UNIVERSITÀ DI PISA
Scuola di Dottorato in Ingegneria "Leonardo da Vinci"

Corso di Dottorato di Ricerca in
INGEGNERIA DELL'INFORMAZIONE

Tesi di Dottorato di Ricerca

Semi-Automated Text Classification

Giacomo Berardi

Anno 2014

UNIVERSITÀ DI PISA

Scuola di Dottorato in Ingegneria "Leonardo da Vinci"

Corso di Dottorato di Ricerca in
Ingegneria dell'Informazione

Tesi di Dottorato di Ricerca

Semi-Automated Text Classification

Autore:

Giacomo Berardi

Relatori:

Prof. Luca Simoncini

Dott. Fabrizio Sebastiani

Dott. Andrea Esuli

Anno 2014

SSD ING-INF/05

Sommario

Al giorno d'oggi esiste una forte domanda di sistemi informatici per l'analisi automatica dei dati testuali. Le grandi industrie e organizzazioni hanno bisogno di elaborare enormi quantità di dati testuali, un'attività che non può essere eseguita con il solo lavoro umano. La *Text Classification* (TC) è l'attività di etichettare automaticamente i documenti testuali di un insieme $\mathcal{D}$ con le categorie tematiche di un insieme predefinito $\mathcal{C}$. I sistemi di text classification lavorano con un'alta efficienza, ma non possono garantire un'accuratezza impeccabile dell'etichettatura.

La *Semi-Automated Text Classification* (SATC) consiste nell'attività di ordinare un insieme di documenti testuali etichettati automaticamente $\mathcal{D}$ in modo che, se un annotatore umano validasse (i.e., ispezionasse e correggesse dove appropriato) una porzione dei primi documenti dell'ordinamento con l'obiettivo di incrementare l'accuratezza dell'etichettatura di $\mathcal{D}$, l'incremento atteso venga massimizzato. Una strategia ovvia è quella di ordinare $\mathcal{D}$ in modo che i documenti che il classificatore ha etichettato con confidenza più bassa siano i primi dell'ordinamento. In questa tesi dimostriamo che questa strategia è subottimale. Sviluppiamo nuovi metodi di ordinamento basati sulla teoria dell'utilità e sul concetto di *guadagno della validazione*, definito come il miglioramento dell'efficacia di classificazione che deriverebbe validando un dato documento etichettato automaticamente. Proponiamo inoltre nuove misure di efficacia per i metodi di ordinamento orientati alla SATC, basati sulla riduzione attesa dell'errore di classificazione, riduzione ottenuta dalla validazione di parte della lista do documenti generata da un dato metodo di ordinamento.

Riportiamo i risultati degli esperimenti che dimostrano che, in confronto al metodo di base di cui sopra, e secondo le misure proposte, i nostri metodi di ordinamento basati sulla teoria dell'utilità sono in grado di ottenere una riduzione attesa dell'errore di classificazione sostanzialmente maggiore. Esploriamo quindi l'attività della SATC e il potenziale dei nostri metodi, in molteplici contesti della text classification. Questa tesi è, al meglio delle nostre conoscenze, la prima ad affrontare l'attività della semi-automated text classification.

Abstract

Nowadays there is a high demand of information systems for the automatic analysis of textual data. Large industries and organizations need to process huge amounts of textual data, an activity that cannot be performed with human work only. *Text Classification* (TC) is the task of automatically labelling textual documents from a set $\mathcal{D}$ with thematic categories from a predefined set $\mathcal{C}$. Text classification systems work with high efficiency, but they cannot guarantee impeccable labelling accuracy.

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

We report the results of experiments showing that, with respect to the baseline method above, and according to the proposed measures, our utility-theoretic ranking methods can achieve substantially higher expected reductions in classification error. We therefore explore the task of SATC and the potential of our methods, in multiple text classification contexts. This dissertation is, to the best of our knowledge, the first to address the task of semi-automated text classification.

Contents

1 Introduction 1
- 1.1 Text classification 2
- 1.2 Semi-automated text classification 3
- 1.3 Our contribution 4
- 1.4 Structure of the thesis 5

2 Text Classification 7
- 2.1 Supervised learning 7
  - 2.1.1 Text representation 9
  - 2.1.2 Support vector machines 10
  - 2.1.3 Boosted decision trees 12
- 2.2 Evaluating text classification 15
  - 2.2.1 Evaluation measures 15
  - 2.2.2 Datasets for text classification benchmarks 17
- 2.3 Conclusions 19

3 A Ranking Method for Semi-Automated Text Classification 21
- 3.1 A worked-out example 21
- 3.2 Ranking by probability of misclassificaton 24
- 3.3 Utility theory 26
  - 3.3.1 Ranking by utility 27
  - 3.3.2 Validation gains 28
  - 3.3.3 Smoothing contingency cell estimates 29
  - 3.3.4 Ranking by total utility 30
- 3.4 Expected normalized error reduction 31
  - 3.4.1 Error reduction at rank 31
  - 3.4.2 Normalized error reduction at rank 33
  - 3.4.3 ... and its expected value 33
- 3.5 Experiments 35
  - 3.5.1 Experimental protocol 35

3.5.2 Probability calibration . . . 35
3.5.3 Learning algorithms . . . 36
3.5.4 Lower bounds and upper bounds . . . 36
3.5.5 Results and discussion . . . 37
Mid-sized test sets . . . 37
Small test sets . . . 42
Tiny test sets . . . 42
Large test sets . . . 44
Discussion . . . 46
3.6 Further readings . . . 46
3.6.1 Probability estimates of classification . . . 48
3.6.2 Evaluating rankings by modelling user behavior . . . 48
3.6.3 Human interaction in learning systems . . . 49
3.7 Conclusions . . . 50

4 Additional Ranking Methods for Semi-Automated Text Classification . . . 53
4.1 An improved, "dynamic" ranking function . . . 53
4.1.1 Experiments . . . 56
4.2 A "micro-oriented" ranking function . . . 62
4.2.1 Experiments . . . 65
4.3 Conclusions . . . 69

5 Semi-Automated Text Classification and Active Learning . . . 71
5.1 Active learning . . . 72
5.1.1 Active learning strategies . . . 72
5.1.2 Multi-label active learning . . . 74
5.2 A comparison of active learning with semi-automated text classification 76
5.2.1 Semi-automated text classification methods for active learning 76
Experimental protocol . . . 77
Results and discussion . . . 80
5.2.2 Active learning methods for semi-automated text classification 83
Results and discussion . . . 83
5.3 Conclusions . . . 88

6 Evaluating Semi-Automated Text Classification Applications . . . 89
6.1 Estimating accuracy . . . 89
6.1.1 Comparing learning methods globally . . . 91
6.1.2 Results and discussion . . . 91
6.2 Other measures of classification accuracy . . . 95
6.2.1 Results and discussion . . . 96
6.3 A new measure of error reduction . . . 99
6.3.1 Results and discussion . . . 101

6.4 Experiments with automated verbatim coding 104
6.5 Conclusions 109

7 Conclusions 111
7.1 Future directions 112

References 115

List of Figures

2.1 An example of a hyperplane which brings the largest separation margin of documents for class c_j. The vector space has two dimensions for terms t_1 and t_2. 11
2.2 The MP-Boost algorithm 13
2.3 An example of a contingency table of a single-label classification of $TP + TN + FP + FN = 20$ documents. 15
2.4 Cumulative distributions of confidence estimates for MP-Boost (Figure a) and SVMs (Figure b) on the Reuters21578 test set. The total number of classifications is $|Te| \cdot |\mathcal{C}|$, that for Reuters21578 is equivalent to $3299 \cdot 115 = 379385$ 19

3.1 A worked-out example, representing a contingency table (upper left part of the figure) deriving from the automatically labelled examples (lower part) and from which accuracy is computed (upper right part). . 22
3.2 The generalized logistic function. 25
3.3 Error reduction, measured as ER^M_ρ, as a function of validation depth. The dataset is Reuters-21578, the learner is MP-Boost. The Random curve indicates the results of our estimation of the expected ER of the random ranker via a Monte Carlo method with 100 random trials. Higher curves are better. 32
3.4 Same as Figure 3.3, but with the SVM learner in place of MP-Boost. 40
3.5 Results obtained by (i) splitting the Reuters-21578 test set into 10 random, equally-sized parts, (ii) running the analogous experiments of Figure 3.3 independently on each part, and (iii) averaging the results across the 10 parts. 43
3.6 Same as Figure 3.5 but with Reuters-21578/100 in place of Reuters-21578/10. 45

4.1 Error reduction, measured as ER^M_ρ, as a function of validation depth. The dataset is Reuters-21578, the learner is MP-Boost. The Random curve indicates the results of our estimation of the expected ER of the random ranker via a Monte Carlo method with 100 random trials. Higher curves are better. 57
4.2 Same as Figure 4.1, but with the SVM learner in place of MP-Boost. 58
4.3 Results obtained by (i) splitting the Reuters-21578 test set into 10 random, equally-sized parts, (ii) running the analogous experiments of Figure 4.1 independently on each part, and (iii) averaging the results across the 10 parts. 59
4.4 Same as Figure 4.3 but with Reuters-21578/100 in place of Reuters-21578/10. 59
4.5 Error reduction, measured as ER^μ_ρ, as a function of validation depth. The dataset is Reuters-21578, the learner is MP-Boost. 68

5.1 Results for macro-averaged F_1 of active learning runs, the evaluation is made on Te, learners are MP-Boost (5.1a and 5.1c) and SVMs (5.1b and 5.1d), datasets are Reuters-21578 (5.1a and 5.1b) and OHSUMED-S (5.1c and 5.1d). 81
5.2 Results for micro-averaged F_1 of active learning runs, the evaluation is made on Te, learners are MP-Boost (5.2a and 5.2c) and SVMs (5.2b and 5.2d), datasets are Reuters-21578 (5.2a and 5.2b) and OHSUMED-S (5.2c and 5.2d). 82
5.3 Results for macro-averaged error reduction of active learning methods applied on SATC, for learners MP-Boost (5.3a and 5.3c) and SVMs (5.3b and 5.3d), on datasets Reuters-21578 (5.3a and 5.3b) and OHSUMED-S (5.3c and 5.3d). 84
5.4 Results for micro-averaged error reduction of active learning methods applied on SATC, for learners MP-Boost (5.4a and 5.4c) and SVMs (5.4b and 5.4d), on datasets Reuters-21578 (5.4a and 5.4b) and OHSUMED-S (5.4c and 5.4d). 85

6.1 Comparison of the actual and estimated F^M_1 and F^μ_1 on the test sets. The curves describe the increment in F_1 during manual validation of the test sets. 93
6.2 Results obtained on Reuters-21578 with MP-Boost for the measures $ER^M_{0.5}$ (Figure 6.2a) and ER^M_2 (Figure 6.2b). 97

List of Tables

2.1 Characteristics of the test collections used. From left to right we report the number of training documents (2), the number of test documents (3), the number of classes (4), and the average number of classes per test document (5). Columns 6-9 report the initial accuracy (both F_1^M and F_1^μ) generated by the MP-BOOST and SVMs classifiers. 18

3.1 Results of various ranking methods, applied to MP-BOOST and several test collections, in terms of $ENER_\rho^M(\xi)$, for $\xi \in \{0.05, 0.10, 0.20\}$. Improvements listed for the various methods are relative to the baseline. REUTERS-21578/10 and REUTERS-21578/100 are respectively named as R-21578/10 and R-21578/100 38

3.2 Results of various ranking methods, applied to SVMs and several test collections, in terms of $ENER_\rho^M(\xi)$, for $\xi \in \{0.05, 0.10, 0.20\}$. Improvements listed for the various methods are relative to the baseline. REUTERS-21578/10 and REUTERS-21578/100 are respectively named as R-21578/10 and R-21578/100 39

3.3 Characteristics of the test collections used. From left to right we report the number of test sets (Column 2) and, for each test set, the number of training documents (3), the number of test documents (4), the number of classes (5), and the average number of classes per test document (6). Columns 7-10 report the initial error (both E_1^M and E_1^μ) generated by the MP-BOOST and SVMs classifiers. 41

4.1 Results of various ranking methods, applied to MP-BOOST and several test collections, in terms of $ENER_\rho^M(\xi)$, for $\xi \in \{0.05, 0.10, 0.20\}$. . . . 60

4.2 Results of various ranking methods, applied to SVMs and several test collections, in terms of $ENER_\rho^M(\xi)$, for $\xi \in \{0.05, 0.10, 0.20\}$. 61

4.3 Comparison between the actual computation times (in seconds) of the U-Theoretic(s) and U-Theoretic(d) methods on our five datasets. 62

4.4 As Table 4.1, but with $ENER_\rho^\mu(\xi)$ in place of $ENER_\rho^M(\xi)$. 66

4.5 As Table 4.2, but with $ENER^{\mu}_{\rho}(\xi)$ in place of $ENER^{M}_{\rho}(\xi)$. 67

5.1 Results of ranking methods based on active learning strategies, applied to MP-Boost and SVMs, on Reuters-21578 and OHSUMED-S, in terms of $ENER^{M}_{\rho}(\xi)$, for $\xi \in \{0.05, 0.10, 0.20\}$. Improvements listed for the various methods are relative to the baseline. 86
5.2 Results of ranking methods based on active learning strategies, applied to MP-Boost and SVMs, on Reuters-21578 and OHSUMED-S, in terms of $ENER^{\mu}_{\rho}(\xi)$, for $\xi \in \{0.05, 0.10, 0.20\}$. Improvements listed for the various methods are relative to the baseline. 87

6.1 Comparison of learners effectiveness and estimated F^{M}_{1} and F^{μ}_{1}. Datasets are indicated as R for Reuters-21578, O for OHSUMED and O-S for OHSUMED-S. In the first column we indicate the evaluation of the classifiers on Te, in the second column we indicate the evaluations estimated on Tr via 10-fold cross-validation, and in the third column we indicate the estimated evaluation after the total manual validation of Te. 94
6.2 Errors generated by the MP-Boost and SVMs classifiers, evaluating the classification accuracy with $F_{0.5}$ and F_2. 96
6.3 Results of ranking methods applied to MP-Boost and SVMs, on Reuters-21578, OHSUMED and OHSUMED-S, in terms of $ENER^{\mu}_{0.5}(\xi)$, for $\xi \in \{0.05, 0.10, 0.20\}$. Improvements listed for the various methods are relative to the baseline. 98
6.4 Results of ranking methods applied to MP-Boost and SVMs, on Reuters-21578, OHSUMED and OHSUMED-S, in terms of $ENER^{\mu}_{2}(\xi)$, for $\xi \in \{0.05, 0.10, 0.20\}$. Improvements listed for the various methods are relative to the baseline. 100
6.5 Results of ranking methods applied to MP-Boost and SVMs, on Reuters-21578, Reuters-21578/10 and Reuters-21578/100, in terms of $EDNER^{M}_{\rho}(\xi)$, for $\xi \in \{0.05, 0.10, 0.20\}$. Improvements listed for the various methods are relative to the baseline. 103
6.6 Results of ranking methods applied to SVMs, on Reuters-21578, Reuters-21578/10 and Reuters-21578/100, in terms of $EDNER^{\mu}_{\rho}(\xi)$, for $\xi \in \{0.05, 0.10, 0.20\}$. Improvements listed for the various methods are relative to the baseline. 104

6.7 Characteristics of the ten market research datasets (LL-A to LL-L), four customer satisfaction datasets (Egg-A1 to Egg-B2), and one social science dataset (ANES L/D), that we have used here for experimentation. The columns represent the name of the dataset, the number of training verbatims ($|Tr|$) and the number of test verbatims ($|Te|$), the number of codes in the codeframe ($|\mathcal{C}|$), the average number of positive training verbatims per code (AVC), the average training verbatim length (AVL), and the initial error (both E_1^M and E_1^μ) generated by the MP-Boost and SVMs classifiers. 105
6.8 Results of SATC ranking methods on survey datasets, in terms of $ENER_\rho^M(\xi)$, for $\xi \in \{0.05, 0.10, 0.20\}$. Improvements listed for the various methods are relative to the baseline. 107
6.9 Results of SATC ranking methods on survey datasets, in terms of $ENER_\rho^\mu(\xi)$, for $\xi \in \{0.05, 0.10, 0.20\}$. Improvements listed for the various methods are relative to the baseline. 108

1

Introduction

Textual data is present in our everyday activities, e.g., when we read the news on the Web, when we fill out paperwork, etc. Analyzing textual data is an important activity in many scenarios of our society, but it sometimes involves difficult challenges that we can barely manage with human effort only. One of the main problems in text analysis is that texts are now generated at high rates, thus it is often the case that the size of the data forces us to entrust this analysis to computing systems.

There is a high demand of automatic systems for extracting information and understanding the content of textual data. The need of structuring knowledge from texts can be tackled computationally, so in the last years many research studies in information technology have pushed forward the state of the art of *text mining* technologies.

Several tasks make up the text mining field, and for each task different approaches have been proposed to solve them. Applications of text mining usually have to follow specific requirements, which are imposed by the need of extracting or accessing particular types of information. The challenge for a researcher is to find and apply the right methods, in order to optimize the effectiveness and the efficiency of the applications.

Modern search engines are a popular application for automatic access to textual data. Search engines are a solution to the activity of *information retrieval* (IR) [54, 78]. A retrieval application has to answer the information need of a user, by exploring and identifying the relevant data. Information retrieval embraces a wide area of the text mining research and the impact of IR research is huge in our society. Sometimes a text mining application has to meet more specific requirements, which do not respond to a generic need as in the case of search engines. Some applications answer the demand of a single customer, or they are applied on particular textual data, publicly accessible or private. This scenario is more common for today's large organizations and industries, which are faced with the need of processing large amounts of textual data in order to understand the satisfaction of their clients or to manage their own knowledge bases.

1.1 Text classification

Suppose an organization needs to classify a set $\mathcal{D}$ of textual documents under a classification scheme $\mathcal{C}$, that consists of a set of labels $c_j \in \mathcal{C}$, each of which can be assigned or not to any of the documents $d_i \in \mathcal{D}$. Suppose that $\mathcal{D}$ is too large to be classified manually, so that resorting to some form of automated *text classification* (TC) [2, 73] is the only viable option. TC is the activity of automatically labelling natural language texts with thematic *classes* (or *categories* or *labels*) from this predefined set $\mathcal{C}$.

In this task the knowledge we can access is in the form of textual documents, and no other information is available. The categories are just symbolic labels, and they identify a set of documents according to some predefined criteria. One could group documents by topic, by opinion, or according to other aspects, depending of the TC application; some examples of applications are:

- the categorization of news articles, where documents are classified according to the topic they deal with (e.g., politics, sport, etc.), or are filtered according to the profile of the user [42];
- *spam filtering* [15], in which the TC application is integrated in the e-mail client, so that the e-mails automatically labelled as spam are filtered out by the client;
- *sentiment classification* [52], in which TC is used together with *computational linguistic* technologies, in order to label documents (e.g., product reviews or political debates) according to the opinions they contain.

Typical applications of TC are in scenarios where a large quantity of documents has to be analyzed, and the human cost needed to manually perform the annotation is, most of the times, prohibitive. TC applications may be not as accurate as human labellers, but they are cheaper and surely more efficient.

The TC task can be further characterized depending on the desired application. For example, one might accept that any number of categories can be assigned to a document; this is the *multi-label* case of TC. Alternatively, or one could restrict this number to one, so that exactly one category from the set must be assigned to a document (the *single-label* case). In general these characteristics are parameters of a TC framework. The modern and effective approach to TC is through *supervised learning*. A TC framework built on supervised learning algorithms works in two phases: (i) it receives in input the set $\mathcal{C}$ and a set of documents, labelled with the categories in $\mathcal{C}$ (the so-called *training data*); it learns from the data, understanding the characteristics which link the documents to the categories, and it models the learned information into a *classifier*; (ii) the classifier performs the automatic classification of the documents in $\mathcal{D}$, according to the categories in $\mathcal{C}$.

The effectiveness of a classifier is dependent of the information on which it is built. The quality of the training data, that is usually generated manually, and the choice of the algorithm and its parameters determine the accuracy of the classifier.

But what if, as a consequence, the customer is not satisfied by the accuracy of the automatic classification? What if the customer is willing to do, or pay for, a part of the classification manually? We answer these questions by introducing a novel scenario for text classification.

1.2 Semi-automated text classification

Suppose that an organization which needs to classify documents has strict accuracy standards, so that the level of effectiveness obtainable via state-of-the-art TC technology is not sufficient. In this case, the most plausible strategy to follow is to classify the set $\mathcal{D}$ of documents by means of an automatic classifier (which we assume here to be generated by a supervised learning algorithm), and then to have a human editor validate (i.e., inspect and correct where appropriate) the results of the automatic classification. The human annotator will obviously validate only a subset $\mathcal{D}' \subset \mathcal{D}$ (since it would not otherwise make sense to have an initial automated classification phase), e.g., until she is confident that the overall level of accuracy of $\mathcal{D}$ is sufficient, or until she has time. We call this scenario *semi-automated text classification* (SATC).

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

In this dissertation we will devise effective ranking strategies for this task, and we will explore the feasibility of SATC applications and their integration into TC systems.

Anyone already familiar with TC technologies might ask: why not just use *active learning*? In active learning (AL) [74] the human effort is spent for annotating new documents (typically from the set of automatically labelled documents), which are added to the training data in order to learn from a richer set of documents. The goal of SATC is not improving classifier quality, but directly improving the accuracy of the set of automatically classified documents. The human validation typical of SATC comes *after* the automatic classification phase, guarantees a positive gain in effectiveness, which is a strict requirement.In this dissertation we will investigate the relationship between SATC and AL.

Another question can come to mind: is there a real need for SATC applications? The answer is yes, there are multiple scenarios in which automatic classification and human intervention both play a role, and we will discuss them in this dissertation. An example task in which human validation is a critical phase is *e-discovery* [29, 63]. E-discovery is a process, in a legal case, where one party has to make available to the other the digital material in its possession that is pertinent with the case. This material is often in the form of textual data, and the size of these data makes this task difficult to process by means of manual work only. The process of reviewing

the documents is often *technology-assisted*, and split in several steps: in a first phase a subset of documents is retrieved and annotated according to its relevance to the case, then a classifier is built so as to find the maximum possible number of relevant documents. The final goal is to produce an accurate classification of relevant and non-relevant documents, with a continuous interaction of human reviewers throughout the process. Documents are usually ranked, for presentation to the reviewers, by means of their relevance (computed from the classification output scores), so as to optimize the reviewers' work. In this scenario SATC seems to be the right solution for optimizing the manual, expensive work of human labelling. In fact, ranking documents by relevance is suboptimal (as we will show in this dissertation), if the objective is validating the classification process. We can thus realize that SATC is applicable in any process that involves the automatic analysis of data with the support of human validators.

1.3 Our contribution

An obvious strategy for a SATC method is to rank the documents in ascending order of the confidence scores returned by the classifier. In fact, classifiers based on supervised learning algorithms usually output a real value along with the classification decision, which is proportional to their confidence in the decision. By employing this strategy, the top-ranked documents are the ones that the classifier has labelled with the lowest confidence. We call this strategy "obvious" because of the evident similarities between SATC and active learning, where this strategy is an often used baseline. However, to the best of our knowledge, the application of this ranking method (or of any other ranking method, for that matter) to SATC has never actually been discussed in the literature. The rationale for this method is that an increase in effectiveness can derive only by validating *misclassified* documents, and that a good ranking method is simply the one that top-ranks the documents with the highest probability of misclassification, which (in the absence of other information) we may take to be the documents which the classifier has categorized with the lowest confidence.

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

The contributions of this dissertation are the following:

- we develop new utility-theoretic ranking methods for SATC based on the notion of *validation gain*, i.e., the improvement in effectiveness that would derive by correcting a given type of mistake (i.e., false positive or false negative);
- we introduce a further ranking method, which is based on the notion of "dynamic" adjustment of validation gains;
- we formulate the above ranking methods for two different methods of averaging classification accuracy across different classes;
- we propose a new evaluation measure for SATC, and use it to evaluate our experiments on standard datasets;
- we perform a number of analyses and extensions of the methods above and the evaluation measures above, and we apply our SATC methods in different application contexts.

The results of all the experiments in this dissertation show that, with respect to the confidence-based baseline method above, our ranking methods are substantially more effective.

This dissertation presents, to the best of our knowledge, the first systematic work on the task of semi-automated text classification. This work has resulted in the following scientific publications:

I) A utility-theoretic ranking method for semi-automated text classification. Giacomo Berardi, Andrea Esuli, and Fabrizio Sebastiani. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2012)*, Portland, US. This is The first technical paper we have published on the topic.

II) Optimising Human Inspection Work in Automated Verbatim Coding. Giacomo Berardi, Andrea Esuli, and Fabrizio Sebastiani. *International Journal of Market Research*. To appear. This is a paper which introduces SATC in easy-to-understand terms to a non-specialized audience (market researchers) of potential users of this technology.

III) Utility-Theoretic Ranking for Semi-Automated Text Classification. Giacomo Berardi, Andrea Esuli, and Fabrizio Sebastiani. *ERCIM News*, 2013(92). This is an extended abstract of I), published in an international computer science bulletin.

IV) Utility-Theoretic Ranking for Semi-Automated Text Classification. Giacomo Berardi, Andrea Esuli, and Fabrizio Sebastiani. Submitted to *ACM Transactions on Information Systems*. This is a substantially revised and extended version of I), submitted to the top information retrieval journal.

1.4 Structure of the thesis

The dissertation is organized as follows.

In Chapter 2 we describe how automated text classification is actually achieved. We introduce preliminary definitions, notations, and the approach to TC by means

of supervised learning. We thus describe algorithms, evaluation measures, and data collections that we use as the core in the development of SATC methods.

In Chapter 3 we introduce our approach to SATC. We describe our basic utility-theoretic strategy for ranking the automatically labelled documents. Furthermore, we propose a novel effectiveness measure for this task based on a probabilistic user model. We report the results of our experiments in which we test the effectiveness of ranking strategies by simulating the work of a human annotator who validates variable-sized portions of the labelled test set. Finally, we propose some further readings about the discussed subjects.

In Chapter 4 we address a potential problem deriving from the "static" nature of our strategy, by describing a "dynamic" (albeit computationally more expensive) version of the same strategy, and draw an experimental comparison between the two. Then we acknowledge the existence of two different ways ("micro" and "macro") of averaging effectiveness results across classes, and show that the methods we have developed so far are optimized for macro-averaging; we thus develop and test methods optimized for micro-averaged effectiveness.

In Chapter 5 we investigate the task of active learning and its relationships with SATC. We look at the common aspects and the differences of the two tasks. We compare methods specific to AL or SATC in two steps: first we test our SATC ranking methods in an AL scenario, then we apply AL methods to a SATC setting.

In Chapter 6 we further explore the potential of our SATC methods, facing a number of subtasks: first we try to determine if we can dynamically estimate the achieved level of accuracy of the classification, which is a central aspect of a SATC application. We then study other classification accuracy measures, and a new variant of our evaluation function for SATC. We finally we apply SATC ranking methods in the context of automatic classification of market research surveys.

In Chapter 7 we conclude by charting avenues for future research.

2

Text Classification

In this chapter we describe the basic notions and the methods for performing automatic text classification [2, 73]. Along the chapter we introduce the notation used in this dissertation.

Given a set of textual documents $\mathcal{D}$ and a predefined set of classes (or categories) $\mathcal{C} = \{c_1, \ldots, c_m\}$, *multi-class multi-label* TC is usually defined as the task of estimating an unknown *target function* $\Phi : \mathcal{D} \times \mathcal{C} \rightarrow \{-1, +1\}$, that describes how documents ought to be classified, by means of a function $\hat{\Phi} : \mathcal{D} \times \mathcal{C} \rightarrow \{-1, +1\}$ called the *classifier*[1]; $+1$ and -1 represent membership and non-membership of the document in the class. Here, "multi-class" means that there are $m \geq 2$ classes, while "multi-label" refers to the fact that each document may belong to zero, one, or several classes at the same time. Multi-class multi-label TC is usually accomplished by generating m independent binary classifiers $\hat{\Phi}_j$, one for each $c_j \in \mathcal{C}$, each entrusted with deciding whether a document belongs or not to a class c_j.

In this dissertation we will actually restrict our attention to classifiers $\hat{\Phi}_j$ that, aside from taking a binary decision $D_{ij} \in \{-1, +1\}$ on a given document d_i, also return a *confidence estimate* C_{ij}, i.e., a numerical value representing the strength of their belief in the fact that D_{ij} is correct (the higher the value, the higher the confidence). We formalize this by taking a binary classifier to be a function $\hat{\Phi}_j : \mathcal{D} \rightarrow \mathbb{R}$ in which the sign of the returned value $D_{ij} \equiv sgn(\hat{\Phi}_j(d_i)) \in \{-1, +1\}$ indicates the binary decision of the classifier, and the absolute value $C_{ij} \equiv |\hat{\Phi}_j(d_i)|$ represents its confidence in the decision.

2.1 Supervised learning

In order to obtain the classifier $\hat{\Phi}$ we use a Machine Learning (ML) approach to TC. In ML a classifier is estimated through an automatic process of learning. The learner builds a classifier by observing the documents that a human annotator, who is an

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

expert in the domain of the considered textual data, has classified. For a category c_j the learner analyzes the characteristics of the documents labelled with both c_j and $\bar{c_j}$. This inductive process models the characteristics that a new, unseen, document should have in order to be classified under c_j. This solution to classification is also known as *supervised learning*, that is the activity of learning under the supervision of the knowledge extracted from the set of training documents. The key aspect for building an effective classifier is in the available resources, these are the documents in $\mathcal{D}$ and the categories in $\mathcal{C}$, which are assigned to the documents. The data resources are usually provided by the customer of the classification service, an organization or a company, that owns a knowledge base and wants to automate the activity of extending this knowledge.

In supervised learning we need an initial, classified, set of documents $D = \{d_1, \ldots, d_{|D|}\} \subset \mathcal{D}$. Documents in D are classified under $\mathcal{C}$, so the values of Φ are known for each pair $\langle d_i, c_j \rangle \in D \times \mathcal{C}$. This set is used for two fundamental goals: (i) building the classifier, that is eventually used on new documents; (ii) tuning and evaluating the classification model.

