IV.

F_1^μ and F_1^M values obtained on the full set of classes (top 5 rows) and on the 30 most infrequent classes (bottom 5 rows) of our four datasets, with classifiers trained before or after performing TLC on the corrupted training sets by means of the CONF technique with $K = \frac{|Tr|}{100}$ (i.e., only the top 1% training documents are cleaned); the value of P_K (i.e., the precision at $K = \frac{|Tr|}{100}$ that had been obtained by CONF) is shown in each case. The "before cleaning" results are taken from Table III. **Boldface** indicates a statistically significant improvement (two-tailed paired t-test on F_1 value over categories, $P < 0.01$).

			Random corruption					Targeted corruption				
		Π	P_K	F_1^μ		F_1^M		P_K	F_1^μ		F_1^M	
				before	after	before	after		before	after	before	after
REUTERS-21578	FULL SET	.000	—	.852	.852	.606	.606	—	.852	.852	.606	.606
		.001	.090	.822	**.847**	.356	**.468**	.082	.821	**.850**	.448	**.498**
		.010	.520	.583	**.749**	.227	**.399**	.579	.632	**.780**	.254	**.412**
		.050	.910	.138	**.607**	.074	**.252**	.783	.209	**.632**	.094	**.312**
		.100	.884	.064	**.173**	.047	**.090**	.761	.116	**.213**	.061	**.208**
	30 INFR	.000	—	.373	.373	.245	.245	—	.373	.373	.245	.245
		.001	.094	.190	**.260**	.114	**.187**	.084	.139	**.202**	.137	**.197**
		.010	.553	.038	**.219**	.036	**.174**	.599	.056	**.201**	.052	**.183**
		.050	.936	.004	**.077**	.004	**.064**	.813	.011	**.080**	.011	**.072**
		.100	.916	.002	**.013**	.002	**.013**	.776	.006	**.020**	.005	**.019**
RCV1-v2	FULL SET	.000	—	.572	.572	.423	.423	—	.572	.572	.423	.423
		.001	.079	.557	**.567**	.368	**.412**	.070	.558	**.569**	.354	**.409**
		.010	.512	.348	**.480**	.224	**.337**	.525	.441	**.510**	.324	**.345**
		.050	.642	.138	**.331**	.074	**.218**	.761	.209	**.366**	.094	**.234**
		.100	.633	.064	**.122**	.047	**.066**	.697	.116	**.190**	.061	**.087**
	30 INFR	.000	—	.164	.164	.062	.062	—	.164	.164	.062	.062
		.001	.100	.102	**.097**	.044	**.049**	.076	.038	**.101**	.035	**.050**
		.010	.636	.025	**.074**	.024	**.041**	.667	.063	**.081**	.039	**.043**
		.050	.837	.006	**.038**	.005	**.032**	.925	.015	**.037**	.014	**.032**
		.100	.780	.005	**.014**	.003	**.011**	.854	.010	**.018**	.008	**.015**
OHSUMED	FULL SET	.000	—	.624	.624	.508	.508	—	.624	.624	.508	.508
		.001	.098	.340	**.437**	.235	**.328**	.090	.403	**.591**	.273	**.481**
		.010	.246	.129	**.153**	.072	**.086**	.584	.129	**.189**	.110	**.154**
		.050	.326	.010	.012	.007	.008	.666	.070	**.112**	.063	**.098**
		.100	.475	.002	.002	.001	.001	.653	.021	**.027**	.019	**.024**
	30 INFR	.000	—	.465	.465	.327	.327	—	.465	.465	.327	.327
		.001	.100	.118	**.176**	.080	**.129**	.095	.134	**.289**	.139	**.187**
		.010	.269	.061	**.081**	.037	**.056**	.731	.017	**.081**	.029	**.053**
		.050	.352	.003	.004	.002	.002	.852	.008	.012	.007	.010
		.100	.501	.001	.002	.001	.001	.823	.001	.004	.001	.004
OHSUMED-S	FULL SET	.000	—	.707	.707	.478	.478	—	.707	.707	.478	.478
		.001	.083	.539	**.706**	.422	.474	.048	.526	**.701**	.396	**.478**
		.010	.639	.459	**.686**	.279	**.375**	.535	.431	**.682**	.257	**.379**
		.050	.939	.215	**.513**	.141	**.218**	.809	.171	**.388**	.147	**.206**
		.100	.977	.177	**.430**	.119	.174	.798	.093	**.222**	.095	.128
	30 INFR	.000	—	.320	.320	.314	.314	—	.320	.320	.314	.314
		.001	.091	.093	**.203**	.284	.307	.062	.107	**.260**	.294	.301
		.010	.701	.022	**.042**	.045	**.120**	.570	.032	**.084**	.071	**.129**
		.050	.937	.008	.013	.008	.014	.712	.015	**.028**	.013	**.023**
		.100	.959	.002	.005	.002	.005	.760	.006	.008	.005	.007

increases in F_1. For instance, for the full set of REUTERS-21578 classes and targeted corruption, the value of $P_K = .082$ obtained for $\Pi = .001$ leads to a mere 3.5% increase in F_1^μ (from .821 to .850), but the much higher value of $P_K = .783$ obtained for $\Pi = .050$ leads to the much higher 302.3% increase in F_1^μ (from .209 to .632). All these results are indicative of the fact that TLC is an important and cost-effective way of improving accuracy for all the datasets of less-than-perfect annotation quality.

Finally, one question we might ask ourselves is: Can we provide guidelines on how much cleaning we should perform (i.e., what value of K we should use) in order to reach a desired improvement in effectiveness? Unfortunately, the results reported in Table IV seem to indicate this is not possible, even assuming one already knows in advance the corruption ratio Π that affects the dataset (and it is far from clear how Π could be known in practice). In fact, our tests show that there is a wide variability across datasets: for instance, Table IV shows that, for targeted corruption and $\Pi = .001$, cleaning 1% of the training set brings F_1^μ from .569 to .572 (a mere +0.5%) on RCV1-V2 but brings F_1^μ from .591 to .624 (+5.5%) on OHSUMED. This means that we cannot easily "learn" this function on a dataset and assume that these findings carry over to another dataset.

6. FURTHER EXPERIMENTS

6.1. Using a Low-Variance Learner

A potential concern regarding the "targeted corruption" experiments presented in Section 5 is that the same learner (MP-BOOST) is used both to corrupt the datasets *and* to learn classifiers from the corrupted training sets, which looks somehow self-referential. In other words, it might be argued that, if the training examples we corrupt are the ones that the classifier is least confident about (i.e., that are closer to the separating surface that the classifier itself identifies between the class and its complement), then a classifier generated from the corrupted training set *by means of the same learning technology used for corrupting the training set* will be less affected by the mislabelled training examples than a classifier generated by means of a different learning technology.

A related potential concern is that boosting is well known for being a low bias / high variance learning method [Geman et al. 1992], that is, is known for its sensitivity to the presence of noise in the training set [Dietterich 2000; Friedman et al. 2000; Maclin and Opitz 1997]. This might suggest that the levels of degradation in the classification accuracy of MP-BOOST that we have observed as a result of corruption (see Table III) are excessive with respect to what learning algorithms characterized by lower variance might experience.

In order to address both concerns, we have run a batch of experiments in which we employ, in place of MP-BOOST, an SVM that learns linear classifiers (a) in order to clean (via an SVM-based version of the CONF technique) our training sets corrupted via MP-BOOST-based targeted corruption, and (b) in order to test the degradation in accuracy resulting from the presence of mislabelled training examples. By doing so we address

(1) the first concern, by having two different learners at play in the two phases of targeted corruption and cleaning, respectively;
(2) the second concern, by using, in the classification phase, a low-variance learning method such as linear SVMs.

Implementing CONF via SVMs essentially means using the distance of the example from the separating surface (which is returned by the classifier together with the binary prediction for the example) as the confidence with which the example has been

Table V.

Mean average precision (MAP) of the CONF technique, implemented either via MP-BOOST or via SVMs, on the full set of classes (top 4 rows) and on the 30 most infrequent classes (bottom 4 rows) of REUTERS-21578, RCV1-v2 and OHSUMED. The MP-BOOST results are taken from Table II.