In a common experimental setting, D is divided in two subsets:

- The training set $Tr = \{d_1, \ldots, d_{|Tr|}\} \subset D$, that is used in the *learning* process, namely the construction of the classification model;
- The test set $Te = \{d_{|Tr|+1}, \ldots, d_{|D|}\} \subset D$, that is used for evaluating the classifier $\hat{\Phi}$ built on Tr. The evaluation of the effectiveness of the classifier is performed comparing the values of $sgn(\hat{\Phi}(d_i))$ against the values of $sgn(\Phi(d_i))$, for each $d_i \in Te$.

The key aspect of this experimental setting is that the classification model is built working exclusively on Tr. The purpose is to simulate a real scenario in which we do not know anything about the documents to be classified, as those belonging to Te.

The effectiveness of a classifier can be estimated on Tr through a k-fold cross-validation. In this approach Tr is divided in k partitions, so k experiments are conducted using the training/test approach. The validation is composed of k learning/-classification stages, i.e. k classifiers are built and evaluated on the training and test set pairs $\langle Tr_i = D - Te_i, Te_i \rangle$, $\forall i \in [1, k]$. The final evaluation is computed as the average of the k evaluations performed on every Te_i. In some learning algorithms it is possible to set parameters in order to adjust the specific behaviors of the learners. The k-fold cross-validation is a solid tool for exploring and selecting the parameter values: a validation is performed for each value, the best parameter values are the ones for which the k classifiers have the best average effectiveness.

In the next sections we introduce in detail the techniques for analyzing and representing textual data; two ML algorithms are discussed: SVMs and Boosting. Finally the reference evaluation measures are presented. All these elements, with the experimental methodology discussed above, are the basis of SATC methods and their evaluation.

2.1.1 Text representation

Text semantics have to be represented in a form that can be interpreted by the ML algorithms. Documents are passed under a process of *indexing*, that is performed uniformly for all the documents in $\mathcal{D}$. In this process the meaningful units of the texts are identified and each document is converted to a numerical representation. In TC each sample of the data is a document, and it is represented by a vector; each component of a document vector is a feature, and its value defines the feature weight; features are units, characteristics, which describe a document. The most common features for texts are the terms of the documents; a term can be identified as a word, so a term weight can be defined as the importance of the word for a document.

In the learning phase we identify the features from the available data, represented by the training set. In the vector space of documents each dimension is a term of the dictionary drawn from Tr, so the representation of d_i is $\mathbf{d_i} = \langle w_{1i}, \ldots, w_{|\mathcal{T}|i} \rangle$ where $\mathcal{T}$ is the dictionary, namely the set of terms that occur at least once in at least one document of Tr. If the dictionary is made of words, we call this approach Bag of Words (BOW). Word weights can be binary (i.e. 0 or 1 is the word is respectively present or not in the document) or real values: $w_{ki} \in [0, 1]$. The higher the weight, the greater is its contribution to the semantics of a document. With this approach words are taken as independent units, discarding the semantic dependencies of the language, which are not employed by the learners.

In order to obtain document vectors, a function for computing weights is necessary. In the case of binary weights we simply set w_{ki} to 1 if the term t_k occurs in the document d_i. A richer semantic could be the number of occurrences of the term, this measure is usually called *term frequency*, defined by the function $tf(t_k, d_i)$. The intuition behind the term frequency is that the representativeness of a term for a document is directly proportional with its frequency in that document. The problem with term frequency is that the most common terms of a language, like prepositions, conjunctions, etc., have high tf in every document. The same is true for the most common terms of a specific domain, e.g. the word "game" is very frequent in a training set about the subject "sport". In order to mitigate this problem, we can weight the term frequency by the inverse of the document frequency. The importance of a term is then a combination of its occurrences in the documents and its occurrences in the whole training set. The function $tfidf$ [70] is defined as

$$tfidf(t_k, d_i) = tf(t_k, d_i) \cdot log \frac{|Tr|}{df(t_k)}$$

where $df(t_k)$ is the number of documents in Tr in which t_k occurs. An alternative weighting function is the normalized $tfidf$, which returns values that fall in $[0, 1]$ interval. We can normalize the document vectors through the cosine normalization:

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

In the BOW approach some preprocessing steps are usually performed before the vectorization of documents. Textual data is cleaned in order to keep only the useful semantics and discards useless units of text, the standard process consists in filtering out punctuation and stop words. *Stop words* are the elements of a language that are used for the construction of the syntax, which do not carry useful semantics: articles, prepositions, conjunctions, "to be", "to have", etc. A successive common preprocessing step is *stemming*, in which words are reduced to their stem, base or root form. Multiple words can be related to the same stem (e.g. "logically" and "logics" are related to "logic"), by using stems we obtain a more compact and meaningful representation of the document semantics.

2.1.2 Support vector machines

Support vector machines (SVMs) [16] are supervised learning algorithms that learn from the training data by separating the space of the samples according to their classes. A SVMs learner builds a *hyperplane* in the multi-dimensional space of features, which better separates the samples that belong to a class from the others (binary classification case). The optimal separation is the one that brings about the largest margin between the hyperplane and the samples of both sides. The closest samples to the hyperplane are called the *support vectors*, as they alone define the decision boundary between the two sides.

One big advantage of SVMs is their modularity, that allows to train the model on samples that are not linearly separable. In TC it is common to explore only linear solutions (linear SVMs), as they have proven to be effective [17]. SVMs have been adopted in TC [36, 37] for their favorable characteristics: they behave efficiently for high dimension space inputs; they can handle irrelevant features and very sparse vectors, two aspects that are very common in textual data; it is possible to easily tune a SVM learner in order to overcome the problem of *overfitting*. However, SVMs are generally effective in TC with the standard setting of their parameters.

Multi-class multi-label TC with SVMs can be achieved employing the binary classification approach, so we build an independent classifier for each class c_j. In the vector space of documents we can define any hyperplane h as

$$\mathbf{w} \cdot \mathbf{d} - b = 0$$

For each category c_j, and any document d_i in D, we want to find a hyperplane such as

$$\begin{aligned} \mathbf{w_j} \cdot \mathbf{d_i} - b_j \geq +1 \quad if\, D_{ij} = +1 \\ \mathbf{w_j} \cdot \mathbf{d_i} - b_j \leq -1 \quad if\, D_{ij} = -1 \end{aligned}$$

This can be rewritten as:

$$D_{ij}(\mathbf{w_j} \cdot \mathbf{d_i} - b_j) - 1 \geq 0$$

The points that intersect the planes $h_{+1} : \mathbf{w_j} \cdot \mathbf{d_i} - b_j = +1$ and $h_{-1} : \mathbf{w_j} \cdot \mathbf{d_i} - b_j = +1$ are the support vectors, they are marked in red in Figure 2.1.

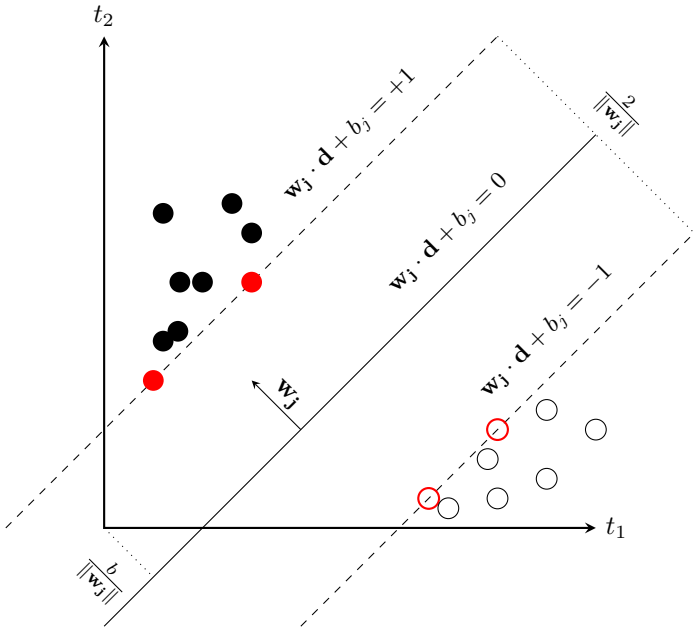


Figure 2.1: An example of a hyperplane which brings the largest separation margin of documents for class c_j. The vector space has two dimensions for terms t_1 and t_2.

In order to find the best hyperplane, i.e. the one for which the distance between it and the support vectors is maximum, we solve an optimization problem. We search for the vector $\mathbf{w_j}$ that maximizes the distance from h_{+1} to the hyperplane h, that is equivalent to the distance between h_{-1} and h:

$$\frac{\mathbf{w_j} \cdot \mathbf{d_i} - b_j}{\|\mathbf{w_j}\|} = \frac{1}{\|\mathbf{w_j}\|}$$

The problem to be solved is a minimization problem, with the constraints defined by the samples of the training set:

$$\min_{(\mathbf{w_j}, b_j)} \|\mathbf{w_j}\| \quad s.t.:$$
$$D_{ij}(\mathbf{w_j} \cdot \mathbf{d_i} - b_j) - 1 \geq 0 \quad \forall d_i \in Tr$$

We can alter the initial function $\min_{(\mathbf{w_j}, b_j)} \|\mathbf{w_j}\|$ into $\min_{(\mathbf{w_j}, b_j)} \frac{1}{2}\|\mathbf{w_j}\|^2$, thus the optimization problem is transformed in a quadratic programming problem, which can be solved with the method of Lagrange multipliers [16].

After the optimal separation is found, we can classify new documents according to their relative position to the hyperplane. In the learning phase we obtain the parameters of the function $\hat{\Phi}_j(d_i) = \mathbf{w_j^{best}} \cdot \mathbf{d_i} - b_j^{best}$, that returns strictly positive values if the document d_i is classified with the class c_j, strictly negative values otherwise. The

function returns 0 if the document lies on the hyperplane. The confidence of classification is then proportional to the distance of the document point from the decision boundary defined by the optimal hyperplane.

In [16] the authors introduce a further parameter into the learning algorithm, in order to manage noisy points - outliers that can negatively affect the decision boundary. The C parameter plays a role of regularization of the function to be minimized; it controls the relative importance of minimizing $\|\mathbf{w_j}\|$, that is equivalent to controlling the size of the margin. When we assign to C values close to 0, we allow samples near the decision boundary to violate the constraints of the minimization problem, so the learner excludes points that can easily overfit the model on the training data. The opposite happens when we set C to values close to ∞, the constraints imposed on the training samples are strictly followed in the solution of the optimization problem.

2.1.3 Boosted decision trees

The family of ML algorithms called *boosting* algorithms is based on the principle of combining several learners, in order to produce one effective classifier. The typical boosting method starts by creating weak classifiers, models whose classification effectiveness is low. Weak classifiers are usually built learning only a few (limited) aspects of the training data, consequently their classification accuracy is limited. In boosting several weak classifiers are built with the final goal of combining them in a global strong classifier, that is able to deal with all the characteristics of the data.

One popular variant of boosting is AdaBoost, its name stands for Adaptive Boosting. In AdaBoost weak classifiers are built iteratively, each one is tuned in order to perform better on the data that the previous weak classifier has misclassified, the algorithm adapts itself to the training data.

A variant of AdaBoost is MP-Boost [19], that is an improved version of AdaBoost.MH [72]. These two variants are optimized for multi-label classification, the weak classifiers they create are *decision trees*, trained on specific terms. Each weak classifier is built around a pivot term and sets up its classification on the presence or the absence of the term in the document (binary feature weights). The improvement of MP-Boost, compared to AdaBoost.MH, is in how pivot terms are chosen. At each iteration, more pivot terms are selected, one for each class. The algorithm then constructs a set of weak classifiers at each iteration, each of them is optimized for a specific class.

MP-Boost algorithm

In MP-Boost weak classifiers are generated iteratively, so if S is the number of iterations, we have $\hat{\Phi}^1 \dots \hat{\Phi}^S$ weak classifiers. At the end of each iteration the final classifier is obtained summing up the weak classifiers $\hat{\Phi} = \sum_{s=1}^{S} \hat{\Phi}^s$, so both the

Input:
a training set Tr, the number of iterations S
Output:
the classifier $\hat{\Phi}(d_i, c_j) = \sum_{s=1}^{S} \hat{\Phi}^s(d_i, c_j)$
Body:
set $P_1(d_i, c_j) = \frac{1}{|Tr| \cdot |\mathcal{C}|}$
for $s = 1, \ldots, S$ do:
1. pass distribution P_s to the weak learner;
2. get the weak classifier $\hat{\Phi}^s$ from the weak learner;
3. set $P_{s+1}(d_i, c_j) = \frac{P_s(d_i, c_j) \cdot exp(-\Phi(d_i, c_j) \cdot \hat{\Phi}_j^s)}{Z_j^s}$
   where $Z_j^s = \sum_{i=1}^{|Tr|} P_s(d_i, c_j) \cdot exp(-\Phi(d_i, c_j) \cdot \hat{\Phi}_j^s)$ is a normalization factor chosen so that $\sum_{i=1}^{|Tr|} \sum_{j=1}^{|\mathcal{C}|} P_{s+1}(d_i, c_j) = 1$

Figure 2.2: The MP-Boost algorithm

confidence estimates and the classification decisions of $\hat{\Phi}$ are obtained from the sum of the values of the single weak classifiers.

The algorithm (Figure 2.2) creates a weak classifier for each class at each iteration, then we have $|\mathcal{C}|$ weak classifiers at each iteration. For class c_j, at iteration s, the classifier $\hat{\Phi}_j^s$ is built. MP-Boost updates a distribution of weights on the training set pairs $\langle d_i, c_j \rangle$ at each iteration s. We define P_{s+1} as the distribution that is used by the algorithm at iteration $s+1$. The value of $P_{s+1}(d_i, c_j)$ is updated in order to capture the effectiveness of the classifiers $\hat{\Phi}_j^1 \ldots \hat{\Phi}_j^s$ in assigning class c_j to the document d_i. The weights are proportional to the ability of the classifiers of correctly assigning the class, the higher the weight, the more difficult the assignment. The weak classifier $\hat{\Phi}_j^{s+1}$, in order to better classify documents with higher weights, concentrates on those features that allows to better discriminate the class c_j. Every weak learner at iteration $s+1$ takes in input the training set and the distribution P_{s+1}.

The initial distribution P_1 is uniform; at iteration s weights are updated according to:

$$P_{s+1}(d_i, c_j) = \frac{P_s(d_i, c_j) \cdot exp(-\Phi(d_i, c_j) \cdot \hat{\Phi}_j^s)}{Z_j^s}$$

where

$$Z_j^s = \sum_{i=1}^{|Tr|} P_s(d_i, c_j) \cdot exp(-\Phi(d_i, c_j) \cdot \hat{\Phi}_j^s)$$

is a normalization factor, so $P_{s+1}(d_i, c_j)$ is a distribution, the sum of all the weight values is 1.

A weight is updated proportionally to its previous value, it is increased if the document is wrongly classified with class c_j, decreased otherwise. The product

$-\Phi(d_i, c_j) \cdot \hat{\Phi}_j^s$ is positive if the target function and the weak classifier have different sign, in this case the exponential function exp is ≥ 1.

MP-Boost weak classifiers

Text is represented with binary feature weights. The weight w_{ki} of a document vector d_i is equal to 1 if the term t_k is contained in the document, 0 otherwise. A weak classifier $\hat{\Phi}_j^s$ is a decision tree, with only one root and two leaves. It returns a specific real value if a term is present in the document and a different value if it is not. The term is called *pivot term*, thus we have $|\mathcal{C}|$ pivot terms at each iteration s. The weak classifier is defined as

$$\hat{\Phi}_j^s(d_i) = \begin{cases} a_0^j \text{ if } w_{ki}^j = 0 \\ a_1^j \text{ if } w_{ki}^j = 1 \end{cases}$$

Creating a weak classifier means choosing a term t_k^j and the values a_0^j and a_1^j. These three parameters are obtained during the learning phase, that consists in minimizing the error of the weak classifier. This error can be measured by the normalization factor Z_j^s.

In order to obtain $\hat{\Phi}_j^s$ these two steps are executed (at iteration s):

1. Select among all $\hat{\Phi}_j^s$ that have a specific term as pivot term, the one which minimizes Z_j^s. In this step we create $|\mathcal{T}|$ weak classifiers, and for each one we choose the values a_0^j and a_1^j.
2. Select among all the best weak classifiers previously created, the one which minimize Z_j^s. In this step we choose the unique pivot term t_k^j for the current iteration.

The first step is critical, since we can not enumerate all the possible values of a_0^j and a_1^j. MP-Boost deals with this step using the same approach of AdaBoost.MH; this approach is argued in [71], where the authors have proven that:

$$\hat{\Phi}_j^{best(k)}(d_i) = \begin{cases} \frac{1}{2} log \frac{W_{+1}^{0jk}}{W_{-1}^{0jk}} \text{ if } w_{ki}^j = 0 \\ \frac{1}{2} log \frac{W_{+1}^{1jk}}{W_{-1}^{1jk}} \text{ if } w_{ki}^j = 1 \end{cases}$$

where $best(k)$ denotes the best weak classifier for term t_k^j, selected at the first step, and where

$$W_b^{xjk} = \sum_{i=1}^{|tr|} P_s(d_i, c_j) \cdot [\![w_{ki}^j = x]\!] \cdot [\![\Phi(d_i, c_j) = b]\!]$$

for $b \in \{-1, +1\}$, $x \in \{0, 1\}$, and where $[\![\pi]\!]$ is the characteristic function of predicate π, so if π is true the function returns 1, 0 otherwise. The values W_b^{xjk} are equivalent to the weight, with respect to the distribution P_s, of the documents that contain (or does not contain) the term t_k^j, which are classified (or are not) with c_j.

Reminding that $\hat{\Phi}^s(d_i, c_j) \equiv \hat{\Phi}_j^s(d_i)$, the final classifier is $\hat{\Phi}(d_i, c_j) = \sum_{s=1}^{S} \hat{\Phi}^s(d_i, c_j)$, where S is the number of iterations (the only free parameter of MP-Boost algorithm). The classification values are thus a combination of the values a_0^j and a_1^j of the pivot terms of each iteration.

2.2 Evaluating text classification

Several measures exist for evaluating the accuracy of text classifiers. It is possible to estimate the accuracy of a classifier during the learning phase, performing a validation step (e.g. k-fold cross-validation). While these operations are performed on Tr, the final evaluation is executed on Te. We use Te in order to simulate a "production" phase, in which the classifier is applied on new, unseen and unlabelled documents. The evaluation scores we obtain on Te are used for comparing the classifier effectiveness. We implicitly refer to the evaluation on Te hereafter.

In a classification task, for each assignment of a class to a document, we can define four possible events, according to the true label of the document. We can obtain a true positive, a true negative, a false positive or a false negative. The terms "positive" and "negative" refer to the predicted assignment made by the classifier, while "true" and "false" refer to the correctness of the prediction with respect to the real assignment of the document. We use $TP(ij)$ to indicate that $\hat{\Phi}_j(d_i)$ is a true positive, and use $FP(ij)$ (false positive), $FN(ij)$ (false negative), and $TN(ij)$ (true negative) with analogous meanings; e.g. if $TN(ij)$ is true, then the classifier has correctly not assigned the class c_j to the document d_i.

With TP_j, FP_j, FN_j, and TN_j we indicate the numbers of true positives, false positives, false negatives and true negatives in the test set Te for category c_j. The four values can be formulated in a *contingency table* as in Figure 2.3, each value is represented in a *contingency cell*.

		predicted	
		Y	**N**
true	**Y**	$TP = 4$	$FP = 3$
	N	$FN = 4$	$TN = 9$

Figure 2.3: An example of a contingency table of a single-label classification of $TP + TN + FP + FN = 20$ documents.

2.2.1 Evaluation measures

The well known notions of *precision* and *recall* in Information Retrieval, can be designed as evaluation measures in TC. The two measure can be formulated on the contingency table:

$$precision_j(\hat{\Phi}_j(Te)) = \frac{TP_j}{TP_j + FP_j}$$
$$recall_j(\hat{\Phi}_j(Te)) = \frac{TP_j}{TP_j + FN_j}$$

For a class c_j, $precision_j$ is the fraction of correct positive predictions on all the positive predictions, while $recall_j$ is the fraction of correct positive predictions on the number of documents in Te belonging to c_j. The values of both functions range between 0 (worst) and 1 (best).

Precision and recall can have different priority in the evaluation of a TC system, according to the type of application we evaluate. An example of application that gives precedence to precision is the classification of medical reports, where the category set is made of diagnosable diseases. Assigning the right classes is critical for choosing the right treatment. An example of application that requires high recall is the e-mail spam filter, where e-mails are classified as positive if they are not spam. In this case high recall means low probability that good e-mails are wasted in the spam folder.

A measure that combines precision and recall is the F-measure. The harmonic mean of precision and recall is defined as the F_1 measure:

$$F_1(\hat{\Phi}_j(Te)) = \frac{2 \cdot precision_j \cdot recall_j}{precision_j + recall_j} = \\ = \frac{2 \cdot TP_j}{2 \cdot TP_j + FP_j + FN_j}$$

F_1 is a specific instance of the generic F_β measure [78]. The parameter β is used to weight the importance of precision over recall, with positive values lower than 1 we put emphasis on precision while with values grater than 1 we put emphasis on recall. With $\beta = 1$ we have the F_1 defined above.

$$F_\beta(\hat{\Phi}_j(Te)) = (1 + \beta^2) \cdot \frac{precision_j \cdot recall_j}{(\beta^2 \cdot precision_j) + recall_j} = \\ = \frac{(1 + \beta^2) \cdot TP_j}{(1 + \beta^2) \cdot TP_j + FP_j + (1 + \beta^2) \cdot FN_j}$$

Note that F_1 is undefined when $TP_j = FP_j = FN_j = 0$; in this case we take $F_1(\hat{\Phi}_j(Te)) = 1$, since $\hat{\Phi}_j$ has correctly classified all documents as negative examples. Note also that in the evaluation based on F_1 the correct negative classifications are excluded, in fact the function does not embrace the true negative. We will concentrate on the F_1 measure in this dissertation, but it will be successively evident that any kind of measure computed on the contingency table can be used in SATC methods.

We have restricted the discussion of evaluation measures on binary classification. In order to evaluate the effectiveness of multi-class multi-label TC we use two approaches for averaging on the classes:

- *macro-averaged* F_1 (noted F_1^M or macro F_1), which is obtained by computing the class-specific F_1 values and averaging them across all the $c_j \in \mathcal{C}$.

$$F_1^M(Te) = \frac{\sum_{j=1}^{|\mathcal{C}|} F_1(\hat{\Phi}_j(Te))}{|\mathcal{C}|}$$

- *micro-averaged* F_1 (noted F_1^{μ} or micro F_1), which is obtained by computing the F_1 on a global contingency table, where each contingency cell is the sum of the respective class-specific contingency cells.

$$F_1^{\mu}(Te) = \frac{2\sum_{j=1}^{|\mathcal{C}|} TP_j}{2\sum_{j=1}^{|\mathcal{C}|} TP_j + \sum_{j=1}^{|\mathcal{C}|} FP_j + \sum_{j=1}^{|\mathcal{C}|} FN_j}$$

The two methods of averaging can be applied to any measure, they differ in the importance given to the frequency of assigned classes. Macro-averaging weights equally each class; with a macro-averaged measure the evaluation of infrequent classes, which are more likely to produce errors, has the same importance of the evaluation of the most frequent ones. Micro-averaging gives the same importance to each binary classification, so it is independent of the distribution of the classes.

Many evaluation measures exist for classification, in this dissertation we will often use "accuracy of classification" concerning to any measure of effectiveness. We will quantify this definition mostly with the F_1 measure.

2.2.2 Datasets for text classification benchmarks

Text collections are set of documents and categories made for benchmarking purposes. These collections are publicly available and they come with the true labels assigned to documents, in order to use the entire set of samples for evaluation. They also come with configurations of dataset splitting, in order to define common training and test sets across experiments. TC evaluations are performed choosing datasets together with specific splitting configurations.

Our first dataset is the REUTERS-21578 [45] corpus. It consists of a set of 12,902 news stories, partitioned (according to the standard "ModApté" split we adopt) into a training set of 9603 documents and a test set of 3299 documents. The documents are labelled by 118 categories; the average number of categories per document is 1.08, ranging from a minimum of 0 to a maximum of 16; the number of positive examples per class ranges from a minimum of 1 to a maximum of 3964. In our experiments we restrict our attention to the 115 categories with at least one positive training example. This dataset is publicly available[2] and it is probably the most widely used benchmark in TC research; this fact allows other researchers to easily replicate the results of our experiments.

Another dataset we use is OHSUMED [31], a test collection consisting of a set of 348,566 MEDLINE references spanning the years from 1987 to 1991. Each entry consists of summary information relative to a paper published on one of 270 medical journals. The available fields are title, abstract, MeSH indexing terms, author, source, and publication type. Not all the entries contain abstract and MeSH indexing terms. In our experiments we scrupulously follow the experimental setup presented in

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

[48]. In particular, (i) we use for our experiments only the 233,445 entries with both abstract and MeSH indexing terms; (ii) we use 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 use the *main heading* MeSH index terms assigned to the entries. Concerning this latter point, we restrict our experiments to the 97 MeSH index terms that belong to the *Heart Disease* (HD) subtree of the MeSH tree, and that have at least one positive training example. This is the only point in which we deviate from [48], which experiments only on the 77 most frequent MeSH index terms of the HD subtree.

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

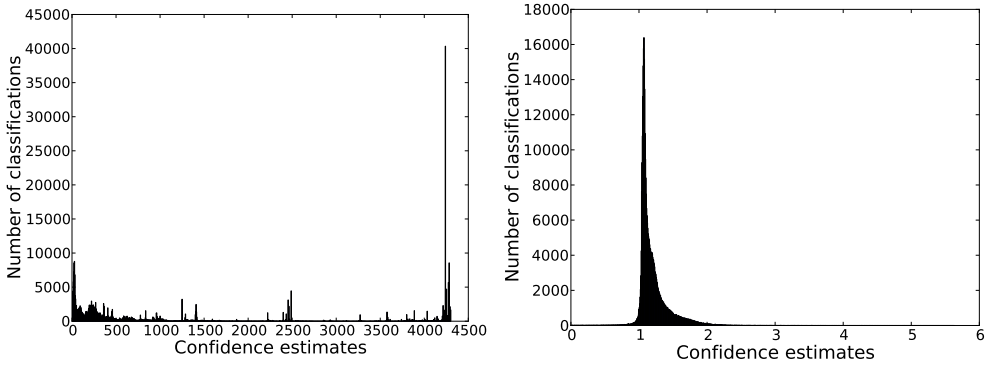
1. All of them are publicly available and very widely used in TC research, which allows other researchers to easily replicate the results of our experiments.
2. OHSUMED is one of the largest datasets used to date in TC research, which lends robustness to our results.

The main characteristics of our datasets are conveniently summarized in Table 2.1. The evaluations on the two collections' test sets point out the effectiveness of the two learners. It is clear the OHSUMED collection is a more difficult benchmark dataset for TC. The learners get different effectiveness on both collections, while MP-BOOST is stronger on F_1^M, SVMs is better on F_1^μ; this means that SVMs is less effective on the infrequent classes (assigned to very few documents), so the accuracy averaged on all classes is negatively affected by the rare ones, but it is more effective on the few very frequent categories. Datasets such as REUTERS21578 have a large number of infrequent categories in the test set, which predominate in the computation of macro-averaged classification accuracy. For example in REUTERS21578 we have 54 categories with frequency ≤ 10, 45 with frequency ≤ 100, 14 with frequency ≤ 1000, and two categories with frequency 1650 and 2877.

Dataset	$\|Tr\|$	$\|Te\|$	$\|\mathcal{C}\|$	ACD	F_1^M		F_1^μ	
					MP-B	SVMs	MP-B	SVMs
OHSUMED	183229	50216	97	0.132	.447	.423	.611	.676
REUTERS-21578	9603	3299	115	1.135	.608	.527	.848	.860

Table 2.1: Characteristics of the test collections used. From left to right we report the number of training documents (2), the number of test documents (3), the number of classes (4), and the average number of classes per test document (5). Columns 6-9 report the initial accuracy (both F_1^M and F_1^μ) generated by the MP-BOOST and SVMs classifiers.

Another interesting analysis is on the confidence scores of classifiers, that is useful for understanding some intrinsic processes of the learners. The values returned by the classifiers range in $[-\infty, +\infty]$, so the confidence values range in $[0, +\infty]$, but they are distributed differently, according to the machine learning algorithm. In Figure 2.4 we show the histograms of the distributions of the confidence estimates returned by the two classifiers, applied on the test set of Reuters21578. We notice that SVMs produces values more uniformly distributed, in a narrow range, while MP-Boost confidence values are concentrated on a discrete set of values. This ensues different confidence estimates of the two learners, an aspect that influences the development of SATC methods presented in Chapter 3.



(a) MP-Boost confidences distribution.

(b) SVMs confidences distribution.

Figure 2.4: Cumulative distributions of confidence estimates for MP-Boost (Figure a) and SVMs (Figure b) on the Reuters21578 test set. The total number of classifications is $|Te| \cdot |\mathcal{C}|$, that for Reuters21578 is equivalent to $3299 \cdot 115 = 379385$

2.3 Conclusions

In this chapter we have introduced the basis for understanding text classification and its technical aspects. The notions described have been studied in depth in the last years, in the machine learning community. We have limited this dissertation to few of the many techniques for achieving automatic classification of texts, and in the next chapters we apply these techniques to a new task connected to TC.

Semi-automated text classification originates with the idea of improving the current standard methods for text classification. In an established TC framework we can plug in SATC methods and evaluation functions, without modifying the solid approaches defined in this chapter. This allow us to extend any classification system, bringing new tools into the TC ecosystem.

3

A Ranking Method for Semi-Automated Text Classification

Semi-Automated Text Classification (SATC) is the task of improving the final accuracy of an automatic TC process via human inspection of the classified documents. We introduced the importance of this task in a scenario in which accuracy of classification is critical. If we can rely on the support of human work we can aim to maximize the improvement in classification accuracy. In SATC we have to deal with the cost of human effort, with the goal of minimizing it and making the best of the limited availability of human validators. For this reason the principal objective is to develop methods for ranking the documents to be validated. We want to rank Te in order to maximize the expected increase in classification effectiveness, obtained by a human annotator that inspects a subset of Te by working down from the top of the list.

In this chapter we are going to present a ranking method based on an "utility function". The method and the solutions for its implementation are discussed in detail. We introduce the chapter with a worked-out example of SATC (Section 3.1). Sections 3.2 and 3.3 describe our basic utility-theoretic strategy for ranking the automatically labelled documents, while in Section 3.4 we propose a novel effectiveness measure for this task based on a probabilistic user model. Section 3.5 reports the results of our experiments, in which we test the effectiveness of our utility-theoretic ranking strategy, by simulating the work of a human annotator that inspects variable portions of the labelled test set.

3.1 A worked-out example

In order to see how human annotators may be effectively supported in their post-editing work, let us look at a specific example. Let us assume that the classifying task consists in deciding whether a given class applies or not to any of a set of unlabelled documents (binary case). Let us also assume that a set of unlabelled documents have been automatically classified; for simplicity of illustration we here assume that this set consists of 20 documents only. We can measure the accuracy obtained in this automatic classification job by (a) choosing an accuracy measure, (b) filling out a

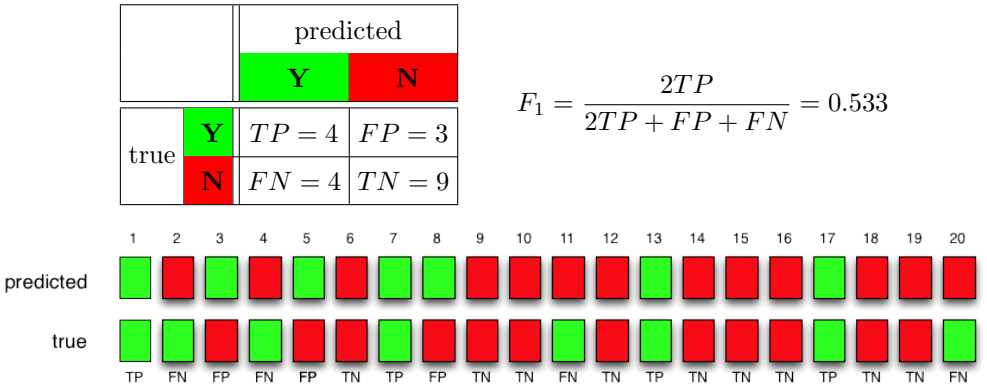


Figure 3.1: A worked-out example, representing a contingency table (upper left part of the figure) deriving from the automatically labelled examples (lower part) and from which accuracy is computed (upper right part).

contingency table, and (c) evaluating the chosen measure on this table. For illustration purposes we assume that our accuracy measure is the well-known F_1, described in Chapter 2. Figure 3.1 depicts a situation in which the automatic classification process has returned 4 true positives, 3 false positives, 4 false negatives, and 9 true negatives, resulting in a value of $F_1 = \frac{2 \cdot 4}{(2 \cdot 4)+3+4} = 0.533$. The 20 documents are represented in the two rows at the bottom via green and red cards; the upper row represents the classification decisions of the system ("predictions"), while the lower row represents the correct decisions that an ideal system would have taken. A green card represents a "yes" (the documents has the category), while a red card represents a "no" (the documents does not have the category); a correct decision is thus represented by the upper and lower card in the same column having the same colour.

Let us imagine a scenario in which the customer insists that the data must be classified with an accuracy level of at least $F_1 = 0.800$. In this case, after checking that the value of F_1 that the automatic classifier has obtained is 0.533, the human annotator decides to validate some of the documents until the desired level of accuracy has been obtained[1]. Let us assume that the coder examines the documents at the bottom of Figure 3.1 in left-to-right order. The first document that the annotator examines is a true positive; no correction needs to be done, the value of F_1 is unmodified, and the annotator moves on to the second document. This is a false negative, and correcting it decreases FN by 1 and increases FP by 1, which means that F_1 now becomes $F_1 = \frac{2 \cdot 5}{(2 \cdot 5)+3+3} = 0.625$. The third is a false positive, and correcting it decreases FP by 1 and increases TN by 1, which means that F_1 now becomes $F_1 = \frac{2 \cdot 5}{(2 \cdot 5)+2+3} = 0.667$.

[1] Actually, the human annotator does *not* know the level of accuracy that the automatic classifier has obtained, since she does not know the true classification of documents. However, this level of accuracy can be at least *estimated* via a k-fold cross-validation.

The fourth is a false negative, which brings about $F_1 = \frac{2 \cdot 6}{(2 \cdot 6)+2+2} = 0.750$, and the fifth is a false positive, which yields $F_1 = \frac{2 \cdot 6}{(2 \cdot 6)+1+2} = 0.800$. At this point, having reached the minimum level of accuracy required by the customer, the human annotator's task is over.

Bringing F_1 from 0.533 up to 0.800 has required the validation of 5 documents, i.e., 25% of the entire set. Could the human annotator have achieved the same improved in accuracy with a smaller effort (i.e., by validation fewer documents)? Could she, by putting in the same effort, have reached a level of accuracy higher than $F_1 = 0.800$? The answer to both questions is yes, and the key to doing better is the order in which the documents are validated.

For instance, the fact that the first validated document was a true positive was suboptimal. Validating documents that have been classified correctly is, for the human annotator, wasted time, since no correction is performed and F_1 remains thus unmodified. Of course, there is no way to know in advance if the document has been classified incorrectly or not. However, it would at least be desirable to know how *likely* it is that the document has been classified incorrectly, i.e., to know its *probability of misclassification*; in this case, the system might rank the automatically classified documents in such a way as to top-rank the documents that have the highest such probability.

A second fact that jumps to the eye is that the increase in accuracy (i.e., the *gain*) determined by the correction of a false negative is higher (sometimes much higher) than the gain determined by the correction of a false positive. For instance, in correcting the second document (a false negative) F_1 jumped from 0.533 to 0.625 (a +17.1% relative increase), while in correcting the third document (a false positive) F_1 only moved from 0.625 to 0.667 (a mere +6.6% relative increase). This is not an idiosyncrasy of the F_1 measure, since for many accuracy measures the gain deriving from the correction of a false positive is different than the one deriving from the correction of a false negative[2]. So, the fact that two false positives were inspected and corrected while two false negatives were left uninspected and uncorrected (in 11th and 20th position, respectively) was also suboptimal. This means that the system should, other things being equal, rank higher those documents (the false negatives, in our case) that bring about a higher gain when corrected. In the example of the previous section, had we ranked the four false negatives at the top four rank positions and a false positive at the fifth, the same amount of human validation work would have brought about an increase in F_1 from 0.533 to 0.889 (instead of 0.800); had we instead been happy with reaching $F_1 = 0.800$, the human annotator would have reached it

[2] Note that this asymmetry holds despite the fact that F_1 pays equal attention to the ability of the system to avoid false positives (i.e. precision) and to the ability of the system to avoid false negatives (i.e. recall). In other measures that pay, e.g., more attention to recall than to precision, the difference between the gain obtained by correcting a false positive and the gain obtained by correcting a false negative is amplified.

by inspecting and correcting only the four top-ranked documents (actually, by doing this she would have reached $F_1 = 0.842$).

In sum, we have learnt two key facts.

The first fact is that the order in which the human annotator validates the automatically classified documents is what determines the cost-effectiveness of her work. This fact should come as no surprise: the task of ranking a set of digital objects in terms of perceived usefulness to a given task is of paramount importance in nowadays' computer science as a whole, as perfectly exemplified by todays' search engines.

The second fact we have learnt is that, if we want to order these documents so as to maximise the cost-effectiveness of the human annotator's work, we should take two main factors into account, i.e., (a) the *probability* of misclassification of a given document, and (b) the *gain* in classification accuracy that the document brings about once inspected and corrected.

3.2 Ranking by probability of misclassificaton

We have seen in Chapter 2 how the scores C_{ij} are generated by the different classification algorithms. The confidence returned by the classifier, on a predicted classification, could be directly used for estimating the probability of a wrong decision. A ranking method based on the classifier confidence puts in the first positions the documents for which the confidence is lower. We use a probabilistic approach to this method, in order to define a ground ranking model, which we will "evolve" during the definition of the successive ranking methods.

For the moment being, let us concentrate on the binary case, i.e., let us assume there is a single class c_j that needs to be separated from its complement $\overline{c}_j$. We define two possible disjoint events after a positive prediction and other two for negative predictions, each event corresponds to an outcome in the contingency table. We have then a set of mutually disjoint events $\Omega = \{tp_j, fp_j, fn_j, tn_j\}$, each of these events implicitly refers to the document d_i under scrutiny (e.g., tp_j denotes the event "document d_i is a true positive for class c_j"). We can define the following probabilities:

- $P(tp_j|D_{ij} = +1)$ the probability of a correct positive prediction;
- $P(fp_j|D_{ij} = +1)$ the probability of a wrong positive prediction;
- $P(tn_j|D_{ij} = -1)$ the probability of a correct negative prediction;
- $P(fn_j|D_{ij} = -1)$ the probability of a wrong negative prediction.

We have that $P(fp_j|D_{ij} = +1) = 1 - P(tp_j|D_{ij} = +1)$ so we only need to compute $P(fp_j|D_{ij} = +1)$, similarly for negative predictions we have $P(fn_j|D_{ij} = -1) = 1 - P(tn_j|D_{ij} = -1)$ so we are left with computing $P(fn_j|D_{ij} = -1)$. We derive the probabilities $P(\cdot)$ by assuming that the confidence scores C_{ij} generated by $\hat{\Phi}_j$ can be trusted (i.e., that the higher C_{ij}, the higher the probability that D_{ij} is correct), and by applying to C_{ij} a *generalized logistic function* $f(z) = e^{\sigma z}/(e^{\sigma z} + 1)$. This results in

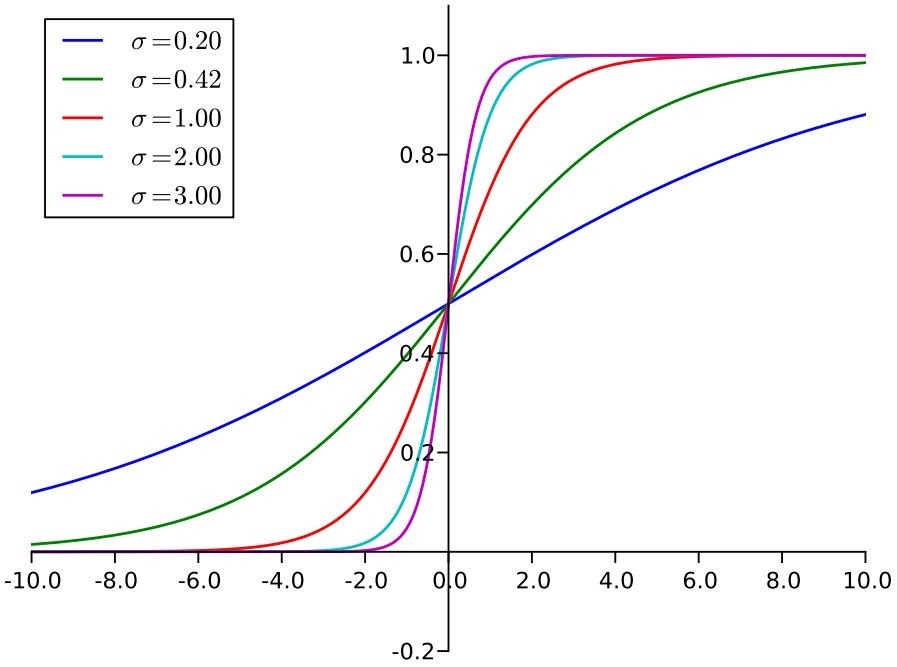


Figure 3.2: The generalized logistic function.

$$\begin{aligned} P(fp_j|D_{ij} = +1) &= 1 - \frac{e^{\sigma C_{ij}}}{e^{\sigma C_{ij}} + 1} \\ P(fn_j|D_{ij} = -1) &= 1 - \frac{e^{\sigma C_{ij}}}{e^{\sigma C_{ij}} + 1} \end{aligned} \tag{3.1}$$

The generalized logistic function (see Figure 3.2) has the effect of monotonically converting scores ranging on $(-\infty, +\infty)$ into real values in the $[0.0, 1.0]$ range. When $C_{ij} = 0$ (this happens when $\hat{\Phi}_j$ has no confidence at all in its own decision D_{ij}), then

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

i.e., the probability of correct classification and the probability of misclassification are identical. Conversely, we have

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

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

The reason why we use a *generalized* version of the logistic function instead of the standard version (which corresponds to the case $\sigma = 1$) is that using this latter within Equation 3.1 would give rise to a very high number of zero probabilities of misclassification, since the standard logistic function converts every positive number above a certain threshold (≈ 36) to a number that standard implementations round up to 1 even by working in double precision. By tuning the σ parameter (the *growth rate*) we can tune the speed at which the right-hand side of the sigmoid asymptotically approaches 1, and we can thus tune how evenly Equation 3.1 distributes the confidence values across the $[0.0, 0.5]$ interval.

The process of optimizing σ within Equation 3.1 is usually called *probability calibration*. How we actually optimize σ is discussed in Section 3.5.1.

3.3 Utility theory

In the worked-out example we have discussed the need of taking into account for probabilities and gains. This immediately evokes *utility theory*, an extension of probability theory that incorporates the notion of *gain* (or *loss*) that accrues from a given course of action [4, 79]. Utility theory is a general theory of rational action under uncertainty, and as such is used in many fields of human activity. For instance, utility theory is of paramount importance in betting, since in placing a certain bet we take into account (a) the probabilities of occurrence that we subjectively attribute to a set of outcomes (say, to the possible outcomes of the Arsenal FC vs. Chelsea FC game), and (b) the gains or losses that we obtain, having bet on one of them, if the various outcomes materialise.

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

$$U(\alpha_j, \Omega) = \sum_{\omega_i \in \Omega} P(\omega_i) G(\alpha_j, \omega_i) \tag{3.2}$$

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

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

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

How does this translate into a method for ranking automatically labelled documents? Assume we have a set $D = \{d_1, ..., d_n\}$ of such documents that we want to rank, and that c_j is the class we deal with. For instantiating Equation 3.2 concretely we need

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

Let us discuss each of these steps in turn.

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

Concerning Step 2, we have discussed in Section 3.1 that the increase in accuracy that derives from validating a document depends on whether the document is a true positive, a false positive, a false negative, or a true negative; as a consequence, we will take $\Omega = \{tp_j, fp_j, fn_j, tn_j\}$, which is equivalent to the homonym set defined in Section 3.2. Our utility function has thus the form

$$U_j(d_i, \Omega) = \sum_{\omega_k \in \{tp_j, fp_j, fn_j, tn_j\}} P(\omega_k) G(d_i, \omega_k) \tag{3.3}$$

We will define the utility function in detail in the next section. How to address Step 3 (defining the gains) will be the subject of Sections 3.3.2 and 3.3.3, while Step 4 (computing the probabilities of occurrence) follows the approach discussed in Section 3.2.

3.3.1 Ranking by utility

The policy we propose for ranking the automatically labelled documents in $\hat{\Phi}_j(Te)$ makes use of a function $U_j(d_i, \Omega)$ that estimates the utility, for the aims of increasing $F_1(\hat{\Phi}_j(Te))$, of manually inspecting the label D_{ij} attributed to d_i by $\hat{\Phi}_j$.

Upon submitting document d_i to classifier $\hat{\Phi}_j$, a positive or a negative decision can be returned. If a positive decision is returned (i.e. $D_{ij} = +1$) then the mutually disjoint events tp_j and fp_j can occur, while if this decision is negative (i.e. $D_{ij} = -1$)

then the mutually disjoint events fn_j and tn_j can occur. We thus naturally define the two utility functions

$$\begin{aligned} U_j(d_i, \Omega^+) &= P(tp_j|D_{ij} = +1) \cdot G(d_i, tp_j) + \\ &\quad P(fp_j|D_{ij} = +1) \cdot G(d_i, fp_j) \\ U_j(d_i, \Omega^-) &= P(fn_j|D_{ij} = -1) \cdot G(d_i, fn_j) + \\ &\quad P(tn_j|D_{ij} = -1) \cdot G(d_i, tn_j) \end{aligned} \tag{3.4}$$

with $U_j(d_i, \Omega^+)$ addressing the case of a positive decision and $U_j(d_i, \Omega^-)$ the case of a negative decision, so $\Omega^+ = \{tp_j, fp_j\}$ and $\Omega^- = \{fn_j, tn_j\}$. We define

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

as the utility function, embracing both positive and negative decisions.

3.3.2 Validation gains

We equate $G(d_i, fp_j)$ in Equation 3.4 with the average increase in $F_1(\hat{\Phi}_j(Te))$ that would derive by manually validating the label attributed by $\hat{\Phi}_j$ to a document d_i in FP_j. We call this the *validation gain* of a document in FP_j. Note that validations gains are independent of a particular document, i.e., $G(d', fp_j) = G(d'', fp_j)$ for all $d', d'' \in Te$. Analogous arguments apply to $G(d_i, tp_j)$, $G(d_i, fn_j)$, and $G(d_i, tn_j)$.

Quite evidently, $G(d_i, tp_j) = G(d_i, tn_j) = 0$, since when the human annotator validates the label attributed to d_i by $\hat{\Phi}_j$ and finds out it is correct, she will not modify it, and the value of $F_1(\hat{\Phi}_j(Te))$ will thus remain unchanged. This means that Equation 3.4 simplifies to

$$\begin{aligned} U_j(d_i, \Omega^+) &= P(fp_j|D_{ij} = +1) \cdot G(d_i, fp_j) \\ U_j(d_i, \Omega^-) &= P(tn_j|D_{ij} = -1) \cdot G(d_i, tn_j) \end{aligned}$$

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

$$\begin{aligned} G(d_i, fp_j) &= \frac{1}{FP_j}(F_1^{FP}(\hat{\Phi}_j(Te)) - F_1(\hat{\Phi}_j(Te))) = \\ &= \frac{1}{FP_j}\left(\frac{2TP_j}{2TP_j + FN_j} - \frac{2TP_j}{2TP_j + FP_j + FN_j}\right) \end{aligned} \tag{3.5}$$

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

$$G(d_i, fn_j) = \frac{1}{FN_j}(F_1^{FN}(\hat{\Phi}_j(Te)) - F_1(\hat{\Phi}_j(Te))) = \\ = \frac{1}{FN_j}(\frac{2(TP_j + FN_j)}{2(TP_j + FN_j) + FP_j} - \frac{2TP_j}{2TP_j + FP_j + FN_j}) \tag{3.6}$$

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

3.3.3 Smoothing contingency cell estimates

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

However, these maximum-likelihood cell count estimates need to be smoothed, so as to avoid zero counts. In fact, if $\hat{TP}_j^{ML} = 0$ it would derive from Equation 3.5 that there is nothing to be gained by correcting a false positive, which is counterintuitive. Similarly, if $\hat{FP}_j^{ML} = 0$ the very notion of $F_1^{FP}(\hat{\Phi}_j)$ would be meaningless, since it does not make sense to speak of "removing a false positive" when there are no false positives; and the same goes for $\hat{FN}_j^{ML}$.

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

Smoothing has extensively been studied in language modelling for speech processing [14] and for ad hoc search in IR [87]. However, the present context is slightly different, in that we need to smooth contingency tables, and not (as in the cases above) language models. In particular, while the $\hat{TP}_j^{ML}$, $\hat{FP}_j^{ML}$, and $\hat{FN}_j^{ML}$ are the obvious counterparts of the document model resulting from maximum-likelihood estimation,

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

there is no obvious counterpart to the "collection model", thus making the use of, e.g., Jelinek-Mercer smoothing problematic. A further difference is that we here require the smoothed counts not only to be nonzero, but also to be ≥ 1 (a requirement not to be found in language modelling).

Smoothing has also been studied specifically for the purpose of smoothing contingency cell estimates [10, 76]. However, these methods are inapplicable to our case, since they were originally conceived for contingency tables characterized by a small (i.e., ≤ 1) ratio between the number of observations (which in our case is $|Te|$) and the number of cells (which in our case is 4); our case is quite the opposite. Additionally, these smoothing methods do not operate under the constraint that the smoothed counts should all be ≥ 1, which is a hard constraint for us.

For all these reasons, rather than adopting more sophisticated forms of smoothing, we adopt simple *additive smoothing* (also known as *Laplace smoothing*), a special case of Bayesian smoothing using Dirichlet priors [87] which is obtained by adding a fixed quantity to each of $\hat{TP}_j^{ML}$, $\hat{FP}_j^{ML}$, $\hat{FN}_j^{ML}$. As a fixed quantity we add 1, since it is the quantity that all our cell counts need to be greater or equal to for Equations 3.5 and 3.6 to make sense. We denote the resulting estimates by $\hat{TP}_j^{La}$, $\hat{FP}_j^{La}$, $\hat{FN}_j^{La}$.

As it will be clear in Section 3.5 and following, this simple form of smoothing proves almost optimal, which seems to indicate that there is not much to be gained by applying more sophisticated smoothing methods to our problem context.

Note that we apply smoothing in an "on demand" fashion, i.e., we check if the contingency table needs smoothing at all (i.e., if any of $\hat{TP}_j^{ML}$, $\hat{FP}_j^{ML}$, $\hat{FN}_j^{ML}$ is < 1) and we smooth it only if this is the case. The reason why we adopt this "on-demand" policy will be especially apparent in Chapter 4.

3.3.4 Ranking by total utility

Our function $U_j(d_i, \Omega)$ of Section 3.3 is thus obtained by plugging Equations 3.5 and 3.6 into Equation 3.3. Therefore, we are now in a position to compute, given an automatically classified document d_i and a class c_j, the utility, for the aims of increasing $F_1(\hat{\Phi}_j(Te))$, of manually validating the label D_{ij} attributed to d_i by $\hat{\Phi}_j$.

Now, let us recall from Chapter 2 that our goal is addressing not just the binary, but the multi-class multi-label TC case, in which binary classification must be accomplished simultaneously for $|\mathcal{C}| \geq 2$ different classes. It might seem sensible to propose ranking, for each $c_j \in \mathcal{C}$, all the automatically labelled documents in Te in decreasing order of their $U_j(d_i, \Omega)$ value. Unfortunately, this would generate $|\mathcal{C}|$ different rankings, and in an operational context it seems implausible to ask a human annotator to scan $|\mathcal{C}|$ different rankings of the same document set (this would mean reading the same document $|\mathcal{C}|$ times in order to validate its labels). As suggested in [20] for active learning, it seems instead more plausible to generate a *single* ranking, according to a score $U(d_i, \Omega)$ that is a function of the $|\mathcal{C}|$ different $U_j(d_i, \Omega)$ scores. In such a way, the human annotator will scan this single ranking from the top, validating all the

$|\mathcal{C}|$ different labels for d_i before moving on to another document. As the criterion for generating the overall utility score $U(d_i, \Omega)$ we use *total utility*, corresponding to the simple sum

$$U(d_i, \Omega) = \sum_{c_j \in \mathcal{C}} U_j(d_i, \Omega) \tag{3.7}$$

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

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

3.4 Expected normalized error reduction

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

3.4.1 Error reduction at rank

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

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

$$ER_\rho(n) = \frac{E_\rho(0) - E_\rho(n)}{E_\rho(0)} \tag{3.8}$$

i.e., a value in [0,1] that indicates the error reduction obtained by a human annotator who has validated the documents at the first n rank positions in the ranking generated by ρ; 0 stands for no reduction, 1 stands for total elimination of error. Note that $ER_\rho(n)$ is undefined when $E_\rho(0) = 0$, i.e., when the automatic classifier does a perfect job in classifying the test set; this is not a problem, since in this case human validation is not needed, and it makes no sense to speak of "error reduction".

Example plots of the $ER_\rho(n)$ measure are displayed in Figure 3.3, where different curves represent different ranking methods $\rho', \rho'', ...$, and where, for better convenience, the x axis indicates the fraction $n/|Te|$ of the test set that has been validated rather than the number n of validated documents. By definition all curves start at the

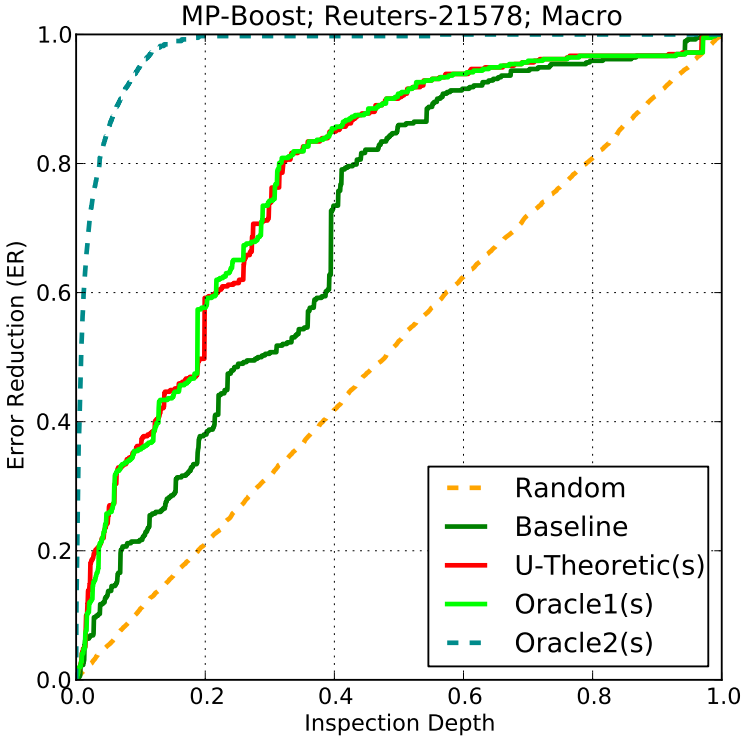


Figure 3.3: Error reduction, measured as ER_ρ^M, as a function of validation depth. The dataset is REUTERS-21578, the learner is MP-BOOST. The Random curve indicates the results of our estimation of the expected ER of the random ranker via a Monte Carlo method with 100 random trials. Higher curves are better.

origin of the axes (i.e, if the annotator validates 0 test documents, no error reduction is obtained) and end at the upper right corner of the graph (i.e., if the annotator validates all the $|Te|$ test documents, a complete elimination of error is obtained). More convex (i.e., higher) curves represent better strategies, since they indicate that a higher error reduction is achieved for the same amount of manual validation effort.

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

increase in accuracy "normalizes" our curves, i.e., allows a meaningful comparison of curves obtained on different datasets and after different classifiers have been used.

Since (as stated in Chapter 2) we use F_1 for measuring effectiveness, as a measure of classification error we use $E_1 \equiv (1 - F_1)$, which indeed (as assumed at the beginning of this section) ranges on $[0, 1]$. In order to measure the overall effectiveness of a ranking method across the entire set $\mathcal{C}$ of classes, we compute *macro-averaged* E_1 (noted E_1^M), obtained by computing the class-specific E_1 values and averaging them across the c_j's; from this it derives that $E_1^M = 1 - F_1^M$. By $ER_\rho^M(n)$ we will indicate macro-averaged $ER_\rho(n)$, also obtained by computing the class-specific $ER_\rho(n)$ values and averaging them across the c_j's.

Note that, while it might be the case that $E_\rho(0) = 0$ (and $ER_\rho(n)$ is undefined as a consequence) for *some* classes, it is highly unlikely that this is so for *all* classes. This means that there is essentially no risk that $ER_\rho^\mu(n)$ and $ER_\rho^M(n)$ might be themselves undefined.

3.4.2 Normalized error reduction at rank ...

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

3.4.3 ... and its expected value

However, $NER_\rho(n)$ is still unsatisfactory as a measure, since it depends on a specific value of n (which is undesirable, since our human annotator may decide to work down the ranked list as far as she deems suitable). Following [66] we assume that the human annotator stops validating the ranked list at exactly rank n with probability $P_s(n)$ (the index s stands for "stoppage"). We can then define the *expected normalized error reduction* of ranking method ρ on a given document set Te as the expected value of $NER_\rho(n)$ according to probability distribution $P_s(n)$, i.e.,

$$ENER_\rho = \sum_{n=1}^{|Te|} P_s(n) NER_\rho(n) \tag{3.9}$$

[4] That the expected $ER_\rho(n)$ value of the random ranker is $\frac{n}{|Te|}$ is something that we have not tried to formally prove. However, that this holds is supported by intuition *and* is unequivocally shown by Monte Carlo experiments we have run on our datasets; see Figures 3.3 to 3.6 for a graphical representation.

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

Different probability distributions $P_s(n)$ can be assumed, but in this task it is not immediately clear how to best model $P_s(n)$, since ranking for SATC is actually quite different from ranking for ad hoc search, or similar other contexts which have been studied before. This task seems a bit akin to a recall-oriented search task, since in our case it is clear that the higher the validation depth, the higher the benefit, which also seems the case in recall-oriented search tasks. But here we have to consider the fact that the validation depth a user chooses might be influenced by the initial error as estimated via k-fold cross-validation (the higher the error, the higher the validation depth is likely to be), by the desired level of effectiveness (the higher this level, the higher the validation depth is likely to be), and by the sheer size of the test set (it is conceivable to scan large portions of the test set if this is small, while it is not conceivable to do so if the test set is large)

In order to base the definition of such a distribution on a plausible model of user behaviour, we here make the assumption (along with [58]) that a human annotator, after validating a document, goes on to validate the next document with probability (or *persistence* [58]) p or stops validating with probability $(1-p)$, so that

$$P_s(n) = \begin{cases} p^{n-1}(1-p) & \text{if } n \in \{1, \ldots, |Te|-1\} \\ p^{n-1} & \text{if } n = |Te| \end{cases} \tag{3.10}$$

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

Using this distribution in practice entails the need of determining a realistic value for p. A value $p = 0$ corresponds to a situation in which the human annotator only validates the top-ranked document, while $p = 1$ indicates a human annotator who validates each document in the ranked list. Unlike in ad hoc search, we think that in a SATC context it would be unrealistic to take a value for p as given irrespective of the size of Te. In fact, given a desired level of error reduction, when $|Te|$ is large the human annotators need to be more persistent (i.e., characterized by higher p) than when $|Te|$ is small. Therefore, instead of assuming a predetermined value of p we assume a predetermined value of ξ, and derive the value of p from the equation $\xi = \frac{1}{|Te|(1-p)}$. For example, in a certain application we might assume $\xi = .20$ (i.e., assume that the average human annotator validates 20% of the test set). In this case, if $|Te| = 1000$, then $p = 1 - \frac{1}{.20 \cdot 1000} = .9950$, while if $|Te| = 10,000$, then $p = 1 - \frac{1}{.20 \cdot 10000} = .9995$. In the experiments of Section 3.5 we will test all values of p corresponding to values of ξ in $\{.05, .10, .20\}$.