			Random corruption		Targeted corruption	
		Π	MP-BOOST	SVMs	MP-BOOST	SVMs
REUTERS-21578	FULL SET	.001	.596	.560	.510	.490
		.010	.653	.610	.608	.592
		.050	.968	.880	.677	.634
		.100	.973	.950	.665	.710
	30 INFR	.001	.748	.643	.648	.534
		.010	.674	.743	.581	.591
		.050	.982	.932	.647	.640
		.100	.981	.962	.673	.633
RCV1-v2	FULL SET	.001	.232	.198	.357	.221
		.010	.752	.650	.519	.490
		.050	.927	.842	.672	.560
		.100	.945	.923	.658	.643
	30 INFR	.001	.222	.182	.323	.280
		.010	.702	.642	.435	.451
		.050	.896	.730	.608	.591
		.100	.919	.832	.613	.620
OHSUMED	FULL SET	.001	.474	.464	.438	.398
		.010	.370	.356	.767	.630
		.050	.331	.321	.758	.690
		.100	.270	.238	.695	.590
	30 INFR	.001	.490	.487	.667	.671
		.010	.387	.380	.790	.750
		.050	.362	.343	.773	.767
		.100	.291	.251	.754	.772

classified; the higher the distance, the higher the confidence. As the SVM-based learner we have used the implementation from the freely available LibSvm library,[12] with a linear kernel and parameters at their default values.

Table V reports a comparison between the MAP results obtained by performing TLC with the MP-BOOST version of the CONF technique (that we had already reported in Table II), and those obtained by performing TLC with the SVM version of the same technique. As can be appreciated from Table V, the SVMs results are not qualitatively different from the MP-BOOST results, since they essentially confirm the insights obtained from the analysis of Table II, that is, (i) that TLC performs better for random corruption than for targeted corruption, and that (ii) MAP tends to increase with Π (increase rates for MP-BOOST and SVMs are also very close to each other). This answers the first concern raised at the beginning of this section, that is, that the results displayed in Table II might be essentially due to the same learner being used in corrupting the training set and in cleaning it.

Table VI reports instead a comparison between MP-BOOST and SVMs in terms of the classification accuracy that the classifiers they generate obtain after training on the corrupted datasets. The side-by-side results show that, in all evidence, there is

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

Table VI.

Comparison between the F_1 values obtained after corruption with MP-BOOST (indicated as MP-B) and SVMs. The MP-BOOST results are taken from Table III.

			Random corruption				Targeted corruption			
			F_1^μ		F_1^M		F_1^μ		F_1^M	
		Π	MP-B	SVMs	MP-B	SVMs	MP-B	SVMs	MP-B	SVMs
REUTERS-21578	FULL SET	.000	.852	.839	.606	.549	.852	.839	.606	.549
		.001	.822	.816	.356	.322	.821	.821	.448	.431
		.010	.583	.684	.227	.234	.632	.644	.254	.183
		.050	.138	.262	.074	.094	.209	.234	.094	.080
		.100	.064	.144	.047	.064	.116	.130	.061	.056
	30 INFR	.000	.373	.369	.245	.240	.373	.369	.245	.240
		.001	.190	.191	.114	.118	.139	.185	.137	.112
		.010	.038	.050	.036	.043	.056	.045	.052	.043
		.050	.004	.006	.004	.005	.011	.012	.011	.010
		.100	.002	.002	.002	.002	.006	.003	.005	.002
RCV1-V2	FULL SET	.000	.572	.565	.423	.421	.572	.561	.423	.421
		.001	.557	.550	.368	.360	.558	.553	.354	.365
		.010	.348	.342	.224	.218	.441	.390	.324	.290
		.050	.105	.098	.096	.085	.211	.150	.160	.124
		.100	.050	.051	.064	.045	.137	.120	.107	.098
	30 INFR	.000	.164	.160	.062	.060	.164	.160	.062	.060
		.001	.102	.110	.044	.050	.038	.109	.035	.051
		.010	.025	.030	.024	.023	.063	.060	.039	.041
		.050	.006	.008	.005	.006	.015	.014	.014	.013
		.100	.005	.007	.003	.006	.010	.008	.008	.008
OHSUMED	FULL SET	.000	.624	.611	.508	.493	.624	.611	.508	.493
		.001	.340	.348	.235	.239	.403	.475	.273	.238
		.010	.129	.115	.072	.069	.129	.415	.110	.165
		.050	.010	.010	.007	.006	.070	.189	.063	.066
		.100	.002	.002	.001	.002	.021	.173	.019	.054
	30 INFR	.000	.465	.432	.327	.291	.465	.432	.327	.291
		.001	.118	.127	.080	.085	.134	.181	.139	.140
		.010	.061	.064	.037	.038	.017	.120	.029	.074
		.050	.003	.004	.002	.002	.008	.106	.007	.062
		.100	.001	.001	.001	.002	.001	.060	.001	.030

qualitatively not much difference between the two learners in terms of how much their performance is degraded by corruption, with the two learners experiencing similar levels of degradation in effectiveness for similar corruption values. For instance, the degradation in F_1^μ experienced by MP-BOOST on the full set of REUTERS-21578 classes corrupted with TC at $\Pi = .010$ is from .852 to .632 (a -25.8% degradation), which is slightly higher than that experienced by SVMs (from .839 to .644, a -23.2% degradation); however, the reverse happens on RCV1-V2, with MP-BOOST experiencing smaller degradation (from .572 to .441, i.e., -22.9%) than SVMs (from .561 to .390, i.e., -30.5%). However, all in all the levels of degradation suffered by MP-BOOST and SVMs, for identical experimental conditions, is of the same order of magnitude. This answers also the second of the concerns raised at the beginning of this section, that is, that the degradation in classification effectiveness due from corruption, as reported in Table III, might essentially be due to the sensitivity to noise of boosting algorithms.

6.2. Using a Committee of Independent Members

As hinted in Section 5.4.1, one possible explanation for the fact that COMM dramatically underperforms CONF and NN, is that the classifier committee generated by MP-BOOST is made of members that are hardly independent of each other, which means that the patterns of agreement and disagreement among the members of the committee might be substantially different from the case of full independence. It might indeed be claimed that the intuition upon which COMM rests, that is, that ranking should be performed in terms of the agreement among a set of subjects, *inherently* requires independence among the members of the set.

As a result, we have performed a batch of experiments in which we have replaced the classifier committee generated by MP-BOOST with a classifier committee generated via the *bagging* technique [Breiman 1996]. Bagging consists of learning a set of S classifiers $\hat{\Phi}_s^j$, with $1 \leq s \leq S$, by training a learner on s different training sets Tr_s, each generated by sampling "with replacement" the original training set Tr until $|Tr_s| = |Tr|$. Given a test document d_i, its classification score is $\hat{\Phi}^j(d_i) = \frac{1}{S}\sum_{s=1}^{S}\hat{\Phi}_s^j(d_i)$. The fact that sampling with replacement is used is a guarantee of the mutual independence of the classifiers generated.

As the learning device for generating the classifier we have chosen the same weak learner as used in MP-BOOST (one that generates decision stumps), and as the size S of the committee we have chosen the same size as we have used in the MP-BOOST experiments; therefore, our COMM experiments with bagging are different from our COMM experiments with MP-BOOST only in terms of how the committee is generated.

The results of these experiments are reported in Table II, where COMM with bagging is labelled "BAG." Unfortunately, an analysis of these results does *not* confirm our conjecture that the bad results of COMM were the result of lack of independence among the members of the committee, since BAG does not systematically outperform (MP-BOOST -based) COMM, and is also frequently outperformed by it. The BAG experiments are thus a further confirmation that CONF should be the TLC method of choice.