Note that the values of $ENER_\rho$ are bound above by 1, but a value of 1 is not attainable. In fact, even the "perfect ranker" (i.e., the ranking method that top-ranks all misclassified documents, noted *Perf*) cannot attain an $ENER_\rho$ value of 1,

since in order to achieve total error elimination all the misclassified documents need to be validated anyway, one by one, which means that the only condition in which $ENER_{Perf}$ might equal 1 is when there is just 1 misclassified document. We do not try to normalize $ENER_\rho$ by the value of $ENER_{Perf}$ since $ENER_{Perf}$ cannot be characterized analytically, and depends on the actual labels in the test set.

3.5 Experiments

We have now fully specified (Section 3.3) a method for performing SATC-oriented ranking and (Section 3.4) a measure for evaluating the quality of the produced rankings, so we are now in a position to test the effectiveness of our proposed method. In Sections 3.5.1 to 3.5.4 we will describe our experimental setting, while in Section 3.5.5 we will report and discuss the actual results of these experiments.

3.5.1 Experimental protocol

Any dataset is partitioned into a training set Tr and a test set Te. In each experiment on SATC ranking methods we adopt the following experimental protocol:

1. For each $c_j \in \mathcal{C}$
   a) Train classifier $\hat{\Phi}_j$ on Tr and classify Te by means of $\hat{\Phi}_j$;
   b) Run k-fold cross-validation on Tr, thereby
      i. computing TP_j^{Tr}, FP_j^{Tr}, and FN_j^{Tr};
      ii. optimizing the σ parameter of Equation 3.1 (see Section 3.5.2 below for the actual optimization method used);
2. For every ranking policy ρ tested
   a) Rank Te according to ρ;
   b) Scan the ranked list from the top, correcting possible misclassifications and computing the resulting values of $ENER_\rho^M$ for different values of ξ.

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

The above protocol simulates the activity of a human annotator who, after a classifier $\hat{\Phi}$ has been generated from Tr and has been used to automatically label the documents in Te and to rank them, manually validates the top-ranked automatically labeled documents.

3.5.2 Probability calibration

We optimize the σ parameter by picking the value of σ that minimizes the average (across the $c_j \in \mathcal{C}$) absolute value of the difference between Pos_j^{Tr}, the number of positive training examples of class c_j, and $E[Pos_j^{Tr}]$, the *expected* number of such examples as resulting from the probabilities of membership in c_j computed in the

k-fold cross-validation. That is, we pool together all the training documents classified in the k-fold cross-validation phase, and then we pick

$$\begin{aligned}
&\arg\min_{\sigma} \frac{1}{|\mathcal{C}|} \sum_{c_j \in \mathcal{C}} |Pos_j^{Tr} - E[Pos_j^{Tr}]| = \\
&\arg\min_{\sigma} \frac{1}{|\mathcal{C}|} \sum_{c_j \in \mathcal{C}} |Pos_j^{Tr} - \sum_{d_i \in Tr} P(c_j|d_i)| = \\
&\arg\min_{\sigma} \frac{1}{|\mathcal{C}|} \sum_{c_j \in \mathcal{C}} |Pos_j^{Tr} - \sum_{d_i \in Tr} \frac{e^{\sigma \hat{\Phi}_j(d_i)}}{e^{\sigma \hat{\Phi}_j(d_i)} + 1}|
\end{aligned} \tag{3.11}$$

This method (which [60] attributes to [64] and calls *Platt calibration*) is a much faster calibration method than the traditional method of picking the value of σ that has performed best in k-fold cross-validation. In fact, unlike the latter, it does not depend on the ranking method ρ. Therefore, this method spares us from the need of ranking the training set several times, i.e., once for each combination of a tested value of σ and a ranking method ρ.

3.5.3 Learning algorithms

As our first learning algorithm for generating our classifiers $\hat{\Phi}_j$ we use the boosting-based learner MP-Boost [19]. In all our experiments we set the S parameter of MP-Boost (representing the number of boosting iterations) to 1000.

As the second learning algorithm we use support vector machines (SVMs). We use the implementation from the freely available LibSvm library[5], with a linear kernel and parameters at their default values.

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

3.5.4 Lower bounds and upper bounds

As the baseline for our experiments we use the confidence-based strategy discussed in Section 3.2, which corresponds to using our utility-theoretic method with both $G(fp)$ and $G(fn)$ set to 1. While this strategy has not (to the best of our knowledge) explicitly been proposed before, it seems a reasonable, common-sense strategy anyway.

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

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

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

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

In the second such method (Oracle2) we instead peek *at the true labels* of the documents in Te, which means that we will be able to (a) use the actual values of TP_j, FP_j, FN_j in the computation of $G(d_i, fp_j)$ and $G(d_i, fn_j)$ (as in Oracle1), *and* (b) replace the probabilities in Equation 3.3 with the true binary values (i.e., replacing $P(x)$ with 1 if x is true and 0 if x is false), after which we apply our utility-based ranking method as usual. The difference in effectiveness between Oracle2 and our method will be due to factors (i) and (ii) already mentioned for Oracle1 *and* to our method's (obvious) inability to perfectly predict whether a document was classified correctly or not.

3.5.5 Results and discussion

The results of our experiments are given in Tables 3.1 and 3.2, where we present the results of running, for each of two learners (MP-Boost and SVMs respectively) and five datasets (Reuters-21578, OHSUMED, and three variants of them – called Reuters-21578/10, Reuters-21578/100, OHSUMED-S – that we will introduce in Sections 3.5.5, 3.5.5, 3.5.5), our utility-theoretic method against the three methods discussed in Section 3.5.4. In Tables 3.1 and 3.2 our method, Oracle1 and Oracle2 are actually indicated as U-Theoretic(s), Oracle1(s) and Oracle2(s), to distinguish them from the "dynamic" methods that will be described in Chapter 4. Tables 3.1 and 3.2 presents $ENER_\rho^M(\xi)$ values for three representative values of ξ, i.e., 0.05, 0.10, and 0.20.

Mid-sized test sets

Figure 3.3 plots the results, in terms of $ER_\rho^M(n)$, of our experiments with the MP-Boost learner on the Reuters-21578 dataset, while 3.4 does the same for the SVM learner. The results of these experiments in terms of $ENER_\rho^M$ as a function of the chosen value of ξ are instead reported in Tables 3.1 and 3.2. The optimal value of σ returned by the k-fold cross-validation phase is .554 for MP-Boost and 7.096 for SVMs; these values, sharply different from 1 and from each other, clearly show the

		MP-Boost					
		$\xi = 0.05$		$\xi = 0.10$		$\xi = 0.20$	
Reuters-21578	Baseline	.071		.108		.152	
	U-Theoretic(s)	.163	(+128%)	.226	(+109%)	.280	(+84%)
	Oracle1(s)	.155	(+117%)	.222	(+106%)	.280	(+84%)
	Oracle2(s)	.693	(+869%)	.738	(+583%)	.707	(+365%)
R-21578/10	Baseline	.063		.097		.135	
	U-Theoretic(s)	.145	(+131%)	.203	(+110%)	.245	(+81%)
	Oracle1(s)	.159	(+153%)	.205	(+112%)	.243	(+80%)
	Oracle2(s)	.555	(+784%)	.643	(+566%)	.648	(+380%)
R-21578/100	Baseline	.069		.121		.164	
	U-Theoretic(s)	.118	(+71%)	.172	(+42%)	.215	(+31%)
	Oracle1(s)	.192	(+178%)	.247	(+104%)	.281	(+71%)
	Oracle2(s)	.429	(+521%)	.537	(+344%)	.575	(+251%)
OHSUMED	Baseline	.385		.479		.512	
	U-Theoretic(s)	.442	(+15%)	.529	(+10%)	.549	(+7%)
	Oracle1(s)	.445	(+16%)	.530	(+11%)	.549	(+7%)
	Oracle2(s)	.838	(+118%)	.839	(+75%)	.769	(+50%)
OHSUMED-S	Baseline	.021		.025		.026	
	U-Theoretic(s)	.087	(+323%)	.118	(+374%)	.132	(+402%)
	Oracle1(s)	.091	(+343%)	.117	(+370%)	.125	(+375%)
	Oracle2(s)	.481	(+2246%)	.554	(+2125%)	.572	(+2075%)

Table 3.1: Results of various ranking methods, applied to MP-Boost and several test collections, in terms of $ENER_\rho^M(\xi)$, for $\xi \in \{0.05, 0.10, 0.20\}$. Improvements listed for the various methods are relative to the baseline. Reuters-21578/10 and Reuters-21578/100 are respectively named as R-21578/10 and R-21578/100

advantage of converting confidence scores into probabilities via a *generalized* logistic function.

The first insight we can draw from these results is that our U-Theoretic(s) method outperforms Baseline in a very substantial way. This can be appreciated both from the plots of Figures 3.3 and 3.4, in which the red curve (corresponding to U-Theoretic(s)) is markedly higher than the green curve (corresponding to Baseline), and from Tables 3.1 and 3.2. In this latter, for $\xi = .10$ (corresponding to $p = .996$) our method obtains relative improvements over Baseline of +109% (MP-Boost) and +51% (SVMs); for

		SVMs		
		$\xi = 0.05$	$\xi = 0.10$	$\xi = 0.20$
REUTERS-21578	Baseline	.262	.352	.420
	U-Theoretic(s)	.442 (+69%)	.531 (+51%)	.562 (+34%)
	Oracle1(s)	.477 (+82%)	.563 (+60%)	.586 (+40%)
	Oracle2(s)	.719 (+174%)	.760 (+116%)	.723 (+72%)
R-21578/10	Baseline	.243	.322	.383
	U-Theoretic(s)	.330 (+36%)	.415 (+29%)	.465 (+21%)
	Oracle1(s)	.392 (+61%)	.482 (+50%)	.522 (+36%)
	Oracle2(s)	.596 (+145%)	.676 (+110%)	.672 (+75%)
R-21578/100	Baseline	.226	.302	.364
	U-Theoretic(s)	.291 (+29%)	.365 (+21%)	.416 (+14%)
	Oracle1(s)	.318 (+41%)	.422 (+40%)	.479 (+32%)
	Oracle2(s)	.458 (+103%)	.568 (+88%)	.600 (+65%)
OHSUMED	Baseline	.526	.630	.644
	U-Theoretic(s)	.623 (+18%)	.685 (+9%)	.666 (+3%)
	Oracle1(s)	.639 (+21%)	.687 (+9%)	.657 (+2%)
	Oracle2(s)	.864 (+64%)	.854 (+36%)	.778 (+21%)
OHSUMED-S	Baseline	.075	.124	.164
	U-Theoretic(s)	.212 (+184%)	.282 (+127%)	.323 (+97%)
	Oracle1(s)	.272 (+265%)	.334 (+169%)	.352 (+115%)
	Oracle2(s)	.511 (+585%)	.589 (+375%)	.603 (+268%)

Table 3.2: Results of various ranking methods, applied to SVMs and several test collections, in terms of $ENER_\rho^M(\xi)$, for $\xi \in \{0.05, 0.10, 0.20\}$. Improvements listed for the various methods are relative to the baseline. REUTERS-21578/10 and REUTERS-21578/100 are respectively named as R-21578/10 and R-21578/100

$\xi = .20$ the improvements, while not as high as for $\xi = .10$, are still sizeable (+84% for MP-BOOST and +34% for SVMs), while for $\xi = .05$ the improvements are even higher than for $\xi = .10$ (+128% for MP-BOOST and +69% for SVMs).

A second insight is that, surprisingly, our method hardly differs in terms of performance from Oracle1(s). The two curves can be barely distinguished in Figure 3.3, and in terms of $ENER_\rho^M$ Oracle1(s) is even slightly outperformed, in the MP-BOOST experiments, by U-Theoretic(s) (e.g., .226 vs. .222 for $\xi = .10$). This shows that (at least judging from these experiments) Laplace smoothing is nearly optimal, and there is

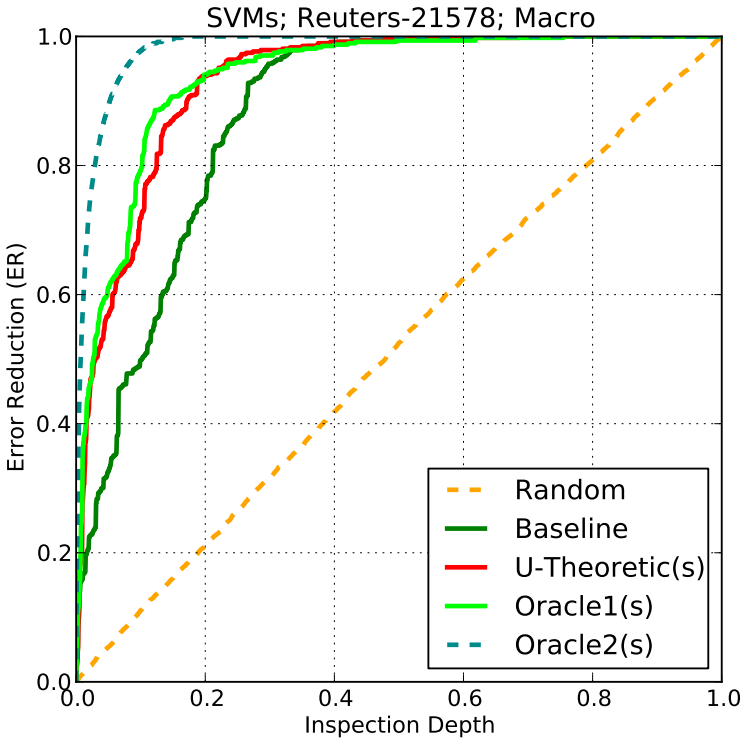


Figure 3.4: Same as Figure 3.3, but with the SVM learner in place of MP-Boost.

likely not much we can gain from applying alternative, more sophisticated smoothing methods. This is sharply different from what happens in language modelling, where Laplace smoothing has been shown to be an underperformer [27]. The fact that with MP-Boost our method slightly (and strangely) outperforms Oracle1(s) is probably due to accidental, "serendipitous" interactions between the probability estimation component (Equation 3.1) and the contingency cell estimation component of Section 3.3.3.

A third interesting fact is that error reduction is markedly better in the SVM experiments than in the MP-Boost experiments. This is evident from the fact that the Figure 3.3 curves for SVMs are much more convex (i.e., are higher) and are closer to the optimum (i.e., closer to the Oracle2(s) curve) than the corresponding Figure 3.3 curves for MP-Boost. This fact is also evident from the numerical results reported in Tables 3.1 and 3.2 where, with U-Theoretic(s), SVMs obtain $ENER_{\rho}^{M}(.10) = .531$, which is +134% better than the $ENER_{\rho}^{M}(.10) = .226$ result obtained by MP-Boost (similar improvements can be observed for the other methods and for the other values of ξ). This provides a striking contrast with the *classification accuracy* results reported in Table 3.3 where, on the same dataset, MP-Boost ($E_1^M = .392$) substantially

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

Table 3.3: Characteristics of the test collections used. From left to right we report the number of test sets (Column 2) and, for each test set, the number of training documents (3), the number of test documents (4), the number of classes (5), and the average number of classes per test document (6). Columns 7-10 report the initial error (both E_1^M and E_1^μ) generated by the MP-BOOST and SVMs classifiers.

outperformed SVMs ($E_1^M = .473$). It is easy to conjecture that, while MP-BOOST yields higher classification accuracy, it generates less reliable (calibrated) confidence scores, i.e., it generates confidence scores that correlate with the ground truth worse than the SVM-generated scores.

The improvements of U-Theoretic(s) over the baseline are instead much higher for MP-BOOST than for SVMs (e.g., for $\xi = .10$ these are +109% and +51%, respectively). (The same goes for the improvements of Oracle1(s) over the baseline.) This is likely due to the fact that, as observed above, the absolute values of $ENER_\rho^M(\xi)$ obtained by the baseline are much higher for SVMs than for MP-BOOST for all methods, so the margins of improvement with respect to the baseline are smaller for SVMs than for MP-BOOST.

As a final note we remark that, while in a standard text classification context REUTERS-21578 would probably qualify as a small dataset, it probably qualifies at least as a mid-sized one in a semi-automatic text classification context. The reason is that, in this context, we need to think of the test set not just as a set of unlabeled documents the classifier is being tested on (the bigger this set, the more reliable the conclusions that can be drawn from it), but as a set of automatically classified documents that a human editor partially validates. Drastically larger test sets would inevitably raise the question whether they *realistically* simulate an operational SATC context, i.e., whether for unlabeled sets of that size it makes sense at all, in an operational condition, to attempt improving the overall quality of the classification via human validation. E.g., in the well-known RCV1-v2 dataset [49] the test set consists of 781,265 documents, and even validating 10% of it (which might seem seem a sensible percentage if we want to improve accuracy to some degree) means validating 78,126 documents, which requires *a lot* of manpower.

Small test sets

We have also run a batch of experiments aimed at assessing how the methods fare when ranking test sets much smaller than REUTERS-21578. This may be *more* challenging than ranking larger sets since, when the test set is small, Laplace smoothing (i) can seriously perturb the relative proportions among the cell counts, which can generate poor estimates of $G(d_i, fp_j)$ and $G(d_i, fn_j)$, and (ii) is performed for more classes, since (as discussed at the end of Section 3.3.3) we smooth "on demand" only, and since the likelihood that $\hat{TP}_j^{ML}$, $\hat{FP}_j^{ML}$, $\hat{FN}_j^{ML}$ are smaller than 1 is higher with small test sets. This is also a realistic setting since, if a set of unlabeled documents is small, it is likely that validating a portion of it that can lead to sizeable enough effectiveness improvements is feasible from an economic point of view.

Rather than choosing a completely different dataset, we generate 10 new test sets by randomly splitting the REUTERS-21578 test set in 10 equally-sized parts (about 330 documents each), details are given in Table 3.3. In our experiments we run each ranking method on each such part individually and average the results across the 10 parts. We call this experimental scenario REUTERS-21578/10. This allows us to study the effects of test set size on our methods in a more controlled way than if we had picked a completely different dataset, since test set size is the only difference with respect to the previous REUTERS-21578 experiments.

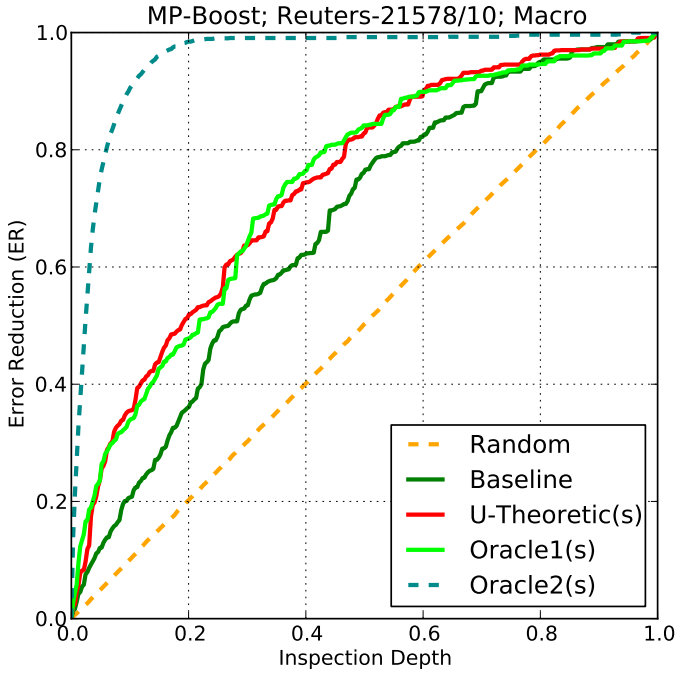
The results displayed in Figure 3.5 allow us to visually appreciate that U-Theoretic(s) substantially outperforms Baseline also in this context. This can be seen also from Tables 3.1 and 3.2: for $\xi = .10$ the relative improvement over Baseline is +110% for MP-BOOST and +30% for SVMs, and similarly substantial improvements are obtained for the two other values of ξ tested.

Incidentally, note that the REUTERS-21578/10 experiments model an application scenario in which a set of automatically labelled documents is split (e.g., to achieve faster throughput) among 10 human annotators, each one entrusted with validating a part of the set. In this case, each annotator is presented with a ranking of her own document subset, and works exclusively on it[6].

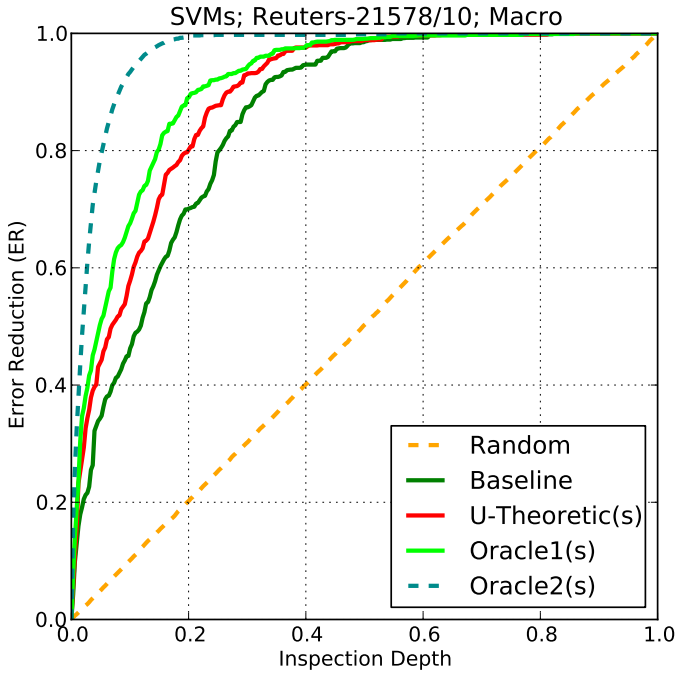
Tiny test sets

In further experiments that we have run, we have split the REUTERS-21578 test set even further, i.e., into 100 equally-sized parts of about 33 documents each (see Table 3.3 for dataset details), so as to test the performance of Laplace smoothing methods

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



(a) MP-Boost results.



(b) SVMs results.

Figure 3.5: Results obtained by (i) splitting the Reuters-21578 test set into 10 random, equally-sized parts, (ii) running the analogous experiments of Figure 3.3 independently on each part, and (iii) averaging the results across the 10 parts.

in even more challenging conditions. We call this experimental scenario REUTERS-21578/100. From an application point of view this is a less interesting scenario than the two previously discussed ones, since applying a ranking method to a set of 33 documents only is of debatable utility, given that a human annotator confronted with the task of validating just 33 documents can arguably check them all without any need for ranking. The goal of these experiments is thus checking whether our method can perform well even in extreme, albeit scarcely realistic, conditions.

The detailed $ER_\rho^M(n)$ plots for this REUTERS-21578/100 scenario are presented in Figure 3.6, while the $ENER_\rho^M$ results are reported in Tables 3.1 and 3.2. U-Theoretic(s) still outperforms Baseline, with a relative improvement of +42% with MP-BOOST and +21% with SVMs with $\xi = .10$, corresponding to $p = .696$; qualitatively similar improvements are obtained with the other tested values of ξ.

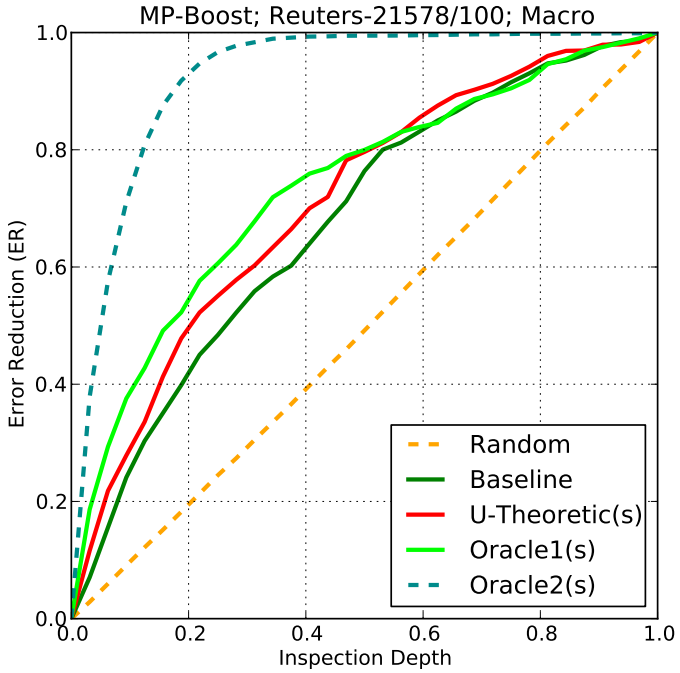
Note that in these experiments, unlike in those performed on the full REUTERS-21578, the Oracle1 method proves to be markedly superior to U-Theoretic(s) (e.g., .247 vs. .172 in terms of $ENER_\rho^M(.10)$ with MP-BOOST, and similarly for other values of ξ and for the SVM learner). The obvious reason is that, for a smaller test set, (a) distribution drift is higher, (b) "smoothing on demand" is invoked more frequently (because the likelihood that contingency table cells have a value ≤ 1 is higher), and (c) when smoothing is indeed applied the distribution across the cells of the contingency table is perturbed more strongly.

Note also that the $ER_\rho^M(n)$ curves are smoother than the analogous curves for the full REUTERS-21578 and, although to a lesser extent, those for REUTERS-21578/10. This is due to the fact that the curves in Figure 3.6 result from averages across 100 different experiments, and the increase brought about at rank n is actually the average of the increases brought about at rank n in the 100 experiments.

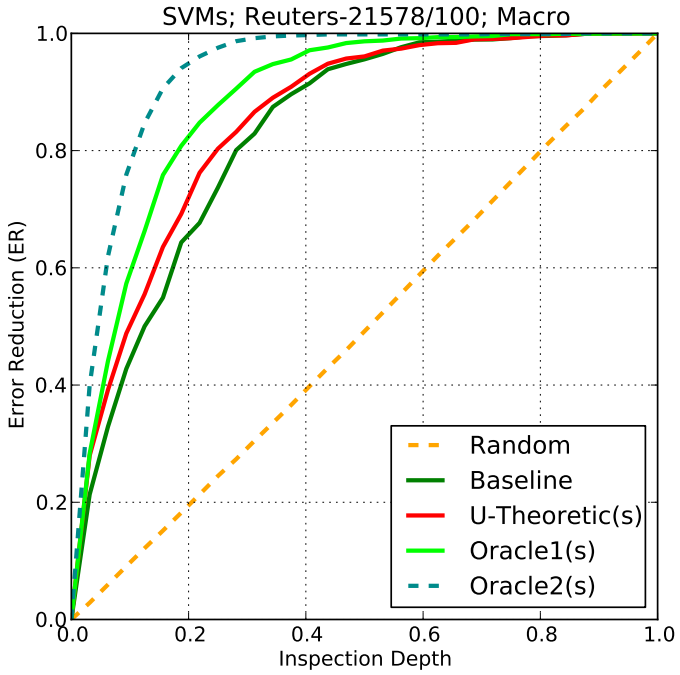
Large test sets

While in the previous sections we have discussed experiments on mid-sized to small (or very small) datasets, we now look at larger datasets such as OHSUMED. The OHSUMED results in Tables 3.1 and 3.2 confirm the quality of U-Theoretic(s), which outperforms the purely confidence-based baseline by +10% (MP-BOOST) and +9% (SVMs) in terms of $ENER_\rho^M(.10)$; qualitatively similar improvements are obtained for the other two values of ξ studied.

The OHSUMED collection is characterized by the presence of an unusually large number (93.1% of the entire lot) of unlabeled documents (i.e., documents, that are negative examples for all $c_j \in \mathcal{C}$) that originally belonged to other subtrees of the MeSH tree. Since such a large percentage is innatural, we have generated (and also used in our experiments) a variant of OHSUMED (called OHSUMED-S) by removing all the unlabeled documents from both the training set and the test set; the datasets are illustrated in Table 3.3.



(a) MP-Boost results.



(b) SVMs results.

Figure 3.6: Same as Figure 3.5 but with Reuters-21578/100 in place of Reuters-21578/10.

As illustrated in Table 3.1 and 3.2, on OHSUMED-S U-Theoretic(s) outperforms the confidence-based baseline by a very large margin (+374% with MP-BOOST and +127% with SVMs for $\xi = .10$, with qualitatively similar results for the other two tested values of ξ).

Discussion

In sum, the results discussed from Section 3.5.5 to the present one have unequivocally shown that U-Theoretic(s) outperforms the confidence-based baseline, usually by a large or very large margin, for all the five tested datasets and for both the two tested learners.

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

3.6 Further readings

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

A standard response to this problem is to adopt *active learning* (AL – see e.g., [32, 77]), i.e., use algorithms that optimize the informativeness of additional training examples provided by a human annotator. Still, providing additional training examples, no matter how carefully chosen, may be insufficient, since in many applicative contexts high enough accuracy levels cannot be attained, irrespectively of the quantity and quality of the available training data. Similar considerations apply when active learning is carried out at the term level, rather than at the document level [28, 65]. In Chapter 5 we will investigate on the differences and similarities between SATC and AL.

A related response to the same problem is to adopt *training data cleaning* (TDC – see e.g., [22, 25, 59]), i.e., use algorithms that optimize the human annotator's efforts at correcting possible labelling mistakes in the training set. Similarly to the case of AL, in many applicative contexts high enough accuracy levels cannot be attained even at the price of carefully validating the entire training set for labelling mistakes. TDC is usually employed by ranking mislabeled examples according to the confidence of the classifiers applied on Tr; other strategies analyze anomalies in the distribution of the training samples, or exploit information specific to the chosen learners [22].