7. RELATED WORK

Several works have used TLC in learning tasks other than text classification, especially within the realm of computational linguistics. For instance, TLC has been applied to POS tagging [Abney et al. 1999; Dickinson and Meurers 2003; Eskin 2000; Nakagawa and Matsumoto 2002; Yokoyama et al. 2005], verb modality identification [Murata et al. 2005], PP-attachment [Abney et al. 1999], and word segmentation for East Asian languages [Shinnou 2001]. Some of these works use task-independent TLC techniques while others do not. Among the former, Abney et al. [1999] and Shinnou [2001] use the DIS technique discussed at the end of Section 4, while Nakagawa and Matsumoto [2002] use a technique analogous to DIS that exploits the characteristics of SVMs. Eskin [2000] uses instead a generative probabilistic model based on the mixture of a majority distribution and an anomalous distribution, and for each training example computes the probabilities that the example has been generated by either of the two distributions, deeming the example a mislabelled one if the ratio between the two falls below a certain threshold. Other works use instead task-specific techniques; for instance, in a POS-tagging application Dickinson and Meurers [2003] top-rank multiple occurrences of the same word that have been labelled with different parts of speech in similar linguistic contexts, a technique that is obviously applicable only to POS-tagging or other sequence labelling tasks, and not to tasks such as TC. Yet other methods discussed in the literature, while not task-dependent, are learner-dependent. For instance, the approach championed in Zeng and Martinez [2001] is only applicable to neural-network learners, since the cleaning operation is incrementally performed

across the training epochs of the neural network. The methods that we propose in this article are both task-independent *and* learner-independent.

To the best of our knowledge the only two works that deal with TLC in the context of text classification are the ones by Fukumoto and Suzuki [2004] and Malik and Bhardwaj [2011].

Fukumoto and Suzuki's method consists in training an SVM, removing from the training set the support vectors that the SVM has identified, training a naive Bayesian classifier on the modified training set, and reclassifying the removed support vectors with this classifier, declaring mislabelled the support vectors whose original label does not match the newly assigned label. The intuition behind this technique is that if a training example has a wrong label for c_j, then it likely ends up being a support vector for the generated classifier. Unlike our techniques, this technique is strictly learner-dependent, since it only works with SVMs as learners. Additionally, the method is only limited to cleaning the support vectors; our method examines (and ranks) instead the entire training set; as a result, experimentally comparing the technique of Fukumoto and Suzuki [2004] with ours would be problematic.

Malik and Bhardwaj [2011] propose a TLC method based on (i) generating a classifier from a set of high-quality labelled documents, (ii) using it to (automatically) clean a set of low-quality labelled documents, and (iii) retraining the classifier by using the cleaned documents as additional training examples. Their work is different from ours in that we do not assume the existence of sets of labelled documents of different quality, an assumption that in many application contexts would likely be too restrictive; our method applies instead to any labelled document set, regardless of its quality.

Note instead that past work on what is often called "noisy text categorization" (see, e.g., [Agarwal et al. 2007; Vinciarelli 2005]) is quite unrelated to the present article, since it deals with the categorization of noisy texts (e.g., as obtained from OCR or automatic speech recognition processes), and not with the presence of noisy labels and their correction. TLC bears also some resemblance to "outlier detection", as used in many fields including data mining, fraud detection, or fault diagnosis. One difference is that the TLC techniques we present here are explicitly addressed to *labelled* data items, while outlier detection techniques are more generic with respect to this. Another difference lies in the very notion of outlier, which is different from the notion of a mislabelled training item, since an outlier may well indicate [John 1995] a "surprisingly veridical data item" (e.g., an instance that, although lying far away in the vector space from all other instances labelled with the class, is also labelled with the class, and correctly so).

Most of the works mentioned at the beginning of this section adopt an *a posteriori* evaluation methodology, that is, they perform no training set corruption, and evaluate their techniques by ranking the original training sets and then asking human annotators to look for mislabelled examples throughout the first K ranks, thus reporting precision-at-K results. We prefer the *a priori* evaluation methodology, since (i) it allows us to work with different corruption ratios, thus addressing the fact that different real-world applications may be characterized by different levels of quality in their data; (ii) it is exempt from evaluator bias, which the *a posteriori* methodology especially suffers from when (as is frequently the case) it is the authors themselves that engage in post-checking the results; (iii) it allows to compute MAP, while the *a posteriori* methodology only allows to compute precision for a specific, usually low value of K (i.e., the mislabelled items from the $(K + 1)$-st position onwards have no impact on the evaluation); (iv) it allows other researchers to replicate the results obtained in the experiments, while the *a posteriori* methodology does not. Additionally, the *a posteriori* methodology suffers from the problem that the human annotators that are engaged in the evaluation are not always qualified to decide whether a document is correctly or incorrectly labelled; labelling a document is sometimes a close call, and in these cases

the only subjects fully qualified to decide whether a given REUTERS-21578 document is correctly labelled or not should be the Reuters editors themselves, since they are the ones who precisely know the *intended* meaning of the labels.[13]

Concerning the *a priori* methodology, we should also note that all the works discussed in this section that employ it, be they about text classification or other learning tasks, use the *random* corruption methodology. As such, the idea of altering a dataset by *targeted* corruption is, to the best of our knowledge, an original contribution of the present article.[14]

Finally, let us note that the COMM technique is somehow reminiscent of the *query-by-committee* active-learning method (see, e.g., [Argamon-Engelson and Dagan 1999; Freund et al. 1992]), in which *unlabelled* examples (and not labelled ones, as in our case) are ranked for human annotation in increasing order of the agreement among a committee of classifiers that try to classify them. As a measure of (dis)agreement, Argamon-Engelson and Dagan [1999] use entropy. We have instead proposed using standard deviation, since entropy can only take into account the binary predictions of the various classifiers, and not the real-valued confidence in their prediction. Conversely, standard deviation can naturally account for predictions expressed as real numbers, and is thus a better fit in our case.

8. CONCLUSIONS AND FUTURE WORK

We have tested three techniques for training label cleaning on three popular multi-label text classification benchmarks, checking their ability at spotting and top-ranking texts that we have purposefully mislabelled, for experimental purposes only, in the training set. This experimental protocol allows to conveniently study *in vitro* the behaviour of these TLC techniques, and to precisely measure the relative merits of the various techniques by means of evaluation measures, such as MAP, standard in the field of ranked retrieval. Studying three TLC techniques with two different corruption models, at five different corruption levels, across three datasets (one of which consisting of more than 800,000 documents), and studying both the quality of the resulting rankings *and* the increase in effectiveness that carrying out TLC may bring about, our work probably qualifies as the first truly-large scale experimentation of TLC in either computational linguistics or IR.

Our experimental results show that one such technique, the confidence-based technique (CONF), achieves good MAP values across different settings deriving from the choice of different datasets, different class frequency, different corruption ratios, and different types of corruption, and generally outperforms the nearest-neighbours-based technique (NN). boosting (DIS) often performs well even if not always the top performer, and it might probably be used as the default choice. A third, committee-based technique (COM) has been shown instead to underperform the other two, regardless of the level of mutual independence among the members of the classifier committee.

A further result of this article is that a fourth technique (DIS), which had been proposed before and which was specific to boosting-based learners, is equivalent to the confidence-based technique (proposed here, which is instead applicable to all learners equipped with a notion of confidence in their own prediction).

[13]Analogously, the only person entitled to decide, for the purpose of giving feedback to a learning-based spam filter, whether an email message is ham or spam, should be the user of the filter herself, as witnessed from the well-known problem of "gray mail" [Yih et al. 2007].

[14]On a somehow similar note, in a single-label multiclass classification task, Brodley and Friedl [1996] corrupt the training data by only switching between classes that tend to be confused with each other. This is not possible in our case since our task is binary classification.

Our results also show that TLC is important, since they show that even a single mislabelled example in a thousand training examples can bring about deteriorations in effectiveness which are considerable in the general case, and no less than dramatic for the most infrequent classes and for macroaveraged F_1 in general.

Note also that the techniques we have presented here are applicable not only for cleaning *training* data, but also for cleaning generic sets of labelled text. That is, the very same techniques discussed here might be applied by a human annotator in order to clean a manually annotated text corpus (e.g., the entire RCV1-v2), regardless of the fact that the corpus is then going to be used for training a text classifier or not. For instance, this is useful for cleaning *test* sets, since incorrectly labelled test examples prevent the accurate measurement of effectiveness, but it is also useful for cleaning labelled datasets produced within organizations that entirely rely on manual classification.

This work still leaves some questions unanswered, which might thus be the subject of future research.

A first question is whether spotting and correcting a training example mislabelled as positive has the same value as spotting and correcting a training example mislabelled as negative. While in this article we have made the simplifying assumption that the two are equally important, future research could address the issue of attributing different importance values to the two cases, thus bringing about the need of evaluating TLC techniques in terms of cost-sensitive evaluation functions such as normalized discounted cumulative gain [Järvelin and Kekäläinen 2000], in place of the cost-insensitive MAP.