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

Few published works deal with the task of SATC. Recently a paper has been published [56], where the authors study a family of methods for exploiting the "document certainty", taking into account the confidence scores computed by the classifiers. They compare their methods on several learners and datasets, using F_1^μ as evaluation measure; the U-Theoretic(s) method is compared on the REUTERS-21578 collection, results are evaluated with the $ENER_\rho$ measure. The authors claim to obtain a slight improvement over U-Theoretic(s), using their best method, but the methodology of comparison is not licit. They use a SVM learner, and compare its results with our results with MP-BOOST, so both the initial accuracy on the test set, the estimated contingency table and the misclassification probability estimations, are different. If we instead compare our results obtained with the SVM learner, our initial classification accuracy is slightly better, and the improvements in terms of $ENER_\rho$ are substantial. Furthermore they compare the macro-averaged rankings with micro evaluation, while in Chapter 4 we present a new class of micro-averaged methods, specific for micro evaluation.

Bianchi et al. [7] apply semi-automated classification to the case in which a desired level of accuracy needs to be reached as stated in a Service Level Agreement between suppliers and customers. The authors propose a method for selecting documents to be validated, based on ranges of confidence. Their approach is exclusively based on probability values of the confidence of classification, but the method for calibrating probabilities is not discussed. They argue that SVMs return best confidence estimates in order to select the documents which, once validated, bring about the best improvement in precision.

While AL (and, to a much lesser degree, TDC) have been investigated extensively in a TC context, semi-automated TC has been almost neglected by the research community. While a number of papers (e.g., [43, 73, 83]) have evoked the existence of this scenario, we are aware of only these two published papers, which either discuss ranking policies for supporting the human annotator's effort, or that attempt to quantify the effort needed for reaching a desired level of accuracy. For instance, while discussing a system for the automatic assignment of ICD9 classes to patients' discharge summaries, Larkey and Croft [43] say "We envision these classifiers being used in an interactive system which would display the 20 or so top ranking [classes] and their scores to an expert user. The user could choose among these candidates (...)", but do not present experiments that quantify the accuracy that the validation activity brings about, or methods aimed at optimizing the cost-effectiveness of this activity.

3.6.1 Probability estimates of classification

The problem of returning probability outputs from classifiers has been tackled with some different approaches in ML. The function for transforming confidence scores into probabilities is usually achieved by cross-validation on the training set, so the effectiveness of the function is determined by the goodness of the estimation of its parameters.

Our calibration method is directly inspired to the Platt calibration [64], which motivates the choice of the function by empirical evaluations on SVMs outputs, which follow a sigmoidal distribution. Niculescu et al. [60, 61] study three calibration methods applied to popular ML algorithms. They principally show how boosting algorithms with full decision trees can be calibrated in order to produce optimum probabilistic outputs. They examine Platt calibration along with other two methods; the experimental study exploits the effectiveness of the Platt's method, especially against a method based on *isotonic regression*. The calibration based on isotonic regression [85, 86] creates a function which splits the range of classifier's outputs on the training set, into subranges; this operation is performed on the training set, each subrange is mapped into a value in $[0, 1]$, so that the probability outputs follow the distribution of the classifier outputs.

An interesting work in [6] explores calibration methods which convert confidence scores into asymmetric distributions. The authors analyze the asymmetric versions of two distributions: Laplace and Gaussian. They apply methods for finding the maximum likelihood estimates of the distribution parameters, and they compute the distributions on each class, distinguishing the ones relative to positive prediction confidences from the negative ones. The empirical evaluation of the methods indicate an effectiveness comparable the calibrations based on logistic functions. The use of logistic functions for classification is developed in [47]. In this work a similar approach to probability calibration is used in order to create a classifier, so the outputs returned by the calibrated function are used to predict class assignments; the learning process is thus performed through an estimator on the training set, comparable to the parametric calibration method of Platt.

3.6.2 Evaluating rankings by modelling user behavior

In information retrieval the evaluation is performed on an ordered list of documents, which are presented to the user, ranked by expected relevance. Several approaches to the evaluation of retrieval effectiveness have been proposed in literature; some approaches take into account the user searching behaviour, in fact the relevance of a document is potentially affected by the user actions and satisfaction (e.g., opening only the first result, reading the first page of results, etc.).

With the $ENER_\rho$ measure we model the "patience" of the annotator as with the *rank-biased precision* (RBP) [58]. Other popular measures are *discontinued cumulative gain* (DCG) [35] or *expected reciprocal rank* (ERR) [13]. DCG has been one of the first

measure in IR which weights the relevance of a document by a function of its rank. ERR considers, for each document, the relevance of the documents at the previous ranks; a user is less likely to continue scanning the ranking if the documents she finds satisfy her "information need".

These evaluation measures in IR can be supported by the information of real user usage of the retrieval systems, for example page clicks, or query logs, in web search. In [12] the authors study a selection of values for the RBP free parameter, which is the already mentioned persistence p. They generate a distribution of values for p, given a distribution of user behaviours generated from "click data"; in this way it is possible to adapt the evaluation of an IR system to observed user actions. In [84] a new measure is developed, namely *expected browsing utility*, which is specific for web search; it integrates parameters which can be easily trained on web usage data.

In Chapter 7 we will discuss some intuitions about extending our evaluation metrics in order to asses the human validation process.

3.6.3 Human interaction in learning systems

Several applications for the automatic analysis of textual documents can require human interaction, in order to improve their effectiveness (for example e-discovery, described in the Introduction). A well known scenario is *relevance feedback* in information retrieval [54, 68], where the user marks the documents returned by a retrieval system, in order to enrich her information need and formulate a new and more accurate search. In [38] a modern approach to relevance feedback is discussed, which recalls our principle of maximizing a measure of utility. The authors develop a ML algorithm for the diversification of the results, in order to optimize the utility of the results in terms of user satisfaction. On the retrieved documents, the system balances values of relevance (for meeting the user needs) and informativeness (for improving the model with feedback).

Crowdsourcing applications are widely used from both the scientific community and the industry (e.g., Amazon Mechanical Turk [7]) in order to obtain labelled data and building robust datasets. The problem of combining the collaborations of several annotator has been studied with the purpose of understanding the quality of the human work [3, 34, 39]. In SATC we assume that the validation is correct, discarding problematics as inter-agreement or expertise of the annotators. The human cost is one critical aspect in crowdsourcing, in our methods we have assumed that the cost is linearly proportional with the number of documents to be validated.

In machine learning different scenarios exist, in which automatic processes and human interactions are combined; they are not limited to active learning or training data cleaning. The following are some examples of applications.

A software for form filling is presented in [41]. The described system allows the users to fill forms (e.g. web forms), supporting this operation with automatic filling

[7] `http://www.mturk.com`

of the empty form fields, employing *constrained conditional random fields*. The effectiveness of the system is measured in terms of expected user actions, so the better solution is the one which minimizes the user writings, and the corrections of wrong information. The software can also highlight the form fields which it has automatic filled, and on which is less confident, thus reducing the ratio between user actions and filling accuracy.

In [26], a particular software architecture for building training data is described, it produces a sort of automatic annotation of the unlabelled data. The goal of the application is learning from a training set with few positive samples and many unlabelled samples, so to reduce the human effort in annotating documents. After sampling potential negative samples, a method for selecting samples to be corrected is applied, and an AL strategy is used in order to assign the correct classes. Uncertain samples are extracted through the recomputation of a separation margin, learned with SVMs.

Weeber et al. [81] examine the assessor disagreement in annotating training and test data. They evaluate different cases, and for each case they compare the accuracy of classification between different scenarios of annotation, e.g.: different or same assessors for training and test set, two groups of assessors for training set, etc. One interesting outcome of this study is in the measurement of the human effort necessary for the annotation, when a specific level of recall is requested, but different assessors are used. The problem is approached by ranking documents according to the probabilistic outputs of a classifier.

Coactive learning [75] is a model of interaction between a learning system and its user. The authors start from the conjecture that the user feedback gives an improvement of the prediction but not necessarily the optimal. The system integrates a measure of the utility of the results, the goal of the learning process is minimizing the difference between the utility of the results annotated after user feedback and the utility of the true labels. Given the assumption that human annotation may not imply total improvement of accuracy, the algorithms learn from the annotated data in order to maximize the utility of the user feedbacks of successive annotations.

3.7 Conclusions

We have presented a method for ranking the documents labelled by an automatic classifier. The documents are ranked in such a way as to maximize the expected reduction in classification error brought about by a human annotator who validates a subset of the ranked list and corrects the labels when appropriate. First we have introduced a probabilistic approach to the task of semi-automated TC, then we have defined our method, based on the concept of utility in validating a ranked document.

We have also proposed an evaluation measure for such ranking method, based on the notion of expected normalized error reduction. We have introduced this measure in three steps, starting from the concept of error reduction, then defining a normalized error reduction, finally formalizing the expected error reduction. This measure is

designed with the aim of modelling the behaviour of the user who validates the ranked documents.

We have pointed out the details of the implementation and the experimental setting. Experiments carried out on standard datasets show that our method substantially outperforms a state-of-the-art baseline method.

It should be remarked that the very fact of using a utility function, i.e., a function in which different events are characterized by different gains, makes sense here since we have adopted an evaluation function, such as F_1, in which correcting a false positive or a false negative brings about different benefits to the final effectiveness score. If we instead adopted standard *accuracy* (i.e., the percentage of binary classification decisions that are correct) as the evaluation measure, utility would default to the probability of misclassification, and our method would coincide with the baseline, since correcting a false positive or a false negative would bring about the same benefit. The methods we have presented are justified by the fact that, in text classification and in other classification contexts in which imbalance is the rule, F_1 is the standard evaluation function, while standard accuracy is a deprecated measure because of its lack of robustness to class imbalance (see e.g., [73, Section 7.1.2] for a discussion of this point).

In the next chapters we will extend the method exploring new intuitions about the ranking function and its applications. A task of SATC can be investigated through its different dimensions, we will follow some of the most interesting ramifications of the problem.

4

Additional Ranking Methods for Semi-Automated Text Classification

In this Chapter we extend the development of utility-theoretic ranking methods for SATC. In Chapter 3 we defined a SATC method that has a "static" nature, i.e., utility gains are computed only once at the beginning of the SATC process.. We now come up with new insights, expanding our work in two directions: (a) we present a new "dynamic" ranking method, in which the gains are computed after each step of the manual validation, with the aim of obtaining a more accurate estimate of the improvement in accuracy, iteratively (Section 4.1); (b) we switch to another way of evaluating the classification accuracy, we reformulate our ranking methods for the micro-averaged effectiveness (Section 4.2).

We present the two ranking methods based on the already discussed concepts of utility and gain, we show how these functions can be modified in order to meet our needs. For each method we reproduce the experimental protocol of Chapter 3, expressly modified for our purposes, and we discuss the effectiveness of the new SATC methods.

4.1 An improved, "dynamic" ranking function

The utility-theoretic method discussed in Section 3.3.1 is reasonable but, in principle, suboptimal, and its suboptimality derives from its "static" nature. To see this, assume that the system has ranked the test documents according to the strategy above, that the human annotator has started from the top of the list and validated the labels of document d_i, that she has found out that its label assignment for class c_j is a false negative, and that she has corrected it, thus bringing about an increase in F_1 equivalent to

$$\frac{2(TP_j+1)}{2(TP_j+1)+FP_j+(FN_j-1)} - \frac{2TP_j}{2TP_j+FP_j+FN_j} \tag{4.1}$$

Following this correction, the value of FN_j is decreased by 1 and the value of TP_j is increased by 1. This means that, when another false negative for c_j is found and

corrected, the value of (4.1) has changed, i.e., the increase in F_1 that derives from correcting a false negative is not the same any more.

> Suppose that, before the validation process begins, $TP_j = 5$, $FP_j = 2$, $FN_j = 3$, $TN_j = 20$, which means that $F_1 = \frac{2 \cdot 5}{2 \cdot 5 + 2 + 3} = 0.666$. Suppose the first document that gets validated is a false negative; F_1 then becomes $F_1 = \frac{2 \cdot 6}{2 \cdot 6 + 2 + 2} = 0.750$, with an improvement of $(0.750 - 0.666) = 0.084$. Suppose the second document to be validated is also a false negative; F_1 then becomes $F_1 = \frac{2 \cdot 7}{2 \cdot 7 + 2 + 1} = 0.823$, which means that the improvement is now only $(0.823 - 0.750) = 0.073$. □

This example shows that the improvement in F_1 due to the validation of a false negative is not constant through the validation process. Of course, similar considerations apply for false positives. This fact is the reason why in Equation 3.6 we had defined $G(d_i, fn_j)$ as the *average* improvement in F_1 deriving from the correction of a false negative, where the average is taken across all the false negatives in the contingency table. This means that, at any specific point in the validation process, validation gains as defined in Equations 3.5 and 3.6 are an *imprecise* measure of the improvement in F_1 deriving from doing validation *at that precise point*.

This suggests redefining the validation gains defined in Equations 3.5 and 3.6 as

$$\begin{aligned} G(d_i, fp_j) &= \frac{2TP_j}{2TP_j + (FP_j - 1) + FN_j} - \frac{2TP_j}{2TP_j + FP_j + FN_j} \\ G(d_i, fn_j) &= \frac{2(TP_j + 1)}{2(TP_j + 1) + FP_j + (FN_j - 1)} - \frac{2TP_j}{2TP_j + FP_j + FN_j} \end{aligned} \tag{4.2}$$

To see the novelty introduced with respect to Equation 4.2, in the following we will discuss the case of false negatives; the case of false positives is completely analogous. The difference between Equation 3.6 and Equation 4.2 is that the former equates $G(d_i, fn_j)$ with the increase in $F_1(\hat{\Phi}_j(Te))$ that would derive by correcting *all of the documents in* FN_j *divided by their number*, while the latter equates $G(d_i, fn_j)$ with the increase in $F_1(\hat{\Phi}_j(Te))$ that would derive by correcting *the next document in* FN_j. In other words, we might say that Equation 3.6 enforces the notion of *average gain*, while Equation 4.2 enforces the notion of *pointwise gain*. The two versions return different values of $G(d_i, fn_j)$: as the following example shows, it is immediate to verify that if FN_j contains more than one document, the validation gains $G(d_i, fn_j)$ that derive by correcting different documents are the same (by definition) if we use Equation 3.6 but are not the same if we use Equation 4.2.

> Suppose we have classified a set of 100 documents according to class c_j, and that the classification is such that $TP_j = 10$, $FN_j = 20$, $FP_j = 30$, and $TN_j = 40$. According to Equation 3.6, $G(d_i, fn_j)$ evaluates to ≈ 0.0190 for each false negative corrected. Instead, according to Equation 4.2, $G(d_i, fn_j)$ evaluates to ≈ 0.0241 for the 1st false negative corrected, ≈ 0.0235 for the 2nd, ≈ 0.0228 for the 3rd, ..., down to ≈ 0.0147 for the 20th. □

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

1. For all classes $c_j \in \mathcal{C}$, compute $G(d_i, fp_j)$ and/or $G(d_i, fn_j)$ using Equations 4.2;
2. If the human annotator does not want to stop validating documents, then identify the document $d_{max} \equiv \arg\max_{d_i \in Te} U(d_i, \Omega)$ for which total utility is maximised, where $U(d_i, \Omega)$ is as in Equation 3.7, and where Equations 4.2 are used in its computation in place of Equations 3.5 and 3.6;
3. Remove d_{max} from Te;
4. For all $c_j \in \mathcal{C}$, have the human annotator check the label attached by $\hat{\Phi}_j$ to d_{max}; if all these labels are correct go to Step 2; else, for all classes $c_j \in \mathcal{C}$ for which the label attached by $\hat{\Phi}_j$ to d_{max} is incorrect:
   a) Have the human annotator correct the label;
   b) If d_{max} was a false positive for c_j, decrease $\hat{FP}_j^{La}$ by 1; if it was a false negative for c_j, increase $\hat{TP}_j^{La}$ by 1 and decrease $\hat{FN}_j^{La}$ by 1;
   c) Re-smooth $\hat{TP}_j^{La}$, $\hat{FP}_j^{La}$, $\hat{FN}_j^{La}$ if needed;
   d) Recompute $G(d_i, fp_j)$ and/or $G(d_i, fn_j)$ and go back to Step 2.

This might also be dubbed an *incremental* ranking strategy, in the sense pioneered in [1] for relevance feedback in ad-hoc search, in the sense that the values of $G(d_i, fp_j)$ and $G(d_i, fn_j)$ are incrementally updated so that the $U(d_i, \Omega)$ function reflects the fact that part of Te has indeed been corrected. In keeping with [8] we prefer to call it a *dynamic* strategy, and to call the one of Section 3.3.1 a *static* one.

Note that in Step 2 we simply compute the maximum element (according to $U(d_i, \Omega)$) of Te instead of sorting the entire set, since we can perform this step in $O(|Te|)$ instead of $O(|Te| \log |Te|)$[1]. Furthermore, note that in this algorithm the re-computation of $U_j(d_i, \Omega)$ does *not* entail the recomputation of the probabilities $P(fp_j)$ and/or $P(fn_j)$ of Equation 3.3, since these probabilities are computed (i.e., calibrated) once for all, immediately after the training phase.

Note also that computing validation gains via Equations 3.5 and 3.6 is the only possibility within the static method (since the values of $G(d_i, fp_j)$ and $G(d_i, fn_j)$ produced must be used unchanged throughout the process), but is clearly inadequate

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

in a dynamic context, in which validation gains are always supposed to be up-to-date reflections of the current situation.

The dynamic nature of this method makes it clear why, as specified at the end of Section 3.3.3, we smooth the cell count estimates only "on demand" (see also Step 4c of the above algorithm), i.e., only if any of $\hat{TP}_j^{ML}$, $\hat{FP}_j^{ML}$, $\hat{FN}_j^{ML}$ is < 1. To see this, suppose that we smooth $\hat{TP}_j^{ML}$, $\hat{FP}_j^{ML}$, $\hat{FN}_j^{ML}$ at each iteration, even when not strictly needed. Adding a count of one to each of them at each iteration means that, after k iterations, k counts have been added to each of them; this means that, after many iterations, the counts added to the cells have completely disrupted the relative proportions among the cells that result from the maximum-likelihood estimation. This would likely make the dynamic method underperform the static method, which does not suffer from this problem since the maximum-likelihood estimates are smoothed only once. As a result, we smooth a contingency table only when strictly needed, i.e., when one of $\hat{TP}_j^{ML}$, $\hat{FP}_j^{ML}$, $\hat{FN}_j^{ML}$ is < 1.

By solving the inequality $G(d_i, fn_j) > G(d_i, fp_j)$ we may find out under which conditions correcting a false negative yields a higher gain than correcting a false positive. It turns out that, when validation gains are defined according to Equation 4.2, $G(d_i, fn_j) > G(d_i, fp_j)$ whenever $FN + FP > 1$, i.e., practically always. Of course, this need not be the case for evaluation functions different from F_1, and in particular for instances of F_β with $\beta \neq 1$.

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

4.1.1 Experiments

The results of the experiments with the dynamic version of our utility-theoretic method and of our two oracle-based methods are reported in Figures 4.1 to 4.4 and in Tables 4.1 and 4.2, where they are indicated as U-Theoretic(d), Oracle1(d) and Oracle2(d). Of course there exists no dynamic version of the baseline method, since this latter does not involve validation gains.

The first observation that can be drawn from these results is the fact that U-Theoretic(d) is not superior to U-Theoretic(s), as could instead have been expected: in fact, in Figures 4.1 to 4.4 the curves corresponding to the former are barely distinguishable from those corresponding to the latter, and the numeric results reported in Tables 4.1 and 4.2 show no substantial difference either. Note that there are extremely small differences also between Oracle1(s) and Oracle1(d); this shows that the

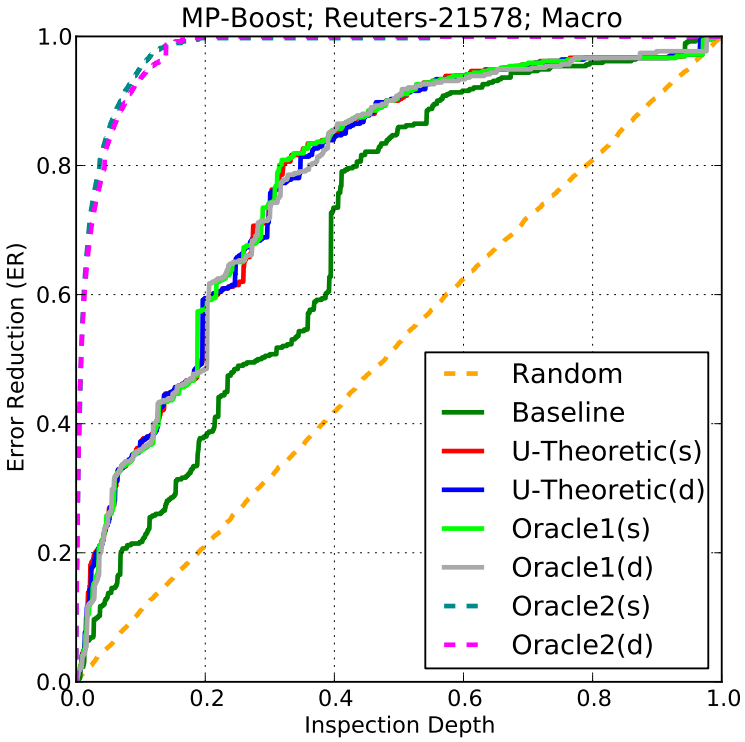


Figure 4.1: Error reduction, measured as ER_ρ^M, as a function of validation depth. The dataset is REUTERS-21578, the learner is MP-BOOST. The Random curve indicates the results of our estimation of the expected ER of the random ranker via a Monte Carlo method with 100 random trials. Higher curves are better.

lack of any substantial difference between static and dynamic is not due to a possible suboptimality of the method for estimating contingency table cells (including the method adopted for smoothing the estimates). Analogously, note also the extremely small differences between Oracle2(s) and Oracle2(d), which indicates that the culprit is not the method for estimating the probabilities of misclassification.

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

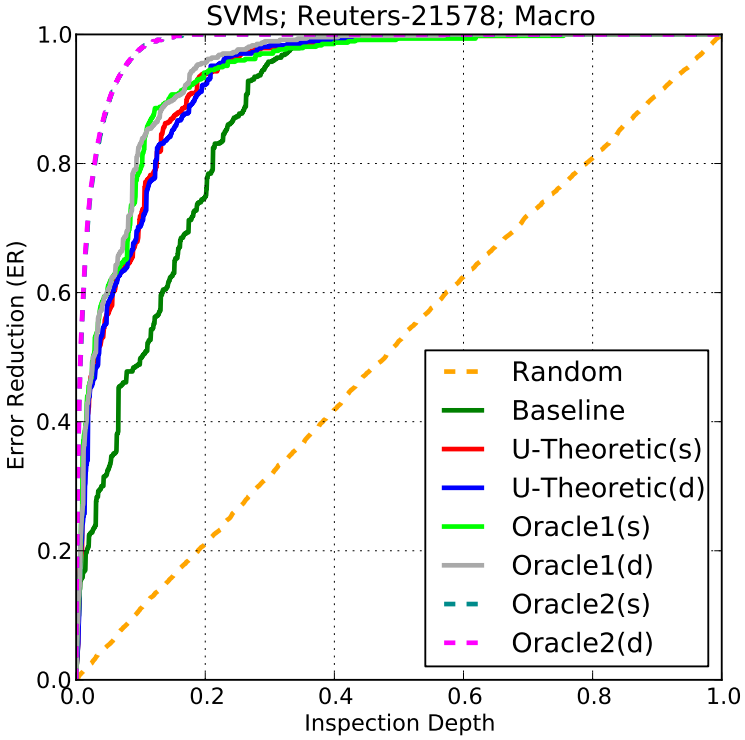
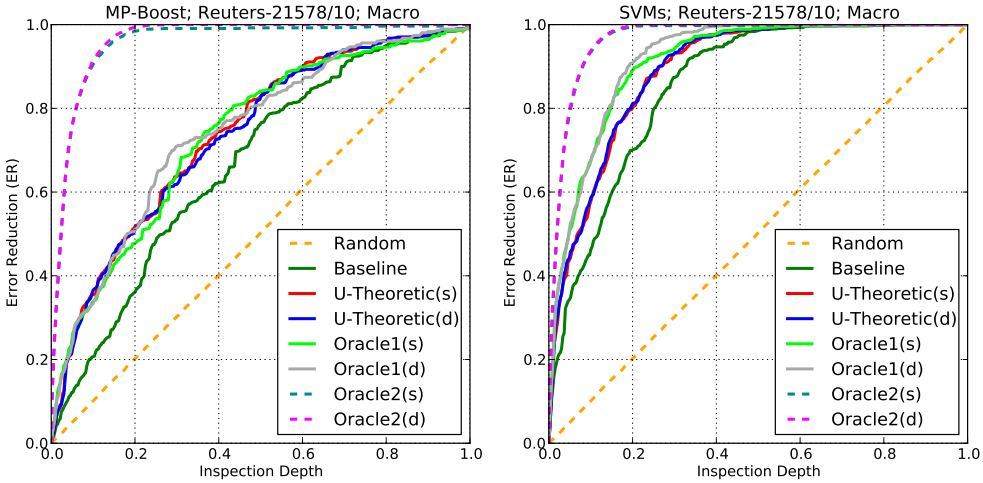


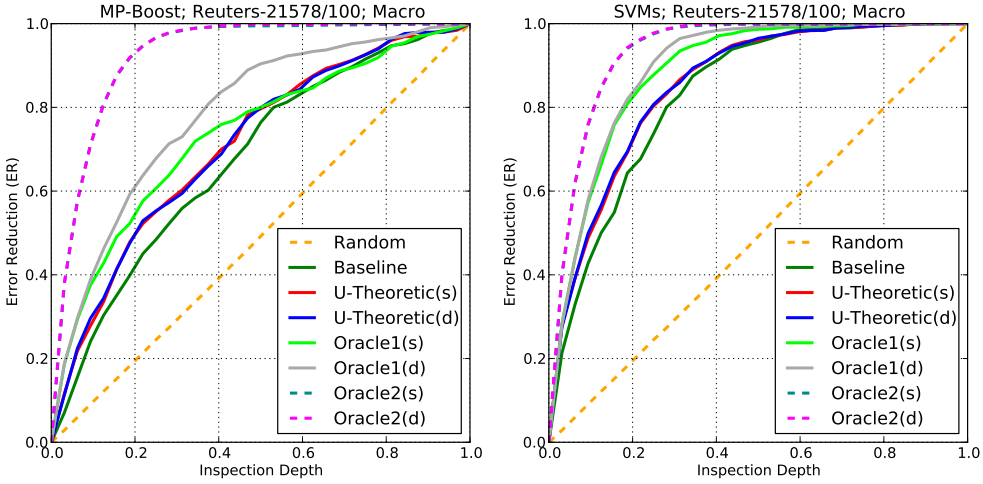
Figure 4.2: Same as Figure 4.1, but with the SVM learner in place of MP-Boost.

Actually, the part of Figure 4.1 relative to MP-Boost displays an apparently strange phenomenon, i.e., the fact that for some values of ξ the Oracle2(s) method outperforms Oracle2(d). A similar phenomenon can be noticed in some of the cells of Tables 4.1 and 4.2, where the static version of either Oracle1 or Oracle2 outperforms, even if by a small margin, the dynamic version. This seems especially strange for Oracle2(d), which is the theoretically optimal method (since it is a method that operates with perfect foreknowledge), and as such should be impossible to beat. The reason for this apparently counterintuitive behaviour lies not in the ranking methods, but in a counterintuitive property of F_1, i.e., the fact that, when $TP = FN = 0$ (i.e., there are no positives in the gold standard – and 25 out of 115 classes in the dataset used in Figure 4.1 have this property), its value is 0 when $FP > 0$ but 1 when $FN = 0$ (so, $TP = FP = FN = 0$ is a "point of discontinuity" for F_1). This essentially means that, when $TP = FN = 0$ and $FP > 0$, $G(d_i, fn_j)$ is $1/|FP|$ for the static method and 0 for the dynamic method; i.e., in this case the dynamic method does not provide any incentive for correcting a false positive, while the static method does. As a result, the static method can speed up the correction of false positives more than the dy-



(a) MP-Boost results. (b) SVMs results.

Figure 4.3: Results obtained by (i) splitting the Reuters-21578 test set into 10 random, equally-sized parts, (ii) running the analogous experiments of Figure 4.1 independently on each part, and (iii) averaging the results across the 10 parts.



(a) MP-Boost results. (b) SVMs results.

Figure 4.4: Same as Figure 4.3 but with Reuters-21578/100 in place of Reuters-21578/10.

namic method does. As mentioned above, this phenomenon exposes a suboptimality not of the dynamic method, but of the F_1 function.

		$\xi = 0.05$		$\xi = 0.10$		$\xi = 0.20$	
Reuters-21578	Baseline	.071		.108		.152	
	U-Theoretic(s)	.163	(+128%)	.226	(+109%)	.280	(+84%)
	U-Theoretic(d)	.160	(+124%)	.224	(+107%)	.279	(+84%)
	Oracle1(s)	.155	(+117%)	.222	(+106%)	.280	(+84%)
	Oracle1(d)	.152	(+113%)	.219	(+103%)	.275	(+81%)
	Oracle2(s)	.693	(+869%)	.738	(+583%)	.707	(+365%)
	Oracle2(d)	.677	(+847%)	.725	(+571%)	.699	(+360%)
Reuters-21578/10	Baseline	.063		.097		.135	
	U-Theoretic(s)	.145	(+131%)	.203	(+110%)	.245	(+81%)
	U-Theoretic(d)	.139	(+121%)	.198	(+105%)	.239	(+77%)
	Oracle1(s)	.159	(+153%)	.205	(+112%)	.243	(+80%)
	Oracle1(d)	.158	(+152%)	.212	(+119%)	.255	(+89%)
	Oracle2(s)	.555	(+784%)	.643	(+566%)	.648	(+380%)
	Oracle2(d)	.558	(+789%)	.648	(+571%)	.654	(+384%)
Reuters-21578/100	Baseline	.069		.121		.164	
	U-Theoretic(s)	.118	(+71%)	.172	(+42%)	.215	(+31%)
	U-Theoretic(d)	.119	(+72%)	.176	(+45%)	.217	(+32%)
	Oracle1(s)	.192	(+178%)	.247	(+104%)	.281	(+71%)
	Oracle1(d)	.197	(+185%)	.266	(+120%)	.318	(+94%)
	Oracle2(s)	.429	(+521%)	.537	(+344%)	.575	(+251%)
	Oracle2(d)	.429	(+521%)	.537	(+344%)	.576	(+251%)
OHSUMED	Baseline	.385		.479		.512	
	U-Theoretic(s)	.442	(+15%)	.529	(+10%)	.549	(+7%)
	U-Theoretic(d)	.443	(+15%)	.531	(+11%)	.550	(+7%)
	Oracle1(s)	.445	(+16%)	.530	(+11%)	.549	(+7%)
	Oracle1(d)	.449	(+17%)	.532	(+11%)	.550	(+7%)
	Oracle2(s)	.838	(+118%)	.839	(+75%)	.769	(+50%)
	Oracle2(d)	.758	(+97%)	.762	(+59%)	.700	(+37%)
OHSUMED-S	Baseline	.021		.025		.026	
	U-Theoretic(s)	.087	(+323%)	.118	(+374%)	.132	(+402%)
	U-Theoretic(d)	.088	(+329%)	.118	(+374%)	.132	(+402%)
	Oracle1(s)	.091	(+343%)	.117	(+370%)	.125	(+375%)
	Oracle1(d)	.094	(+358%)	.119	(+378%)	.128	(+387%)
	Oracle2(s)	.481	(+2246%)	.554	(+2125%)	.572	(+2075%)
	Oracle2(d)	.450	(+2095%)	.498	(+1900%)	.496	(+1786%)

Table 4.1: Results of various ranking methods, applied to MP-Boost and several test collections, in terms of $ENER^M_\rho(\xi)$, for $\xi \in \{0.05, 0.10, 0.20\}$.

		$\xi = 0.05$	$\xi = 0.10$	$\xi = 0.20$
Reuters-21578	Baseline	.262	.352	.420
	U-Theoretic(s)	.442 (+69%)	.531 (+51%)	.562 (+34%)
	U-Theoretic(d)	.431 (+65%)	.523 (+49%)	.557 (+33%)
	Oracle1(s)	.477 (+82%)	.563 (+60%)	.586 (+40%)
	Oracle1(d)	.476 (+82%)	.567 (+61%)	.592 (+41%)
	Oracle2(s)	.719 (+174%)	.760 (+116%)	.723 (+72%)
	Oracle2(d)	.723 (+176%)	.763 (+117%)	.724 (+72%)
Reuters-21578/10	Baseline	.243	.322	.383
	U-Theoretic(s)	.330 (+36%)	.415 (+29%)	.465 (+21%)
	U-Theoretic(d)	.335 (+38%)	.420 (+30%)	.470 (+23%)
	Oracle1(s)	.392 (+61%)	.482 (+50%)	.522 (+36%)
	Oracle1(d)	.394 (+62%)	.488 (+52%)	.531 (+39%)
	Oracle2(s)	.596 (+145%)	.676 (+110%)	.672 (+75%)
	Oracle2(d)	.599 (+147%)	.679 (+111%)	.675 (+76%)
Reuters-21578/100	Baseline	.226	.302	.364
	U-Theoretic(s)	.291 (+29%)	.365 (+21%)	.416 (+14%)
	U-Theoretic(d)	.289 (+28%)	.367 (+22%)	.419 (+15%)
	Oracle1(s)	.318 (+41%)	.422 (+40%)	.479 (+32%)
	Oracle1(d)	.318 (+41%)	.427 (+41%)	.489 (+34%)
	Oracle2(s)	.458 (+103%)	.568 (+88%)	.600 (+65%)
	Oracle2(d)	.458 (+103%)	.569 (+88%)	.601 (+65%)
OHSUMED	Baseline	.526	.630	.644
	U-Theoretic(s)	.623 (+18%)	.685 (+9%)	.666 (+3%)
	U-Theoretic(d)	.618 (+17%)	.676 (+7%)	.655 (+2%)
	Oracle1(s)	.639 (+21%)	.687 (+9%)	.657 (+2%)
	Oracle1(d)	.617 (+17%)	.659 (+5%)	.636 (-1%)
	Oracle2(s)	.864 (+64%)	.854 (+36%)	.778 (+21%)
	Oracle2(d)	.795 (+51%)	.787 (+25%)	.721 (+12%)
OHSUMED-S	Baseline	.075	.124	.164
	U-Theoretic(s)	.212 (+184%)	.282 (+127%)	.323 (+97%)
	U-Theoretic(d)	.210 (+182%)	.280 (+126%)	.321 (+96%)
	Oracle1(s)	.272 (+265%)	.334 (+169%)	.352 (+115%)
	Oracle1(d)	.301 (+303%)	.363 (+193%)	.380 (+132%)
	Oracle2(s)	.511 (+585%)	.589 (+375%)	.603 (+268%)
	Oracle2(d)	.487 (+553%)	.540 (+335%)	.536 (+227%)

Table 4.2: Results of various ranking methods, applied to SVMs and several test collections, in terms of $ENER_\rho^M(\xi)$, for $\xi \in \{0.05, 0.10, 0.20\}$.

In Table 4.3 we report the actual computation times incurred by both U-Theoretic(s) and U-Theoretic(d) on our five datasets[2]. These figures confirm that the dynamic method is (as already discussed above) substantially more expensive to run than the static method; in particular, the magnitude of this difference, together with the marginal (if any) accuracy improvements brought about by the dynamic method over the static one, shows that the static method is much more cost-effective than the dynamic one. In other words, the bad news is that the dynamic method brings about no improvement; the good news is that the computationally cheaper static method is hard to beat.

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

Table 4.3: Comparison between the actual computation times (in seconds) of the U-Theoretic(s) and U-Theoretic(d) methods on our five datasets.

4.2 A "micro-oriented" ranking function

In Chapter 3 we have assumed that the evaluation of classification algorithms across the $|\mathcal{C}|$ classes of interest is performed by *macro-averaging* the F_1 results obtained for

[2] The times reported are relative to an experiment in which the entire test set is validated; this is because, in a simulated experiment, the entire test set must be validated in order to compute the $ER_\rho^M(n)$ values reported in Figures 3.3 to 3.6. In a realistic setting in which only a portion of the ranked list is validated, the difference between U-Theoretic(s) and U-Theoretic(d) is obviously smaller, since the cost of recomputing validation gains is roughly proportional to the validation depth, and since this cost affects U-Theoretic(d) but not U-Theoretic(s).

the individual classes $c_j \in \mathcal{C}$. Consistently with this view, in Section 3.4 we have introduced macro-averaged versions of E_1, ER_ρ, NER_ρ, and $ENER_\rho$. Macro-averaging across the classes in $|\mathcal{C}|$ essentially means paying equal attention to all of them, irrespective of their frequency or other such characteristics.

However, there is an alternative, equally important way to evaluate effectiveness when a set of $|\mathcal{C}|$ classes is involved, namely, *micro-averaged* effectiveness. While macro-averaged measures are computed by first computing the measure of interest individually on each class-specific contingency table and then averaging the results, micro-averaged measures are computed by merging the $|\mathcal{C}|$ contingency tables into a single one (via summing the values of the corresponding cells) and then computing the measure of interest on the resulting table. For instance, micro-averaged F_1 (noted F_1^μ) is obtained by (i) computing the category-specific values TP_j, FP_j and FN_j for all $c_j \in \mathcal{C}$, (ii) obtaining TP as the sum of the TP_j's (same for FP and FN), and then (iii) applying Equation 2.2.1. Measures such as E_1^μ, ER_ρ^μ, NER_ρ^μ, and $ENER_\rho^\mu$ are defined in the obvious way. The net effect of using a single, global contingency table is that micro-averaged measures pay more attention to more frequent classes, i.e., the more the members of a class c_j in the test set, the more the measure is influenced by c_j.

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

While we have not explicitly discussed this, the method of Section 3.3.1 was devised with macro-averaged effectiveness in mind. To see this, note that the $U(d_i, \Omega)$ function of Equation 3.7 is based on an unweighted sum of the class-specific $U_j(d_i, \Omega)$ scores, i.e., it pays equal importance to all classes in $\mathcal{C}$. This means that Equation 3.7 is optimized for metrics that also pay equal attention to all classes, as all macro-averaged measures do. We now describe a way to modify the method of Section 3.3.1 in such a way that it is instead optimal when our effectiveness measure of choice (e.g., $ENER_\rho$) is micro-averaged. To do this, we do away with Equation 3.7 and (similarly to what happens for F_1^μ and E_1^μ) compute instead $U(d_i, \Omega)$ directly on a single, global contingency table obtained by the cell-wise sum of the class-specific contingency tables. That is, we redefine $U(d_i, \Omega)$ as

$$U(d_i, \Omega) = \sum_{c_j \in \mathcal{C}} \sum_{\omega_k \in \{tp_j, fp_j, fn_j, tn_j\}} P(\omega_k) G(d_i, \omega_k) \tag{4.3}$$

where

$$\begin{aligned} G(d_i, fp_j) &= \frac{1}{FP}(F_1^{FP}(\hat{\Phi}(Te)) - F_1(\hat{\Phi}(Te))) \\ &= \frac{1}{FP}(\frac{2TP}{2TP+FN} - \frac{2TP}{2TP+FP+FN}) \\ G(d_i, fn_j) &= \frac{1}{FN}(F_1^{FN}(\hat{\Phi}(Te)) - F_1(\hat{\Phi}(Te))) \\ &= \frac{1}{FN}(\frac{2(TP+FN)}{2(TP+FN)+FP} - \frac{2TP}{2TP+FP+FN}) \end{aligned} \tag{4.4}$$

Equations 4.4 are the same as Equation 3.5 and 3.6, but for the fact that the latter are class-specific (as indicated by the index j) while the former are global. This is due to the fact that, when using micro-averaging, there is a single contingency table, and the gain obtained by correcting, say, a false positive for c_x is equal to the gain obtained by correcting a false positive for c_y, for any $c_x, c_y \in \mathcal{C}$. Of course, Equations 4.4 are to be applied when the *static* method of Section 3.3.1 needs to be optimized for micro-averaging; when we instead want to do the same optimization for the *dynamic* method of Section 4.1, we need instead to apply, in the obvious way, "global" versions of Equations 4.2.

Actually, a second aspect in the method of Section 3.3.1 that we need to change in order for it to be optimized for micro-averaging is the probability calibration method discussed in Section 3.5.2. In fact, Equation 3.11 is clearly devised with macro-averaging in mind, since it minimizes *the average across the* $c_j \in \mathcal{C}$ of the difference between the number Pos_j^{Tr} and the expected number $\mathrm{E}[Pos_j^{Tr}]$ of positive training examples of class c_j. Again, all classes are given equal attention. For our micro-oriented method we thus replace Equation 3.11 with

$$\begin{aligned} &\arg\min_{\sigma} |Pos^{Tr} - \mathrm{E}[Pos^{Tr}]| = \\ &\arg\min_{\sigma} |\sum_{c_j \in \mathcal{C}} Pos_j^{Tr} - \sum_{c_j \in \mathcal{C}} \mathrm{E}[Pos_j^{Tr}]| = \\ &\arg\min_{\sigma} |\sum_{c_j \in \mathcal{C}} Pos_j^{Tr} - \sum_{c_j \in \mathcal{C}} \sum_{d_i \in Tr} P(c_j|d_i)| = \\ &\arg\min_{\sigma} |\sum_{c_j \in \mathcal{C}} Pos_j^{Tr} - \sum_{c_j \in \mathcal{C}} \sum_{d_i \in Tr} \frac{e^{\sigma \hat{\Phi}_j(d_i)}}{e^{\sigma \hat{\Phi}_j(d_i)} + 1}| \end{aligned} \tag{4.5}$$

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

> Suppose that $\mathcal{C} = \{c_1, c_2\}$, that $Pos_1^{Tr} = 20$ and that $Pos_2^{Tr} = 10$. Suppose that when $\sigma = a$ then $\mathrm{E}[Pos_1^{Tr}] = 18$ and $\mathrm{E}[Pos_2^{Tr}] = 8$, while when $\sigma = b$ then $\mathrm{E}[Pos_1^{Tr}] = 17$ and $\mathrm{E}[Pos_2^{Tr}] = 13$. According to Equation 3.11 value a is better than b (since $\frac{1}{|\mathcal{C}|}\sum_{c_j \in \mathcal{C}} |Pos_j^{Tr} - \mathrm{E}[Pos_j^{Tr}]|$ is equal to 2 for $\sigma = a$ and to 3 for $\sigma = b$), but according to Equation 4.5 value b is better than a (since $|Pos^{Tr} - \mathrm{E}[Pos^{Tr}]|$ is equal to 4 for $\sigma = a$ and to 0 for $\sigma = b$). □

The same smoothing methods as discussed in Section 3.3.3 can instead be used; however, note that smoothing is likely to be needed much less frequently (if at all) here since, given that we now have a single global contingency table, it is much less likely that any of its cells have values < 1.

4.2.1 Experiments

The experiments with our "micro-oriented" methods are reported in Tables 4.4 and 4.5. Note that, since the method we use as baseline corresponds (as noted in Section 3.5.4) to using U-Theoretic(s) with all validation gains set to 1, the baseline we use here is different from the baseline we had used in Section 3.5.5, since the latter was optimized for macro-averaging while the one we use here is optimized for micro-averaging. This guarantees that, in both cases, our baselines are strong ones.

The results show that utility-theoretic methods bring about a much slighter improvement with respect to the baseline, compared to what we have seen for the macro-oriented methods. For instance, for the SVM learner, REUTERS-21578 dataset, and validation depth $\xi = .10$, the improvement of our (static) micro-oriented utility-theoretic method with respect to the baseline is just +2%, while the improvement was +51% for the equivalent macro-oriented method. Across the two ranking methods (static and dynamic), five datasets, two learners, and three values of inspection depth studied, improvements range from -1% (i.e., in a few peculiar cases we even have a small deterioration) to +14%, much smaller than in the macro-oriented case in which the improvements ranged between +2% and +402%.

The main reason for these much smaller improvements lies in the combined action of two factors. The first factor is that the validation gains of Equations 4.4 are computed on the global contingency table, whose cells contain very large numbers, $|\mathcal{C}|$ times larger than the values in the local contingency tables of the macro-oriented method. This means that, since the values of the validation gains are very small (given that an increase or a decrease by 1 of very large values brings about little difference), the difference between $G(d_i, fp_j)$ and $G(d_i, fn_j)$ is even smaller. This makes the difference between the utility-theoretic methods and the baseline smaller. The second factor is that the utility function of Equation 4.3, by collapsing all the class-specific utility values for a document into a single value, tends to dwarf the differences between the documents.

It should also be noted that, in the micro-oriented method, improvements are small also *because the margins of improvement are small*. To witness, the improvements brought about by Oracle2(d) (our theoretical upper bound) with respect to the baseline are smaller than for the macro-oriented method. For instance, for the MP-BOOST learner, REUTERS-21578 dataset, and validation depth $\xi = .10$, this improvement is +168%, while it was +571% for the macro-oriented method. So, improving over the baseline is more difficult for the micro-oriented method than for the

		$\xi = 0.05$	$\xi = 0.10$	$\xi = 0.20$
Reuters-21578	Baseline	.107	.167	.222
	U-Theoretic(s)	.107 (+0%)	.168 (+1%)	.224 (+1%)
	U-Theoretic(d)	.107 (+0%)	.167 (+0%)	.224 (+1%)
	Oracle1(s)	.107 (+0%)	.168 (+1%)	.224 (+1%)
	Oracle1(d)	.107 (+0%)	.167 (+0%)	.224 (+1%)
	Oracle2(s)	.333 (+211%)	.448 (+168%)	.512 (+131%)
	Oracle2(d)	.333 (+211%)	.448 (+168%)	.512 (+131%)
Reuters-21578/10	Baseline	.110	.169	.222
	U-Theoretic(s)	.112 (+2%)	.171 (+1%)	.224 (+1%)
	U-Theoretic(d)	.113 (+3%)	.171 (+1%)	.224 (+1%)
	Oracle1(s)	.112 (+2%)	.171 (+1%)	.224 (+1%)
	Oracle1(d)	.113 (+3%)	.171 (+1%)	.224 (+1%)
	Oracle2(s)	.325 (+195%)	.438 (+159%)	.502 (+126%)
	Oracle2(d)	.325 (+195%)	.438 (+159%)	.502 (+126%)
Reuters-21578/100	Baseline	.102	.158	.208
	U-Theoretic(s)	.107 (+5%)	.163 (+3%)	.212 (+2%)
	U-Theoretic(d)	.106 (+4%)	.162 (+3%)	.211 (+1%)
	Oracle1(s)	.115 (+13%)	.170 (+8%)	.216 (+4%)
	Oracle1(d)	.116 (+14%)	.170 (+8%)	.217 (+4%)
	Oracle2(s)	.318 (+212%)	.429 (+172%)	.492 (+137%)
	Oracle2(d)	.318 (+212%)	.429 (+172%)	.492 (+137%)
OHSUMED	Baseline	.442	.552	.583
	U-Theoretic(s)	.440 (+0%)	.549 (-1%)	.580 (-1%)
	U-Theoretic(d)	.442 (+0%)	.552 (+0%)	.582 (+0%)
	Oracle1(s)	.439 (-1%)	.549 (-1%)	.580 (-1%)
	Oracle1(d)	.441 (+0%)	.551 (+0%)	.582 (+0%)
	Oracle2(s)	.660 (+49%)	.733 (+33%)	.711 (+22%)
	Oracle2(d)	.660 (+49%)	.733 (+33%)	.711 (+22%)
OHSUMED-S	Baseline	.044	.068	.094
	U-Theoretic(s)	.044 (+1%)	.069 (+3%)	.096 (+2%)
	U-Theoretic(d)	.044 (+1%)	.070 (+3%)	.097 (+3%)
	Oracle1(s)	.044 (+1%)	.069 (+3%)	.096 (+2%)
	Oracle1(d)	.044 (+1%)	.070 (+3%)	.097 (+3%)
	Oracle2(s)	.149 (+242%)	.221 (+227%)	.287 (+205%)
	Oracle2(d)	.149 (+242%)	.221 (+227%)	.287 (+205%)

Table 4.4: As Table 4.1, but with $ENER^{\mu}_{\rho}(\xi)$ in place of $ENER^{M}_{\rho}(\xi)$.

		$\xi = 0.05$		$\xi = 0.10$		$\xi = 0.20$	
Reuters-21578	Baseline	.240		.325		.389	
	U-Theoretic(s)	.246	(+3%)	.332	(+2%)	.395	(+2%)
	U-Theoretic(d)	.246	(+3%)	.331	(+2%)	.394	(+1%)
	Oracle1(s)	.246	(+3%)	.332	(+2%)	.395	(+2%)
	Oracle1(d)	.246	(+3%)	.331	(+2%)	.395	(+2%)
	Oracle2(s)	.394	(+64%)	.506	(+56%)	.556	(+43%)
	Oracle2(d)	.394	(+64%)	.506	(+56%)	.556	(+43%)
Reuters-21578/10	Baseline	.232		.317		.380	
	U-Theoretic(s)	.237	(+2%)	.323	(+2%)	.386	(+2%)
	U-Theoretic(d)	.238	(+3%)	.322	(+2%)	.383	(+1%)
	Oracle1(s)	.237	(+2%)	.324	(+2%)	.386	(+2%)
	Oracle1(d)	.238	(+3%)	.324	(+2%)	.386	(+2%)
	Oracle2(s)	.385	(+66%)	.496	(+56%)	.547	(+44%)
	Oracle2(d)	.385	(+66%)	.496	(+56%)	.547	(+44%)
Reuters-21578/100	Baseline	.223		.301		.361	
	U-Theoretic(s)	.224	(+0%)	.305	(+1%)	.366	(+1%)
	U-Theoretic(d)	.226	(+1%)	.304	(+1%)	.363	(+1%)
	Oracle1(s)	.232	(+4%)	.317	(+5%)	.377	(+4%)
	Oracle1(d)	.235	(+5%)	.322	(+7%)	.383	(+6%)
	Oracle2(s)	.367	(+65%)	.481	(+60%)	.534	(+48%)
	Oracle2(d)	.367	(+65%)	.481	(+60%)	.534	(+48%)
OHSUMED	Baseline	.492		.600		.620	
	U-Theoretic(s)	.496	(+1%)	.602	(+0%)	.621	(+0%)
	U-Theoretic(d)	.496	(+1%)	.602	(+0%)	.621	(+0%)
	Oracle1(s)	.497	(+1%)	.602	(+0%)	.621	(+0%)
	Oracle1(d)	.497	(+1%)	.603	(+1%)	.621	(+0%)
	Oracle2(s)	.704	(+43%)	.761	(+27%)	.727	(+17%)
	Oracle2(d)	.704	(+43%)	.761	(+27%)	.727	(+17%)
OHSUMED-S	Baseline	.058		.096		.136	
	U-Theoretic(s)	.063	(+10%)	.102	(+7%)	.143	(+5%)
	U-Theoretic(d)	.066	(+14%)	.104	(+9%)	.144	(+6%)
	Oracle1(s)	.064	(+10%)	.103	(+8%)	.143	(+5%)
	Oracle1(d)	.066	(+15%)	.105	(+10%)	.144	(+6%)
	Oracle2(s)	.175	(+203%)	.259	(+171%)	.330	(+143%)
	Oracle2(d)	.175	(+203%)	.259	(+171%)	.330	(+143%)

Table 4.5: As Table 4.2, but with $ENER^{\mu}_{\rho}(\xi)$ in place of $ENER^{M}_{\rho}(\xi)$.

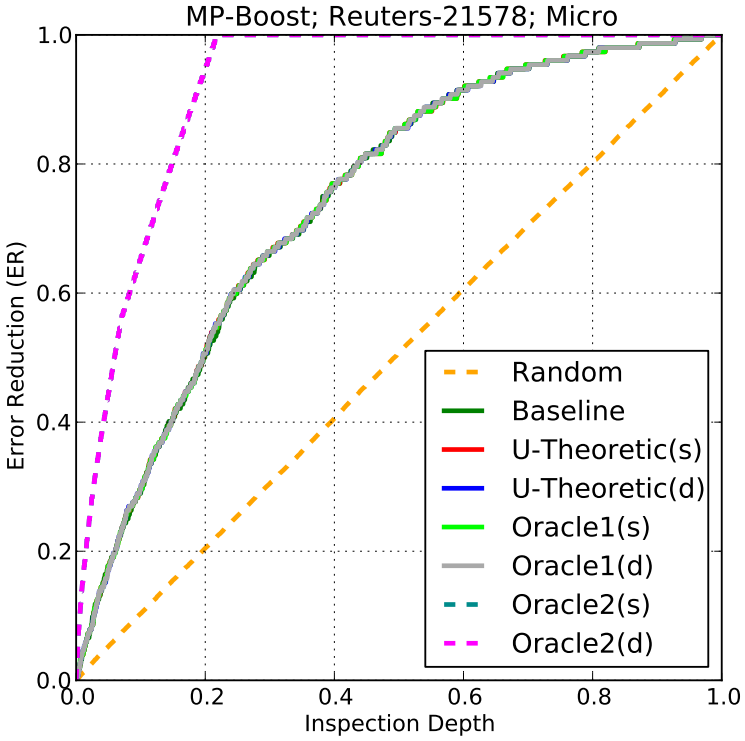


Figure 4.5: Error reduction, measured as ER^{μ}_{ρ}, as a function of validation depth. The dataset is Reuters-21578, the learner is MP-Boost.

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

The difference with macro-averaged effectiveness is highlighted by the plots of ER^{μ}_{ρ}. In Figure 4.5 we show the increase in error reduction of micro-averaged ranking methods applied on Reuters-21578 with MP-Boost. Curves are almost indistinguishable, due to the close performances of the various ranking methods. Note that the ER^{μ}_{ρ} curves are smoother than the ER^{M}_{ρ} curves of Figure 3.3. This is due to the fact that E^{μ}_{1} is evaluated on a single, global contingency table, so that correcting an individual document always has a small effect on E^{μ}_{1}. By contrast, correcting a single

document may have a major effect on a class-specific value of E_1 (especially if the class is infrequent), and this may bring about a relatively major effect on E_1^M too.

4.3 Conclusions

In this Chapter we have formalized new ranking methods: a dynamic method, along with new gain functions, which are recomputed during the iterative process of validation; a micro-averaged family of ranking methods, in which the gains are computed on the global contingency table in a multi-class multi-label scenario of TC.

We have developed the dynamic method with the intent of overcoming the sub-optimality of the static method. In the dynamic approach we keep a more accurate estimation of the utility, which takes into account of the corrections already performed during the whole activity of SATC. The experimental phase has underlined the low impact of the gain updates in the dynamic computation, with the consequence that this method is not effective as assumed. Furthermore its computation is relatively expensive, in respect to the static method. However the idea of iteratively update the gains can be still studied and extended in order to become more effective.

The micro-averaged ranking methods are a reformulation of our SATC methods with the goal of improving the micro-averaged effectiveness of classification. We have measured the classification accuracy with the F_1^μ function, in which each correction weights equally, regardless of the annotated class. Results are biased by the nature of this type of averaging, thus we can realize that the margin of improvements are small. However, micro-averaging the evaluation function is central in multi-label TC; we will see in Chapter 6 that this strategy can be effective in our SATC methods, when it is applied to other evaluation functions.

With this Chapter we have completed the coverage of all the ranking methods elaborated in this dissertation. We have summed up the results of these methods in Tables 4.1 and 4.2 for F_1^M and in Tables 4.4 and 4.5 for F_1^μ, so to have a complete picture of our work. In the rest of the dissertation we will explore new dimensions of the task, and we will also continue the work of evaluating the effectiveness of our methods, in various contexts of text classification.

5

Semi-Automated Text Classification and Active Learning

Semi-automated text classification techniques imply the active participation of a human validator. In a common TC scenario human experts annotate the data in a preliminary phase, then TC models are automatically built by means of supervised learning algorithms, and evaluated on the available data. In SATC the passive process of automatic classification is turned into an active process, in which human experts contribute to the classification task.

In supervised learning there is one task in which the learning process consists of an interaction between the learner and the human expert: *active learning* (AL). In active learning the automatism of supervised learning algorithms is combined with the manual contribution of human experts. The learner can ask the annotator to label selected data instances, in order to improve the effectiveness of the classification model. This process of learning and annotation is iterative and stops when specific requisites are satisfied.

What AL and SATC have in common is the contribution by the human annotator, and both tasks have the aim of obtaining an effective classification of data samples. The aspect of human effort is critical in AL, where we want to obtain the best model by minimizing the annotation work. The same aspect is fundamental in SATC, where we want to increase the final classification effectiveness by minimizing the validation work. There are however big differences in the processes of the two tasks, the labelling work gives different contributions, and the application scenarios are different.

In TC it is easy to provide an efficient interface to the human annotators. Texts are easily processable and classifiable by a person, the crucial point is to balance the computational and the human cost.

In this chapter we describe AL and we compare it with SATC, in the area of multi-class multi-label TC. We will discuss more deeply the differences and the reasons for choosing the right task, and, finally, we will support the discussion empirically. We introduce active learning in Section 5.1, then we present a comparison of the two tasks in Section 5.2; in Section 5.2.1 we discuss about the application of utility-theoretic

methods, originally developed for SATC, to AL; in Section 5.2.2 we do the inverse process, evaluating in SATC contexts methods originally developed for AL.

5.1 Active learning

In active learning the learner chooses which data samples to use for building the classification model. In the typical AL scenario, the learning process starts with few labelled data in the training set. After the first learning iteration, new data is choosen from a collection of unlabelled samples. The learner can then ask an oracle (i.e., a human annotator) to label the samples, using the predefined set of classes. By employing new labelled data, the learner can move to the next iteration, learning on a richer training set. This iterative process advances until the learner decides that the built model is effective, or with the human contribution is no more available.

Not all the data is useful to the learner, and the human effort available is limited. An effective AL strategy selects the samples to use as *query*[1] in order to maximize the ratio between the expected classification accuracy and the number of training samples.

In some scenarios the samples could not be immediately available. In TC, the collections from which new data are drawn are usually static and ready to be queried. It is easy to obtain unlabelled documents, it is rather expensive to manually label them. We will assume that we already own all the unlabelled data. The setting is then similar to SATC, in which the unlabelled data is represented by the test set and is all available to the classifier.

Drawing samples from a static collection is called *pool-based sampling*. We start with a small training set Tr and a large set of unlabelled samples $\mathcal{U}$. At each iteration of the AL algorithm a subset of $\mathcal{U}$ is selected and its samples used as queries. After the annotation they are added to Tr and the learning process proceeds to the next iteration. The strategy for selection is based on a measure of "benefit" for the classification model. This value is computed on all the samples in $\mathcal{U}$, so the samples that bring about the highest benefit are selected. We refer to unlabelled samples as unclassified documents and vice versa.

5.1.1 Active learning strategies

We have described the task of pool-based sampling as the process of drawing documents from a collection. The learner chooses the documents it considers more informative, according to a selection strategy. The strategy is the key of the active learning process. Choosing the most informative documents can be achieved by using the knowledge the learner has gained so far. At each iteration the learner builds a

[1] In active learning a query represents the selected samples to be annotated, that is different from the well-known notion of query in information retrieval.

classifier from the current training set, and this gives the learner a tool for analyzing unclassified documents.

A strategy based on the current learner knowledge is *uncertainty sampling* [46, 57]. A method based on uncertainty sampling selects the documents on which the current learner is less confident. The aim of the learner is to enrich the training set with the most ambiguous documents, in order to improve is ability discriminating documents in the following iterations. The confidence can be quantified in different ways. The current classifier output values can be used for estimating the confidence, that can be obtained with the absolute value of the classifier function (as seen in Chapter 2). Uncertainty is often measured as the proximity of a sample from the decision boundary. Querying the closest sample bears a more accurate fitting of the boundary in the next iteration [77].

The main advantages of this strategy are the simplicity and the lower computational cost. Most of the methods based on uncertainty sampling have a greedy approach, i.e., they select the document with the least confidence and they throw away information from the rest of $\mathcal{U}$. This myopic behavior has a consequence: noisy data and outliers can be queried. Another problem resides in the size of the training set: the initial training set is small and has little information; the learner starts with little knowledge, so its first selections can be ineffective.