A second question arises if we want to compare TLC with active learning, since both are effectiveness-enhancing techniques that attempt to minimize the additional effort requested from a human annotator. Assuming that the annotation of a new unlabelled document requires an effort x times as large as inspecting an existing labelled document (for some $x \in [0, \infty)$), is it more cost-effective to annotate the n unlabelled documents top-ranked by an active learning technique, or to inspect the $x \cdot n$ documents top-ranked by a TLC technique? Presumably, the answer is a function of the corruption ratio of the training set, with high (resp., low) corruption ratios making TLC (resp., active learning) more cost-effective. Identifying the corruption ratio that acts as a threshold between the two cases would be extremely interesting.

ACKNOWLEDGMENTS

We thank Robert Schapire for discussions on the equivalence of the CONF and DIS techniques. We thank Giovanni Resta for investigating the formula for the expected AP of the random ranker on our suggestion; on this theme, thanks also to William Webber and Justin Zobel for useful discussions. Thanks also to the ICTIR 2009 and TOIS anonymous reviewers for critical work and suggestions that greatly helped to improve the quality of the article; TOIS Reviewer 1 is to be especially credited for some of the observations presented in Section 2.

REFERENCES

Abney, S., Schapire, R. E., and Singer, Y. 1999. Boosting applied to tagging and PP attachment. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC'99)*. 38–45.

Agarwal, S., Godbole, S., Punjani, D., and Roy, S. 2007. How much noise is too much: A study in automatic text classification. In *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM'07)*. 3–12.

Argamon-Engelson, S. and Dagan, I. 1999. Committee-based sample selection for probabilistic classifiers. *J. Artif. Intell. Res. 11*, 335–360.

Breiman, L. 1996. Bagging predictors. *Machine Learning 24*, 2, 123–140.

Brodley, C. E. and Friedl, M. A. 1996. Identifying and eliminating mislabeled training instances. In *Proceedings of the 13th Conference of the American Association for Artificial Intelligence (AAAI'96)*. 799–805.

Chapelle, O., Schölkopf, B., and Zien, A., Eds. 2006. *Semi-Supervised Learning*. MIT Press, Cambridge, MA.

Cohn, D., Atlas, L., and Ladner, R. 1994. Improving generalization with active learning. *Machine Learn. 15*, 2, 201–221.

Dickinson, M. and Meurers, W. D. 2003. Detecting errors in part-of-speech annotation. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL'03)*. 107–114.

Dietterich, T. G. 2000. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learn. 40*, 2, 139–157.

Eskin, E. 2000. Detecting errors within a corpus using anomaly detection. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics (NAACL'00)*. 148–153.

Esuli, A. and Sebastiani, F. 2009. Training data cleaning for text classification. In *Proceedings of the 2nd International Conference on the Theory of Information Retrieval (ICTIR'09)*. 29–41.

Esuli, A. and Sebastiani, F. 2010. Machines that learn how to code open-ended survey data. *Int. J. Market Res. 52*, 6, 775–800.

Esuli, A., Fagni, T., and Sebastiani, F. 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'06)*. 1–12.

Freund, Y., Seung, H. S., Shamir, E., and Tishby, N. 1992. Information, prediction, and query by committee. In *Advances in Neural Information Processing Systems*, Vol. 5, MIT Press, Cambridge, MA, 483–490.

Friedman, J., Hastie, T., and Tibshirani, R. J. 2000. Additive logistic regression: A statistical view of boosting. *Ann. Statist. 2*, 337–374.

Fukumoto, F. and Suzuki, Y. 2004. Correcting category errors in text classification. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING'04)*. 868–874.

Galavotti, L., Sebastiani, F., and Simi, M. 2000. Experiments on the use of feature selection and negative evidence in automated text categorization. In *Proceedings of the 4th European Conference on Research and Advanced Technology for Digital Libraries (ECDL'00)*. 59–68.

Geman, S., Bienenstock, E., and Doursat, R. 1992. Neural networks and the bias/variance dilemma. *Neural Comput. 4*, 1, 1–58.

Grady, C. and Lease, M. 2010. Crowdsourcing document relevance assessment with Mechanical fiTurk. In *Proceedings of the NAACL HLT Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*. 172–179.

Hersh, W., Buckley, C., Leone, T., and Hickman, D. 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'94)*. 192–201.

Järvelin, K. and Kekäläinen, J. 2000. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd ACM International Conference on Research and Development in Information Retrieval (SIGIR'00)*. 41–48.

John, G. H. 1995. Robust decision trees: Removing outliers from databases. In *Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining (KDD'95)*. 174–179.

Lewis, D. D. 2004. Reuters-21578 text categorization test collection Distribution 1.0 README file (v 1.3). http://www.daviddlewis.com/resources/testcollections/reuters21578/readme.txt.

Lewis, D. D., Schapire, R. E., Callan, J. P., and Papka, R. 1996. Training algorithms for linear text classifiers. In *Proceedings of the 19th ACM International Conference on Research and Development in Information Retrieval (SIGIR'96)*. 298–306.

Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. 2004. RCV1: A new benchmark collection for text categorization research. *J. Machine Learn. Res. 5*, 361–397.

Maclin, R. and Opitz, D. W. 1997. An empirical evaluation of bagging and boosting. In *Proceedings of the 14th Conference of the American Association for Artificial Intelligence (AAAI'97)*. 546–551.

Malik, H. H. and Bhardwaj, V. S. 2011. Automatic training data cleaning for text classification. In *Proceedings of the ICDM Workshop on Domain-Driven Data Mining*. 442–449.

Murata, M., Utiyama, M., Uchimoto, K., Isahara, H., and Ma, Q. 2005. Correction of errors in a verb modality corpus for machine translation with a machine-learning method. *ACM Trans. Asian Lang. Inform. Process. 4*, 1, 18–37.

Nakagawa, T. and Matsumoto, Y. 2002. Detecting errors in corpora using support vector machines. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*. 1–7.

Resta, G. 2012. On the expected average precision of the random ranker. Tech. rep. IIT TR-04/2012, Istituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche, Pisa, IT. http://www.iit.cnr.it/sites/default/files/TR-04-2012.pdf.

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

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

Schapire, R. E. and Freund, Y. 2012. Boosting: *Foundations and Algorithms*. MIT Press, Cambridge, MA.

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

Sindhwani, V. and Keerthi, S. S. 2006. Large scale semi-supervised linear SVMs. In *Proceedings of the 29th ACM International Conference on Research and Development in Information Retrieval (SIGIR'06)*. 477–484.

Snow, R., O'Connor, B., Jurafsky, D., and Ng, A. Y. 2008. Cheap and fast - but is it good? Evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'08)*. 254–263.

Vinciarelli, A. 2005. Noisy text categorization. *IEEE Trans. Pattern Anal. Mach. Intell. 27*, 12, 1882–1895.

Yang, Y. 1994. Expert network: Effective and efficient learning from human decisions in text categorisation and retrieval. In *Proceedings of the 17th ACM International Conference on Research and Development in Information Retrieval (SIGIR'94)*. 13–22.

Yang, Y. 1999. An evaluation of statistical approaches to text categorization. *Inf. Retriev.* 1, 1/2, 69–90.

Yih, W.-T., McCann, R., and Kolcz, A. 2007. Improving spam filtering by detecting gray mail. In *Proceedings of the 4th Conference on Email and Anti-Spam (CEAS'07)*.

Yokoyama, M., Matsui, T., and Ohwada, H. 2005. Detecting and revising misclassifications using ILP. In *Proceedings of the 8th International Conference on Discovery Science (DS'05)*. 75–80.

Yu, K., Zhu, S., Xu, W., and Gong, Y. 2008. Non-greedy active learning for text categorization using convex transductive experimental design. In *Proceedings of the 31st ACM International Conference on Research and Development in Information Retrieval (SIGIR'08)*. 635–642.

Zeng, X. and Martinez, T. R. 2001. An algorithm for correcting mislabeled data. *Intell. Data Anal. 5*, 6, 491–502.

Zhu, X. and Goldberg, A. B. 2009. *Introduction to Semi-Supervised Learning*. Morgan and Claypool, San Rafael, CA.

Received June 2012; revised January 2013, April 2013; accepted June 2013