After the learning process, we obtain a model that could be used on any test set. We would like to have an estimate of its accuracy on any future data, but we can not measure the effectiveness during learning. If we could compute some approximation of the future error, we could estimate the quality of the model, but above all, we could optimize the AL strategy to reduce this error. Minimizing the expected error is a strategy for selecting documents that, once learned, should produce the maximum error reduction of the classifier on future instances [69]. This strategy is able to better explore $\mathcal{U}$, doing away with the greedy character of uncertainty sampling and many of its drawbacks.

Estimating the error is the key aspect of this strategy, and it is usually performed on $\mathcal{U}$. Error estimation should be computed for each possible query, consequently a process of learning and classification is executed for each query. At each AL iteration, we build several training and validation sets. Each training set is made of the Tr at the current AL iteration, plus one of the document-class assignment candidates (all the possible combinations of document and class assignments are tested). We validate the consequent trained model on the remaining documents in $\mathcal{U}$. This step is repeated for each assignment, therefore the computational cost is prohibitive. In order to cope with the problem of computational cost, several optimizations can be enforced, like sub-sampling the unlabelled documents set [69, 51], or using efficient incremental training [89]. This AL strategy is generally limited by this characteristic of the computational cost, that can restrict the choice of machine learning techniques. On the other hand it is effective and it is not affected by outliers; furthermore several

measures of classification accuracy can be plugged into the computation of error estimation.

The described strategies have some fundamental concepts in common with SATC methods; in fact we have already discussed in the previous sections about techniques which embrace the notions of confidence and accuracy estimation. These approaches to AL and SATC are thus similar, despite the nature of the two problems. We will discuss and compare the two tasks in the last sections of this chapter.

Other strategies exist, based on the reduction of the *Version Space*, the disagreement between ensembles of classifiers, the identification of denser areas in the space of unlabelled samples. We do not go deeply in the description of AL methods, an extensive survey can be found in [74]. We instead analyze the AL approaches to multi-class multi-label TC in which multiple classes have to be managed simultaneously in the learning process.

5.1.2 Multi-label active learning

We have already dealt, in Chapter 3, with the manual annotation of documents in the case of multi-label assignments. The AL approach of asking for the annotation for each single document-class pair is inefficient in multi-label TC. After selecting a document, we expect that the annotator analyzes it in detail, so that she is able to evaluate the assignment of all classes. Once selected and annotated, the document becomes part of the acquired knowledge, and it should not be queried anymore.

The AL strategies of the previous section have been primarily studied on the task of querying for the single-label assignment. The interest of specific AL strategies for multi-label classification has increased in the last years; the aim of such strategies is to select documents that bring the best improvement to the classifier on all the classes. The importance of AL for multi-label classification is enhanced by the intrinsic difficulty of annotating documents with multiple categories: the human cost is greater for the manual classification of a document when there are multiple decisions to take.

In many of the approaches to multi-label AL, the task is firstly divided in $|\mathcal{C}|$ independent tasks, one for each class, then a document is selected by aggregating the information of all the tasks. In other approaches a query is made for each document-class assignment, thus the human annotator may read the same document $|\mathcal{C}|$ times, which requires much more effort. In this dissertation we will consider only those strategies in which the query is a document and the annotator examines all the available classes for each query.

A simple approach, using SVMs classifiers, is described in [9]. The classification problem is decomposed into $|\mathcal{C}|$ binary problems. The criterion for selecting a sample is based on its distance from the margin, for each single class. The concept of distance from the margin, which is a measure of uncertainty, can be extended to all the classification algorithms which return a confidence score. This method first calculates the minimum classification confidence for each document, and then selects the ones with the lowest minimum confidence.

Selecting documents which, once annotated and added to Tr, minimize the expected loss (or error) of the classifier, is a common strategy in AL. In the case of multiple categories the authors in [50] propose two methods for selecting samples, based on a loss function computed on the predicted classes: *Mean Max Loss* (MML) selects the samples which bring about the maximum average loss reduction on the assigned classes; *Max Loss* (ML) selects the samples which bring about the maximum loss reduction on the most confident predictions.

A more sophisticated approach comes from the work in [82]. The authors apply a method which minimizes the expected loss, called *Maximum loss reduction with Maximal Confidence* (MMC). The loss function is enriched with a "prediction of the predictions". This prediction is modelled with another classifier, which is trained in order to predict the number of classes of an unlabelled document, based on the confidence scores of the main current binary classifiers. This method is enhanced in [33], where the authors propose a framework for multi-label AL. The framework generalizes the strategy of employing an auxiliary learner, which is used together with the main learner. The loss function becomes a parameter of the framework, so an AL method can be defined choosing a combination of the two learners and a loss function.

A comparison of several approaches is discussed in [20]. Different AL methods are implemented and evaluated, exploring three dimensions of choice for the design of a method. In the "evidence" dimension the basic information for document selection is chosen, e.g., the confidence of the classifier. The "class" dimension defines the way the evidence of the multiple classes is aggregated, e.g., maximum or average value. The "weight" dimension sets the method for weighting each class, e.g., weighting more those classes on which the classifier has a low effectiveness.

[51] presents some effective query methods, which are evaluated and compared with some of the methods previously described. One method is called *Max-Margin Uncertainty sampling* (MMU) and it is based on the confidence of the binary classifiers. A document is selected according to a margin computed on two confidence values: the minimum confidence of the positive predictions and the minimum of the negative ones. Another method is called *Label Cardinality Inconsistency* (LCI) and it uses a different kind of information, based on the distribution of the classes. The inconsistency measure is defined as the absolute values of the difference between the number of predicted positive labels and the average number of classes of the current training set, the document with the highest distance is selected. A third method is a weighted combination of the previous two: at each iteration few documents are selected, one for each different weight combination; the algorithm selects the document to query with a technique based on the expected error reduction.

5.2 A comparison of active learning with semi-automated text classification

We have analyzed the task of active learning, explaining the motivations and the key aspects of minimizing the human labelling effort. In the previous chapters we have described methods for ranking documents whose goal is bringing about the best improvement in classification accuracy. The AL and SATC approaches can be similar, but the scenarios of application are different.

The problem that we want to handle with AL is to build an effective classifier with the available data. The scenario of application is wide, but it stops in the first phase of TC, learning; AL is often prior to the classification phase (however, we could continue to enrich the model during the classification, but this is often avoided). The learning phase is usually independent, it can be overly expensive, and the model is produced with an accurate choice of the learning techniques and the parameters. Furthermore, retraining several times the model becomes expensive.

A model trained using AL has to be reusable, then a sampling strategy looks further than just minimising the error, it has to draw samples that better represent the underlying distribution. The classifier must have generalization capabilities, otherwise the sample selection can easily drive the model to data overfitting. This is usually avoided with selection strategies that are exploratory, and take into account the distribution of $\mathcal{U}$, while avoiding a greedy selection behavior. Also the stopping criterion is a critical choice, namely, understanding when the model is robust. If we stop too early, the learner could have low accuracy, if we stop too late the learner could overfit. The stopping criterion is also limited by the available human resources. These problems are absent in SATC, where we point at the best effectiveness as soon as possible. In SATC we directly improve the classification accuracy after each annotation, a fact that we can not assure for AL, where there are no guarantees that a retrained model, after an annotation, will be better than the previous one.

SATC methods are meant for a final phase, subsequent to classification. This phase is independent from the previous one, especially from the learning one. SATC is applicable in a common scenario in TC: we have a ready made TC system and we use it for any new test set we receive. We are not always able to access the test set together with the training set, and new data can arrive at separated times. We do not want to rebuild the classification system, that we consider already effective, but we would like to obtain the best accuracy we can, with the available classification tools and human resources. A SATC application is completely independent of the TC system, but it is strictly dependent of the test set and its classification.

5.2.1 Semi-automated text classification methods for active learning

Methods originally defined for SATC could intuitively be suitable for an AL task. We can expect that the informativeness of the top-ranked documents is useful for

improving a classification model. The concepts of difficulty of classification and utility are close to uncertainty and error reduction. Applying a SATC method to AL means ranking $\mathcal{U}$ in order to exploit the documents that, once annotated, can better improve the accuracy of classification of $\mathcal{U}$. Our hypothesis is that this can give us a quantification of the benefit that these documents can have on the retrained model; for instance, the utility, in terms of improvement in accuracy, of $d_i \in \mathcal{U}$, can be used as measure of improvement in effectiveness of the classifier trained on $d_i \cup Tr$. In Section 5.2.1 we will discuss our intuition, with the support of the experiments. The ranking has to be computed at each AL iteration, since probability calibration and contingency tables estimation are performed on the current training set.

A utility-based method can be conceptually close to an AL method that maximizes the expected error reduction. In both we look for the document which minimizes the error on the test set (or $\mathcal{U}$). A substantial difference is in how the expected accuracy, or error, is obtained. The expected utilities are computed for each document, with the information obtained from the training set. The estimation is computed at the category level, and then we combine the class-specific utilities of each document. For estimating the error in AL, the approach is to test a new learner for each document and to perform the estimation on $\mathcal{U}$. The computational cost of this approach is much higher, but we can presume this is more effective in improving the classifier. The real difference is in the goals: a SATC method improves the accuracy on the dataset analyzed, a corrected document will ensure this boosting; while, in AL, we could not be interested in reducing the error on $\mathcal{U}$, but on any new instance drawn from the same distribution of data, thus we can not guarantee any improvement in classification accuracy.

One problem we can expect in the effectiveness of a SATC method in AL is caused by the estimations on the training set. SATC methods are designed for a scenario in which the training set is already fixed and static. In AL we start from a small training set and we iteratively enrich it with new documents, trying to create the most representative set of labelled data. The effectiveness of SATC methods is then dependent on the initial choice, or availability of labelled data, and the effectiveness in sampling new representative data.

Experimental protocol

SATC methods are easily deployable in an AL process. We can define an experimental setup for AL in which we can plug in algorithms from both families of methods. We firstly formulate three parameters of an AL process: a the number of documents in the initial training set, at the first iteration; b the number of documents to query at each iteration, namely the granularity of the retraining; c the maximum number of iterations. The selection of the initial training set is executed following the chronological order of the documents in the considered collections, selecting the first a documents. This selection strategy simulates a real scenario in which we have only access to the

set of documents produced at a specific time, and we filter new documents to annotate as they are generated. An AL process follows these steps:

1. Set the iteration counter $s = 0$;
2. Set as Tr_s the first a documents in Tr, set as $\mathcal{U}_s = Tr \setminus Tr_s$;
3. For $s = 1, \ldots, c$ repeat the following steps:
   a) Build a classifier $\hat{\Phi}^s$ from the current training set Tr_s;
   b) Evaluate the classification accuracy of $\hat{\Phi}^s$ on Te;
   c) If a SATC method is used: estimate the contingency table and execute probability calibration on Tr_s;
   d) Classify $\mathcal{U}_s$ using $\hat{\Phi}^s$;
   e) Rank $\mathcal{U}_s$ according to the AL/SATC method;
   f) Set $Tr_{s+1} \leftarrow Tr_s \cup top(\mathcal{U}_s, b)$ and $\mathcal{U}_{s+1} \leftarrow \mathcal{U}_s \setminus top(\mathcal{U}_s, b)$, where the *top* function returns the b top-ranked documents of the previous step;

The choice of parameter values follows the experimental protocol in [20]. We set $a = 100$, which is a small fraction of documents among the size of the data available for annotation. The number of queries for retraining is $b = 50$; most of the AL strategies are formulated for retraining after each query, this solution can be unfeasible because of the computational costs. Having the room for setting the granularity of retraining is mandatory in some situations, for example when we want to control the scalability of the system. We stop the process after $c = 99$ iterations, thus the number of documents in the final training set is $a + b \cdot c = 5050$.

We compare five methods, the first two are equivalent to the homonymous SATC methods, while the other three are the approaches presented in [51], which are argued by their authors as state-of-the-art methods for multi-label AL:

Baseline. This method is equivalent to uncertainty sampling applied as in multi-label TC, where the uncertainty on a document is the sum of the uncertainties on each class. A similar method is evaluated in [20] where it is called CAN. The difference is in how uncertainty is defined: in our method we compute a probability of misclassification, calibrating the probability on the training set; in CAN the uncertainty is inversely proportional to the confidence values, averaged on the classes.

U-Theoretic(s). We rank the documents of $\mathcal{U}_s$ at each iteration using the homonymous SATC method.

MMU - Max-Margin prediction Uncertainty. This method employs the margin between the group of positive classes and the group of negative ones, predicted for a document. Intuitively a classifier should maximize the margin defined as:

$$margin(d_i) = \min_{j:D_{ij}=+1} \hat{\Phi}_j(d_i) - \max_{j:D_{ij}=-1} \hat{\Phi}_j(d_i) = \min_{j:D_{ij}=+1} C_{ij} + \min_{j:D_{ij}=-1} C_{ij}$$

that is the separation between the least positive prediction confidence and the least negative prediction confidence. Documents are then ranked according to a value of uncertainty, defined as the inverse of the $margin$ function.

LCI - Label Cardinality Inconsistency. The average number of classes assigned to the documents of a dataset is also called the *label cardinality*. The difference between the number of positive classes of a document and the label cardinality can be used as a measure of likely inconsistency of the predictions. We estimate label cardinality of $\mathcal{U}_s$ on Tr_s and rank the documents in $\mathcal{U}_s$ according to the function:

$$inconsistency(d_i) = \left\| |L(d_i)| - \frac{1}{|Tr|}\sum_{e=1}^{|Tr|}|LC(d_e)| \right\|$$

where

$$L(d_i) = \{c_j : D_{ij} = +1,\, c_j \in \mathcal{C}\}\,, \quad LC(d_i) = \{c_j : \Phi_j(d_i) = +1,\, c_j \in \mathcal{C}\}$$

Adaptive. MMU and LCI are combined in this method in order to get the best of the two approaches. A documents is ranked according to the function:

$$q(d_i, \beta) = \frac{inconsistency(d_i)^{1-\beta}}{margin(d_i)^{\beta}}$$

the parameter β weights the contribution of the two functions (i.e., the higher β, the stronger the penalization given by the margin). In order to estimate the optimal value for β, at each iteration, a discrete set B of parameter values is tested. For each value in B a ranking is computed and a sampling strategy, based on the minimization of the expected error, is executed. The strategy selects the first b documents, ranked with q, then it retrains the classifier on Tr_s augmented with the selected documents and their classification. An approximate generalization error is calculated for each set of selected documents, the set which minimize this error is queried. This error is computed through the *Hinge loss*, a loss function specific for SVMs outputs, so we test this method only with this learner.
The details of the error function are described in [51]; the original approach does not integrate the b parameter, so it tests only one document for each value in B. Due to the computational cost of this method, we think that this parameter allows to better scale the method on the size of unlabelled samples. The number of retraining processes is thus tuned by the parameter b and the size of B, that we have set as $B = \{0.0, 0.1, \ldots 0.9, 1.0\}$.

We evaluate the AL methods on the datasets Reuters-21578 and OHSUMED-S. For both datasets the document dates are available, so we can order them chronologically, and we can split the training set on the first a documents. The learners are MP-Boost and SVMs, configured as in the experimental setting of SATC (see Section 3.5). The evaluation measure are F_1^M and F_1^μ; they are used as input parameters for the Baseline (the averaging method determines the probability calibration) and the U-theoretic(s) method. The AL method based on the utility is thus adjusted for a specific accuracy measure, i.e., according to the measure used in the evaluation of the test set.

Results and discussion

We show the plots of the results for the eight combinations of datasets, learners and evaluation measures. In each figure we see how the classification accuracy on Te changes as new documents are added to Tr and the classifier is retrained.

In Figure 5.1 we observe the macro-averaged results. The curves indicate the increase in F_1^M computed on the test sets during the AL process, namely after every retraining. All the learners achieve a low F_1^M at the beginning of the training, while they reach a good accuracy at the end of the process, very close to the final accuracy they achieve using the entire training set (the training sets of REUTERS-21578 and OHSUMED-S have respectively 9603 and 12358 documents). There are few differences in the results on the two datasets, but we see a substantial difference in the AL processes of the two learners. For MP-BOOST the methods have a similar effectiveness, except for the Baseline that is penalized by its heavy reliance on the probabilities of misclassification only. We have previously seen how the probabilities computed for MP-BOOST are inaccurate in reflecting the real errors made by the classifier. This problem shows up again in AL: the confidence scores output mi MP-BOOST seem to correlate little with the ground truth.

In the SVMs results we note the excellent performances of the U-theoretic(s) method. The documents annotated with high validation gains are thus very useful for the retrained models. Two facts can be decisive in these results, compared to MP-BOOST: (a) the Baseline method is strong due to the good confidence estimates of SVMs, then the probabilities of misclassification are accurate; (b) the estimation of the contingency tables on the training sets are precise in reflecting the need of specific annotations required by the model.

The U-theoretic(s) method can help to find new positive documents for an infrequent class; in fact the gain of such classes, computed on the training set, is quite high. On the test set we expect an identical distribution of the classes. The learner can improve much more its macro-averaged effectiveness obtaining more information (i.e. positively annotated documents) on these classes. Therefore the U-Theoretic(s) method selects the documents on which the classifier is less confident on the infrequent classes (with high gain). The ability of selecting documents with infrequent classes is confirmed by the results for micro-averaged effectiveness in Figure 5.2, in which we can not notice the same performance for SVMs.

All AL methods have very similar performances; the Adaptive method, based on minimizing the expected error, does not have the expected boost. It must be said that this method approximates the standard expected error reduction approach in AL: it employs only the document-class assignments made by the classifier, so it does not explore all the possible assignment candidates; it employs only $|B|$ sets of documents, and not the whole $\mathcal{U}$.

The plots show drops in the accuracy, especially in the MP-BOOST results. This happens because of the multi-label fashion of annotation and classification. The initial

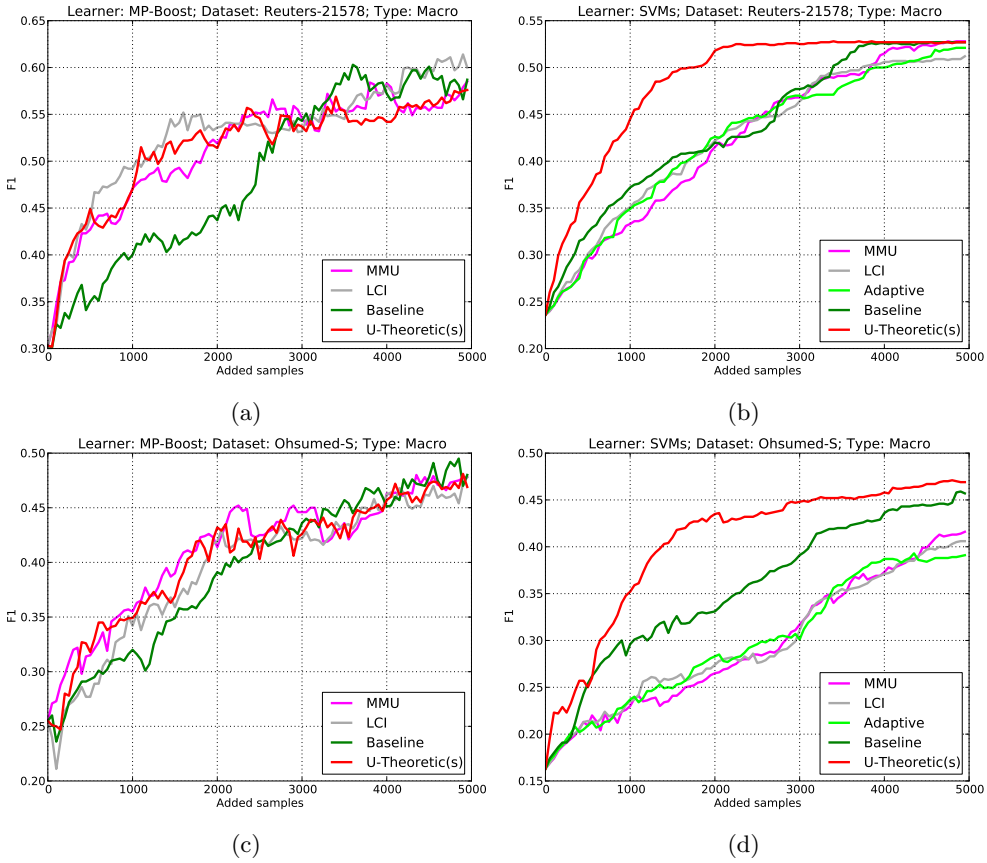


Figure 5.1: Results for macro-averaged F_1 of active learning runs, the evaluation is made on Te, learners are MP-BOOST (5.1a and 5.1c) and SVMs (5.1b and 5.1d), datasets are REUTERS-21578 (5.1a and 5.1b) and OHSUMED-S (5.1c and 5.1d).

training set can miss some classes, so if a new class appears after an iteration, the next learners cover that class. The problem is that after b documents are queried, one or very few documents are positive for a new class. Having few positive samples may cause a low accuracy of the classifier on these classes, and the macro-averaged measure is strongly affected by them. This is even more evident when a class has no positive documents in the test set, but it has positive documents in the unlabelled set. For example, in the results of MP-BOOST on the REUTERS-21578 dataset, in the Baseline ranking, after iteration 10, the category "can" obtains its first positive document in the training set, but it has not positive examples in the test set. On this class the F_1^M is always 1, until iteration 9, because the classifier does not know positive examples for the class, so it correctly does not assign any document of the test set to "can"; after iteration 9, the F_1^M goes from 1 to 0, because the classifier starts to make wrong predictions on the test set, committing 14 false positives. Only at

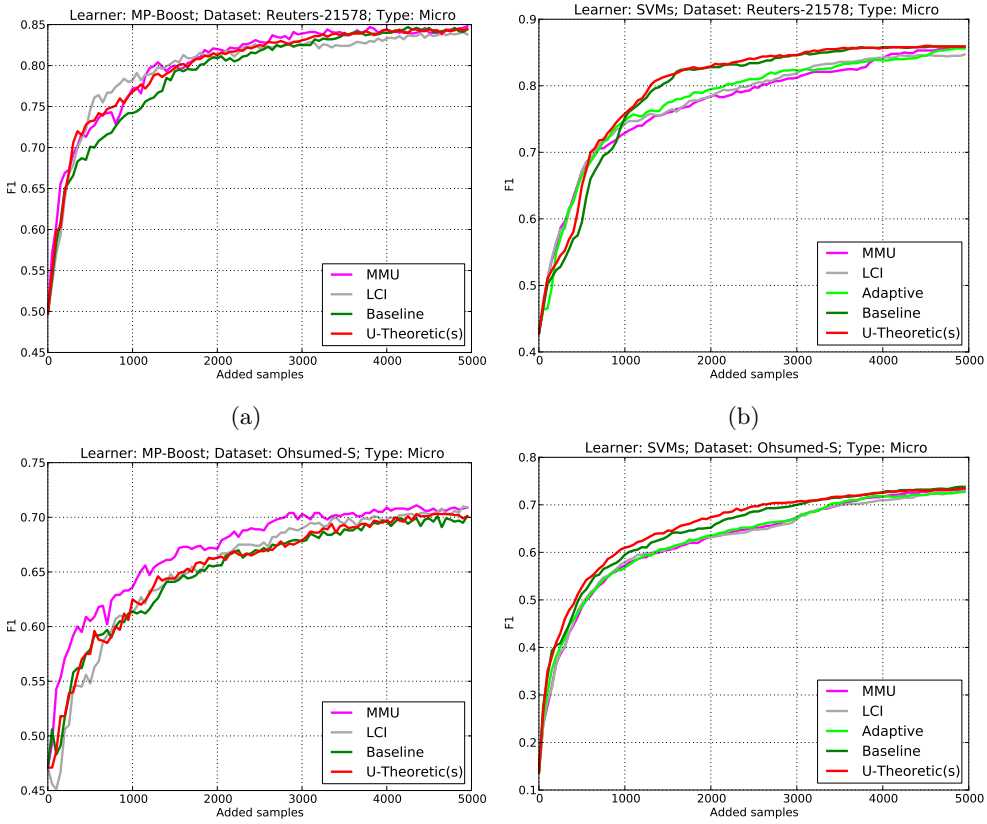


Figure 5.2: Results for micro-averaged F_1 of active learning runs, the evaluation is made on Te, learners are MP-BOOST (5.2a and 5.2c) and SVMs (5.2b and 5.2d), datasets are REUTERS-21578 (5.2a and 5.2b) and OHSUMED-S (5.2c and 5.2d).

iteration 16 the classification model gains enough information on the category "can" in order to correctly classify the test set (only true negatives). In order to prevent this scenario we could sample an initial training set in which, for each class, at least one positive document is present. Thiswould give us a less clear view of the behavior of the learners and the AL strategies on difficult data.

The results on F_1^μ are shown in Figure 5.2, following the same schema of Figure 5.1. The plots for MP-BOOST confirm the results on F_1^M, the methods have similar performance, the drops of the curves are less marked because the micro-averaged effectiveness is less susceptible to errors on infrequent classes.

For SVMs we do not notice the same improvement we have witnessed in the macro-averaged evaluations. As in SATC, the validation gains are computed on the whole estimated contingency table, so there are not class-specific weights. This implies

that the U-theoretic(s) method deviates only slightly from the Baseline. If we look at Figure 5.2b the Baseline is even less effective in the first iterations, consequently the U-Theoretic(s) method is penalized.

5.2.2 Active learning methods for semi-automated text classification

Many of the AL strategies are applicable to SATC, but their use is not always justifiable. Applying an AL strategy to SATC means substituting $\mathcal{U}$ with the test set. There is no need to retrain the model after a query, we only need to obtain a ranking of the test set.

In AL we do not look at the accuracy on the test set, in fact an AL method only analyzes the training set and $\mathcal{U}$ (which is independent of the test set). However, the AL methods based on uncertainty try to understand the behavior of the classifier, similarly to SATC methods where we estimate how much the classifier makes mistakes. An AL method that minimizes the estimated error, instead, does not involve a ranking on all the unlabelled examples. It does not try to exploit the informativeness of an unlabelled document but it only looks at the effectiveness of the built classifier. Another problem that an AL method has to solve is to avoid outliers, noisy samples that do not bring benefits to the classification model. This problem is absent in SATC, since we assume that all the documents in the test set need to be annotated.

We compare two AL methods applied to the SATC task:

MMU - Max-Margin prediction Uncertainty. This is an uncertainty sampling strategy, that is intuitively comparable with the Baseline method. Instead of summing up the probabilities of misclassification it looks at the least confident positive and negative decisions.

LCI - Label Cardinality Inconsistency. Documents with an excessive number of automatic assignments, and documents with too few assignments, can be more prone to errors. These documents can have, respectively, several false positives and many false negatives. This method measures this inconsistency and orders the documents accordingly.

We do not test the Adaptive method for one main reason: we exclude the retraining of Tr in SATC, moreover we should execute $|B|$ retraining for each document.

Results and discussion

Results are plotted in two figures: Figure 5.3 and Figure 5.4 for macro- and micro-averaged error reduction, respectively. The same results for $ENER_\rho$ are represented in Tables 5.1 and 5.2. Our U-Theoretic(s) method performs always the best, except for MP-Boost on the Reuters-21578 dataset (Figure 5.3a). The rankings with MP-Boost are problematic; we again think that the poor confidence estimates on this learner may cause low performances for methods based on confidence scores. In fact

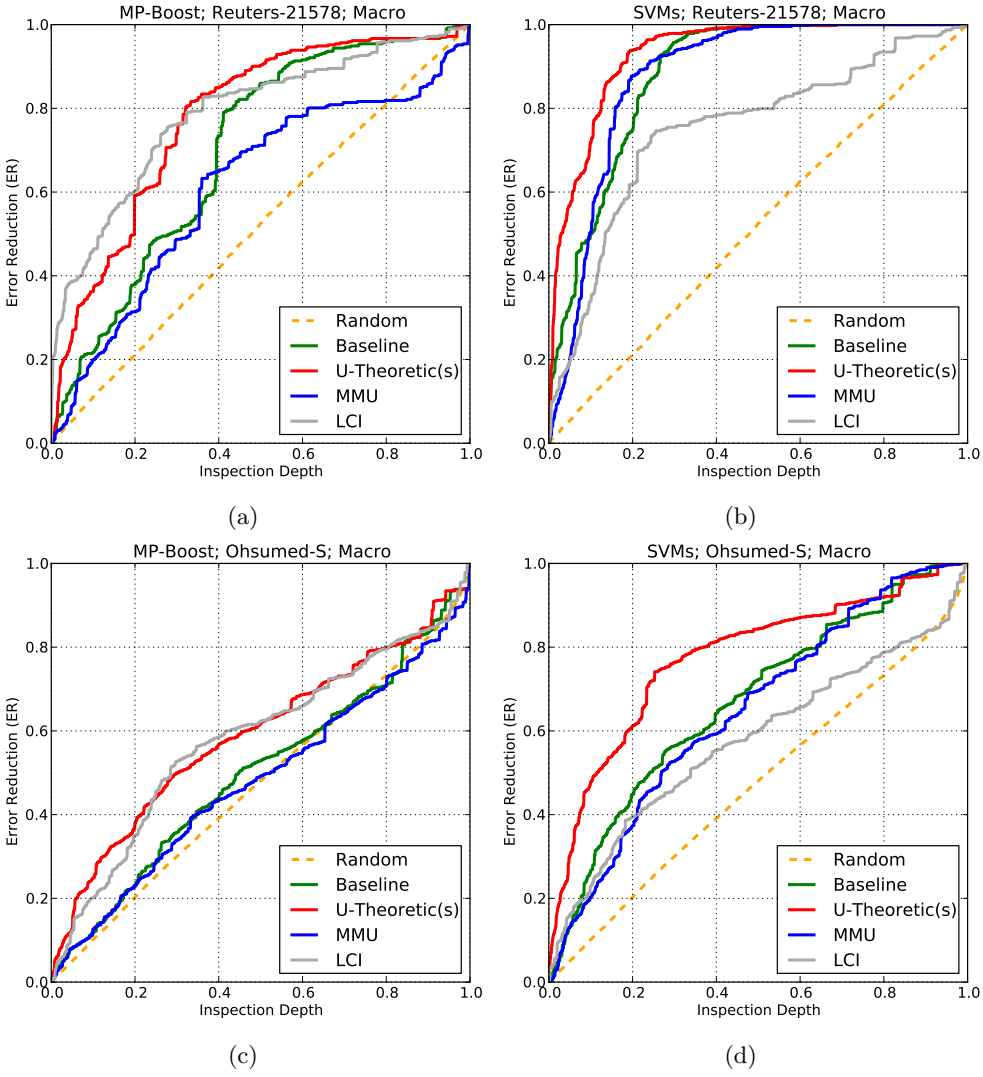


Figure 5.3: Results for macro-averaged error reduction of active learning methods applied on SATC, for learners MP-Boost (5.3a and 5.3c) and SVMs (5.3b and 5.3d), on datasets Reuters-21578 (5.3a and 5.3b) and OHSUMED-S (5.3c and 5.3d).

we can notice the lowest effectiveness of MMU, that is heavily dependent on these values. This method uses only two lowest confidence values (positive and negative predictions) per class, so even if only one confidence is not accurate (e.g., too low) the relative margin is compromised. This phenomenon influences the MMU ranking even more near the last part of the validation process.

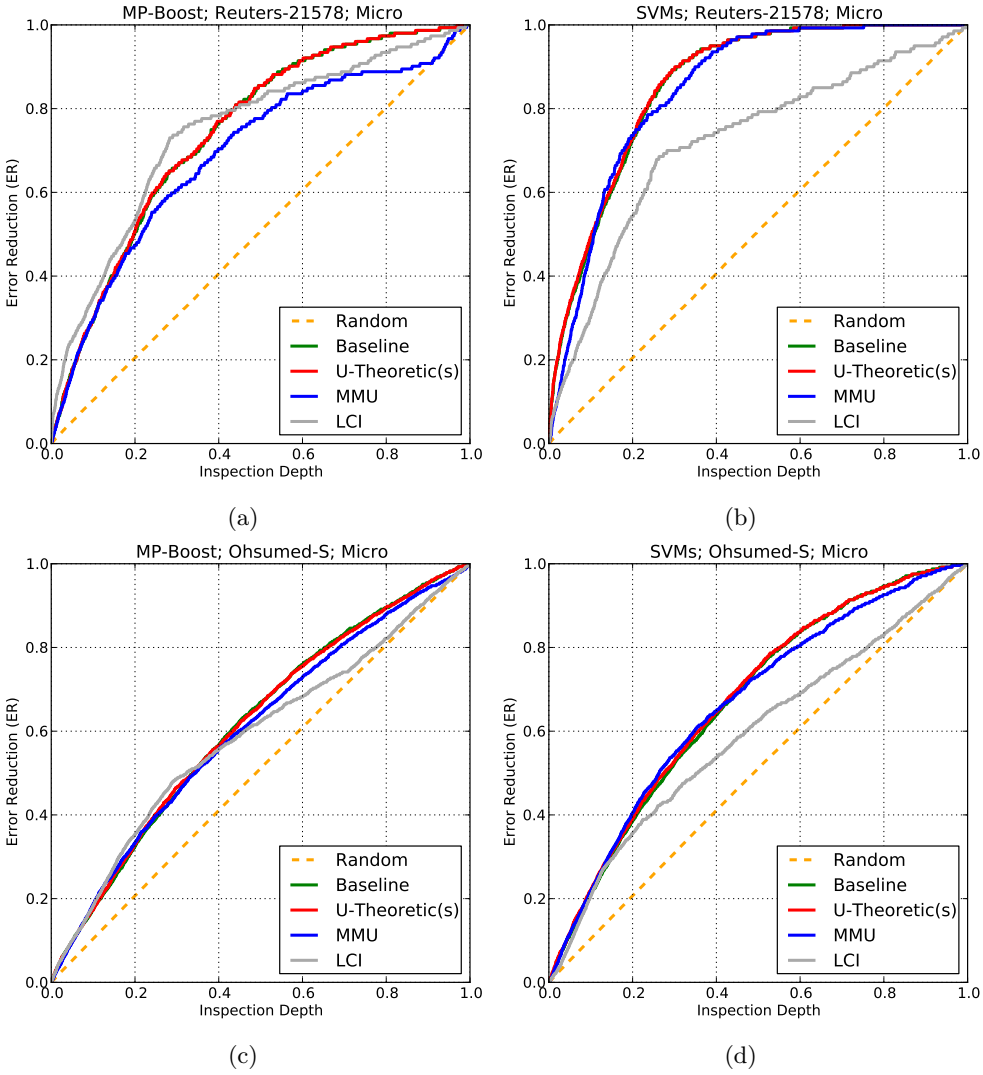


Figure 5.4: Results for micro-averaged error reduction of active learning methods applied on SATC, for learners MP-Boost (5.4a and 5.4c) and SVMs (5.4b and 5.4d), on datasets Reuters-21578 (5.4a and 5.4b) and OHSUMED-S (5.4c and 5.4d).

The MMU method always performs worse than the Baseline. We think that this method is weaker in understanding the quality of the predictions of the classifiers. If we take for example two documents with the same margin, the information about prediction errors remains hidden, e.g.: the predictions behind the margin could be distributed close to it and be potentially wrong, but with MMU we could not include these classes. Employing other methods, beyond the Baseline, based on classifier confidence, is still interesting, because mislassifications could be located only in specific

		$\xi = 0.05$	$\xi = 0.10$	$\xi = 0.20$
		MP-Boost		
Reuters-21578	Baseline	.071	.108	.152
	U-Theoretic(s)	.163 (+128%)	.226 (+109%)	.280 (+84%)
	MMU	.040 (-44%)	.072 (-33%)	.107 (-30%)
	LCI	.295 (+313%)	.335 (+210%)	.355 (+134%)
OHSUMED-S	Baseline	.021	.025	.026
	U-Theoretic(s)	.087 (+323%)	.118 (+374%)	.132 (+402%)
	MMU	.018 (-13%)	.021 (-14%)	.019 (-29%)
	LCI	.056 (+171%)	.085 (+243%)	.108 (+311%)
		SVMs		
Reuters-21578	Baseline	.262	.352	.420
	U-Theoretic(s)	.442 (+69%)	.531 (+51%)	.562 (+34%)
	MMU	.177 (-32%)	.306 (-13%)	.402 (-4%)
	LCI	.152 (-42%)	.224 (-36%)	.278 (-34%)
OHSUMED-S	Baseline	.075	.124	.164
	U-Theoretic(s)	.212 (+184%)	.282 (+127%)	.323 (+97%)
	MMU	.056 (-25%)	.092 (-25%)	.127 (-23%)
	LCI	.084 (+13%)	.111 (-10%)	.122 (-26%)

Table 5.1: Results of ranking methods based on active learning strategies, applied to MP-Boost and SVMs, on Reuters-21578 and OHSUMED-S, in terms of $ENER^M_\rho(\xi)$, for $\xi \in \{0.05, 0.10, 0.20\}$. Improvements listed for the various methods are relative to the baseline.

ranges of confidence; we could identify these confidences and consequently aggregate them for each document. Developing further methods in this way could be a future exploration of ranking strategies for SATC.

The LCI method proves effective on MP-Boost where other methods perform badly. Actually the method puts in the first ranks the documents with many classes assigned, these documents have higher probability to be classified under different classes, a sort of ambiguity that can also produce more errors. A curious behaviour of the method is readable from the curves, especially on the Reuters-21578 test set in micro-averaged results. The error reduction increases linearly until about 25% of inspection depth, then the slope drops and the linear growth is slower. If we look at

		$\xi = 0.05$	$\xi = 0.10$	$\xi = 0.20$
		MP-Boost		
Reuters-21578	Baseline	.107	.167	.222
	U-Theoretic(s)	.107 (+%)	.168 (+1%)	.224 (+1%)
	MMU	.102 (-5%)	.156 (-7%)	.199 (-10%)
	LCI	.166 (+55%)	.222 (+33%)	.267 (+20%)
OHSUMED-S	Baseline	.044	.068	.094
	U-Theoretic(s)	.044 (+1%)	.069 (+3%)	.096 (+2%)
	MMU	.042 (-3%)	.069 (+3%)	.094 (+0%)
	LCI	.047 (+8%)	.076 (+12%)	.099 (+5%)
		SVMs		
Reuters-21578	Baseline	.240	.325	.389
	U-Theoretic(s)	.246 (+3%)	.332 (+2%)	.395 (+2%)
	MMU	.179 (-25%)	.282 (-13%)	.360 (-7%)
	LCI	.124 (-48%)	.185 (-43%)	.235 (-40%)
OHSUMED-S	Baseline	.058	.096	.136
	U-Theoretic(s)	.063 (+10%)	.102 (+7%)	.143 (+5%)
	MMU	.059 (+3%)	.101 (+6%)	.142 (+4%)
	LCI	.042 (-27%)	.075 (-22%)	.098 (-28%)

Table 5.2: Results of ranking methods based on active learning strategies, applied to MP-Boost and SVMs, on Reuters-21578 and OHSUMED-S, in terms of $ENER^{\mu}_{\rho}(\xi)$, for $\xi \in \{0.05, 0.10, 0.20\}$. Improvements listed for the various methods are relative to the baseline.

the ranking we can understand this phenomenon: the last 75% of the documents are the ones classified with one class, reminding that the label cardinality of Reuters-21578 is 1.135, the score of these documents is obviously the lowest. This method, in spite of its simplicity, exploits an interesting fact about classification errors, which are generally less frequent on documents with a number of assignments equal to the average number of classes; in fact the error reduction on these documents rises slowly. In this last section of the ranking the documents position is only determined by the sorting algorithm, so the errors are uniformly distributed, and the error reduction grows linearly. Our methods excel when the confidence estimates are strong, and between the documents with one predicted class, they are able to select the ones that improve the classification accuracy the most.

5.3 Conclusions

We have devoted this chapter to a study of active learning strategies for multi-class multi-label text classification, putting them in comparison with SATC strategies for ranking classified documents. The task of AL and SATC have some aspects in common, but they definitely differ in the intent of application; so combining the solutions of the two tasks may be not really justified, but we think that some characteristics of the methods permit the experimental comparison we have conducted.

In the chapter the two tasks are analyzed in two phases. We have first applied SATC methods to AL processes, and we have then done the inverse operation, applying AL methods to SATC processes. Applying U-Theoretic(s) to AL seems promising, by observing empirical results with SVMs. The reasons and possible developments of the method should be further investigated. We consider taking advantage of the simple information of label cardinality inconsistency in SATC methods, which we have observed to be effective in some conditions.

We can finally say that AL and SATC live in different scenarios, that we need specific approaches for each of them. In multi-label AL a good direction is in weighting classes differently; we can induce from the experiments that the documents which help to discriminate infrequent classes are important, especially for macro-averaged evaluations. For SATC, the AL methods tested in these chapter are not completely effective, often they are under our baseline; this should prompt us to try new methods, that go beyond the intuitions related to AL.

6

Evaluating Semi-Automated Text Classification Applications

The application of semi-automated text classification methods is straightforward in any context of automatic text classification. The developed ranking methods can be configured and applied to any set of classified documents. Now that we have a wider vision of the task, we can explore some extensions of the problem, in order to understand more about the effectiveness of the discussed methods. In this chapter we will look at scenarios in which SATC methods are involved, so as to enrich our comprehension of the task and so as to be able to apply the strategies correctly.

In Section 6.1 we start with a pragmatic vision of SATC, as we try to answer the potential demand of a customer who asks for a guarantee of effectiveness from the validation phase. In Section 6.2 we run experiments with different evaluation measures, in order to understand how the ranking methods respond to a different need of effectiveness. In Section 6.3 we present a new evaluation measure for SATC. Finally in Section 6.4 we apply SATC methods to classification systems devised for the domain of market research.

6.1 Estimating accuracy

We have evaluated our SATC methods by using evaluation measures specifically devised for the task; this allows us to fairly compare the quality of the rankings produced by different learners and on different datasets. We have not directly shown the growth of the classification accuracy, starting from the initial effectiveness of the classifier to the final maximum value. The advantage of showing classification accuracy is that we can understand the ranking effectiveness with more conventional evaluation measures for TC.

One strong motivation for adopting SATC is performing a manual validation in order to reach the required minimum level of classification accuracy. One goal in a SATC task could be to guarantee an accuracy value after a specific number of corrections. Another goal could be to correct until a specific classification accuracy that, a certain point in the process, we have achieved. This kind of applications can

be very useful for the customer, but there is a main obstacle: we do not know the classification accuracy on the test set. In our experimental setting we have used the test sets of the collections (in which documents are already manually classified) for a comparison of the methods; in a production setting a test set can not be evaluated, we can only trust the estimates performed by the method.

In other words, we cannot know the contingency cell counts of the test set but we can estimate them. We can use the same method for obtaining contingency cell estimates that we apply for computing the rankings; the cell estimates allow to compute an estimation of the classification accuracy, which is obtained via cross-validation on the training set. During the manual validation the accuracy estimation has to be updated after each correction. In Section 4.1 we have studied a dynamic method for computing utilities, which operates by adjusting the contingency cell estimates according to the number of corrections performed. We can use the same method for updating the accuracy estimate.

The estimation of classification accuracy is a standard task in TC, the main purpose of estimating effectiveness is to compare different classification models in order to choose the best one. The standard approach is to execute a k-fold cross-validation on the labelled data [40]. For our requirements we have to estimate the contingency table; we achieved this operation in Chapter 3 by summing up the quantities of the single cells of the folds, and producing global contingency tables of the training set. This is a proper approach, which brings about an unbiased estimation of the classification accuracy, especially for a measure like F_1, which is affected by special, undefined cases (e.g., when there are zero true positives) [23].

The difficulty of estimating the classification accuracy is discussed in [80], where the authors study how the F_1 accuracy changes on the test set, sampling new data sequentially. They evaluate three cases, one of which is very similar to our scenario: the training set is fixed and test samples are randomly selected and annotated; after each selection the classifier is evaluated on the augmented test set. The goal is to reach a target value of F_1, which allows the user to accept the classifier as effective. The authors argue that trusting the estimation of F_1 (computed on the annotated subset of Te) cannot give a correct reading of the classifier accuracy; the process often falls in an overestimation of the effectiveness, making the user erroneously accept the classifier.

The problem of minimizing the cost of annotation and ensuring a certain level of effectiveness is dealt with in [5]. In this work the goal is to build training and test data with manual work, with the aim of certifying a given level of classification accuracy. The annotation work begins on the training set and, when a certain stop criterion is reached, continues on the test set. This process is conducted with the constraint of a limited budget of annotation; the authors argue that, using some policies for annotation, it is possible to certify a given level of effectiveness, thus limiting the human effort.

In our task we estimate the effectiveness on the training set, and we take it as the best estimate we can achieve for the trained model. We do not evaluate the test set, as new test samples are validated, but we update the estimated contingency table. From this strategy we can expect that: (a) the estimated accuracy is slightly underestimated, because in k-fold cross-validation we evaluate k models trained on training sets smaller than the whole Tr; (b) by avoiding to estimate the accuracy on the sequentially annotated samples of the test set, we prevent obtaining an overestimated evaluation, as pointed out in [80].

6.1.1 Comparing learning methods globally

When we have to perform a TC task we usually have the data and a number of decisions to make. In the common scenario we have a set of labelled data and a set of unclassified data, the latter can be immediately processable, or be available only in the future. Most of the decisions have to be taken in the learning phase, when we choose the algorithms and their parameters. All the choices are done for obtaining the most effective classifier with the knowledge we have.

In SATC there is another parameter that can come into play after the training: the availability of human work. In order to obtain the best effectiveness we can profit from the number of documents that one or more experts are willing to annotate. In this scenario we have to set up the classification system for both the learning phase and the manual validation phase. The reason is that one classifier can be better than another in TC, but it can be worse in SATC; this is because we obtain different rankings from different classifiers. By employing TC with SATC strategies, a production setting can consist of using a validation set for evaluating the combinations of learners and ranking methods, in order to build the optimal classification setup. The evaluation can be based on $ENER_\rho(\xi)$, the user has to set the ξ parameter in order for it to reflect the available human effort. In the next section we will compare among each other the combinations of learners and SATC methods.

6.1.2 Results and discussion

We now run some comparisons of the rankings generated from different learners. The evaluation is performed on the test sets, the same results of Chapter 3 are plotted, but directly showing the F_1 measure for each ranking. Different classifiers have a different initial effectiveness on the test set, and during the validation that follows the ranking, this effectiveness grows differently. We show only the U-Theoretic(s) method for each learner and each data set (we have seen in Chapter 4 that U-Theoretic(d) performs very similarly to the static method). We define pairs of learner and ranking method; a pair can be considered as a SATC setup, which returns a classification of the test set but also a ranking of the documents.

In order to compare the actual accuracy improvements with the estimated counterparts, for each learner we show the curves of the accuracy estimates. The estimated

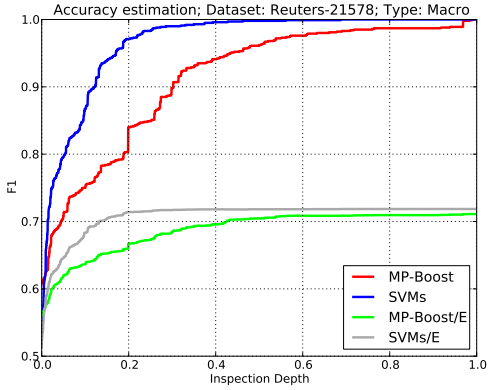
classification accuracy is measured with F_1, the initial value is the one returned by the 10-fold cross-validation on Tr. After each correction, performed following the rankings returned by the SATC method, the estimated contingency table on which F_1 is computed is modified. The updating protocol is the same used for the dynamic method, described in Section 4.1, so the cell estimates are computed and smoothed on demand, as described in Section 3.3.3.

We perform our analysis on the datasets REUTERS-21578, OHSUMED, and OHSUMED-S, with the learners MP-BOOST and SVMs. For each plot (Figure 6.1) we indicate, with the learner λ, the pairs λ/U-Theoretic(s), which are equivalent to the results in Chapters 3 and 4; we indicate the estimated counterparts as λ/E, where λ refers to the learner on which the estimated accuracy is updated according to the ranking computed by U-Theoretic(s).

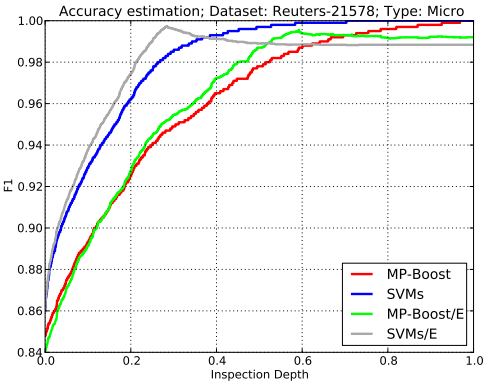
Analyzing the learners on F_1^M, we notice an interesting phenomenon on the test sets: the SVM classifier starts with the lowest F_1^M, which increases quickly as the documents are corrected. The MP-BOOST classifier instead has a good initial effectiveness, but then the ranking is not effective as the SVM one. For instance, on REUTERS-21578 and OHSUMED, the classification accuracy of SVMs is comparable to MP-BOOST after only 25 and 18 annotations, respectively. After this point of intersection, the rankings achieved with SVMs are better for validating the classification. SVMs seems then the preferable algorithm when we have some human effort available; in general this means that we can reverse our choices on the best learner, if we do not base our decision only on the estimated classification accuracy. We can say that SATC rankings give us a sort of evaluation measure of the learners, which is not only based on classification accuracy, but also on the ability to understand which documents affect more this accuracy.

In Table 6.1 we read the F_1 values on Te and the ones estimated on Tr, while in Figure 6.1 we see how the rankings impact on the estimated accuracies. We notice that the estimated values are sometimes higher and sometimes lower than the true values (we expected to have underestimated values), this means that we cannot trust the estimated effectiveness. During the validation the estimated F_1 values are updated, their values grow as classification corrections are made.

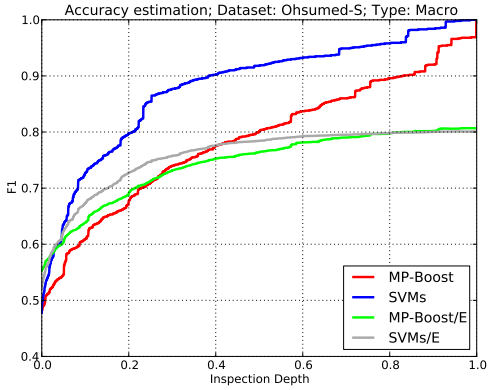
In macro-averaged F_1 the curves show an underestimation of the real improvements in accuracy, thus giving the user an inexact picture of the gained effectiveness. This happens even when the initial estimated accuracy is higher than the actual one; in this case, after few annotations (with an exception for MP-BOOST in Figure 6.1c), the updated real accuracy exceeds the estimated one. The advantage of this fact is that we obtain an estimated accuracy that we can guarantee with "enough" confidence. We achieve a sort of pessimistic approximation of the reached effectiveness, and we can thus satisfy a requirement of minimum level of gained accuracy. The reason why the estimated curves of F_1^M never reach 1 is because, for some classes,



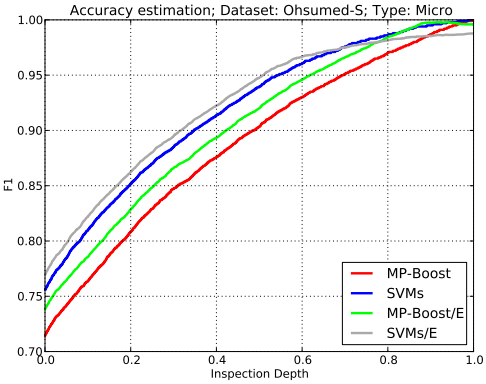
(a) F_1^M on REUTERS-21578



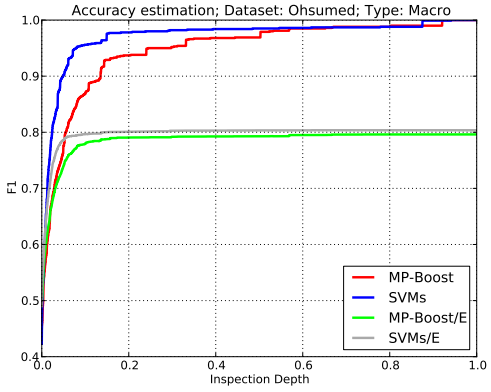
(b) F_1^μ on REUTERS-21578



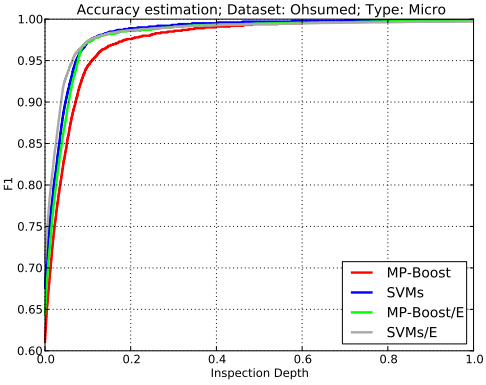
(c) F_1^M on OHSUMED-S



(d) F_1^μ on OHSUMED-S



(e) F_1^M on OHSUMED



(f) F_1^μ on OHSUMED

Figure 6.1: Comparison of the actual and estimated F_1^M and F_1^μ on the test sets. The curves describe the increment in F_1 during manual validation of the test sets.

		Evaluation Te	Estimated	Final estimated
		MP-Boost		
R	F_1^M	.608	.560	.711
	F_1^μ	.848	.840	.992
O	F_1^M	.447	.491	.796
	F_1^μ	.611	.644	.998
O-S	F_1^M	.480	.553	.807
	F_1^μ	.714	.738	.996
		SVMs		
R	F_1^M	.527	.513	.719
	F_1^μ	.860	.860	.988
O	F_1^M	.423	.490	.804
	F_1^μ	.676	.707	.997
O-S	F_1^M	.478	.521	.801
	F_1^μ	.756	.770	.988

Table 6.1: Comparison of learners effectiveness and estimated F_1^M and F_1^μ. Datasets are indicated as R for Reuters-21578, O for OHSUMED and O-S for OHSUMED-S. In the first column we indicate the evaluation of the classifiers on Te, in the second column we indicate the evaluations estimated on Tr via 10-fold cross-validation, and in the third column we indicate the estimated evaluation after the total manual validation of Te.

the corrections of the estimated contingency tables never obtain the total elimination of errors. This happens because the estimated false positives and false negatives of these classes are higher than their actual values. The values in which the curves stop growing determine the margin beyond which we cannot have any information about the classification accuracy; we show these values in the third column of Table 6.1. For example, with MP-Boost on Reuters-21578, if the minimum level of F_1^M requested is .800, we can never reach a number of annotations for which the validation "converges", namely a validation depth in which the returned estimated accuracy is .800.

In the results of micro-averaged F_1 the curves often overestimate the true progress of the manual validations. The increments are proportional, because the likelihood of the estimated contingency table is high, so also the final value is close to 1. In plots like Figure 6.1b we see a notable drop of the curve, which is due to the smoothing of

the contingency cells; it happens after the point in which the estimated false positives, or negatives, reach 0.

These results help us understand the scope of application of SATC methods. In the future we could study more rigorously the power of SATC ranking evaluations, in order to give the user the tools for validating the classification with a higher confidence of the estimations.

6.2 Other measures of classification accuracy

In this dissertation we have explored several dimensions of the task of semi-automated text classification; we have compared these methods on different datasets and learners, fixing the evaluation measure in all the experimental setting. We now show the results of SATC processes which are set up for different accuracy measures. For instance we apply and evaluate SATC methods on F_β [78], with $\beta = \{0.5, 2\}$. This function is defined as:

$$F_\beta(\hat{\Phi}_j(Te)) = (1+\beta^2) \cdot \frac{precision_j \cdot recall_j}{(\beta^2 \cdot precision_j) + recall_j} = \\ = \frac{(1+\beta^2) \cdot TP_j}{(1+\beta^2) \cdot TP_j + FP_j + (1+\beta^2) \cdot FN_j}$$

where parameter β sets the relative importance of precision and recall. With $\beta < 1$ precision weighs more, and with $\beta = 0$ we obtain the precision function; with $\beta > 1$ recall weighs more, and for $\lim_{\beta \to \infty}$ the measure coincides with the recall function. It is possible to compute the macro- and micro-averaged versions of the measure, as seen in Chapter 2.

We can define a generalized version of the average gain functions, employing the F_β measure:

$$G_\beta(d_i, fp_j) = \frac{1}{FP_j}(F_\beta^{FP}(\hat{\Phi}_j(Te)) - F_\beta(\hat{\Phi}_j(Te))) = \\ = \frac{(1+\beta^2) \cdot TP_j}{(1+\beta^2) \cdot TP_j + (1+\beta^2) \cdot FN_j} - \frac{(1+\beta^2) \cdot TP_j}{(1+\beta^2) \cdot TP_j + FP_j + (1+\beta^2) \cdot FN_j}$$

$$G_\beta(d_i, fn_j) = \frac{1}{FN_j}(F_\beta^{FN}(\hat{\Phi}_j(Te)) - F_\beta(\hat{\Phi}_j(Te))) = \\ = \frac{(1+\beta^2) \cdot (TP_j + FN_j)}{(1+\beta^2) \cdot (TP_j + FN_j) + FP_j} - \frac{(1+\beta^2) \cdot TP_j}{(1+\beta^2) \cdot TP_j + FP_j + (1+\beta^2) \cdot FN_j}$$

We can omit the definition of the pointwise gains, which follow the same structure of the functions above. Changing the values of β affects the difference between the gain values. For instance, setting $\beta = 0.5$ implies a bigger gain in correcting a false positive, which is the only type of error that impacts on precision, while setting $\beta = 2$

Dataset	$E^M_{0.5}$		$E^\mu_{0.5}$		E^M_2		E^μ_2	
	MP-B	SVMs	MP-B	SVMs	MP-B	SVMs	MP-B	SVMs
OHSUMED	.513	.514	.359	.244	.577	.617	.417	.388
OHSUMED-S	.488	.453	.264	.180	.540	.565	.306	.300
REUTERS-21578	.372	.449	.133	.091	.403	.504	.170	.185

Table 6.2: Errors generated by the MP-BOOST and SVMs classifiers, evaluating the classification accuracy with $F_{0.5}$ and F_2.

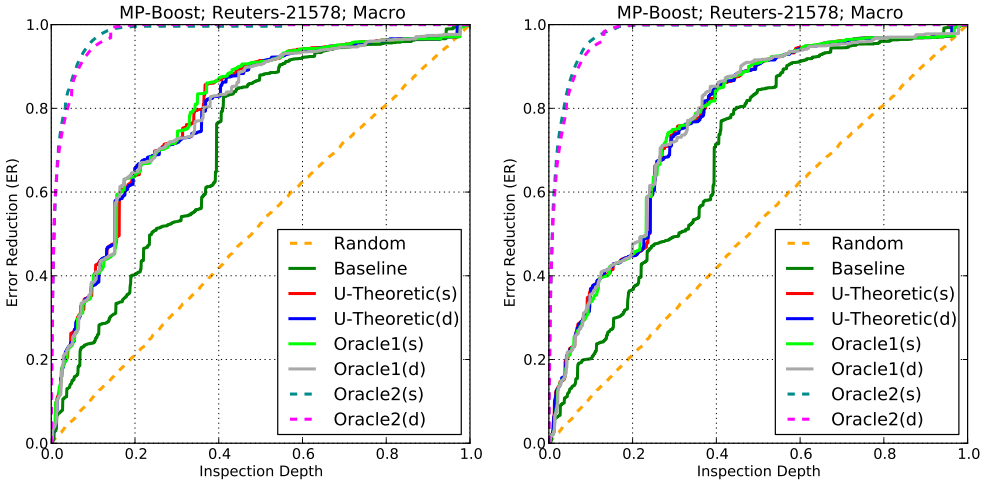
implies a bigger gain in correcting a false negative, which is the only type of error that impacts on recall.

The chosen values of 0.5 and 2 are used for defining precision- or recall-oriented evaluations respectively. An example of a precision-oriented task is *adaptive filtering* [67]. In adaptive filtering a retrieval system answers the information need of a user, returning relevant documents in real-time, as a continuous stream of content. The importance of high precision is given by the format of the results, which are presented one document at a time; the user does not want to read all the documents, but she wants a highly relevant selection, so recall is not critical. The system is effective when it does not produce false positives, at the cost of possibly making false negatives. An interesting aspect of this task is the human interaction: the user can mark non-relevant documents, in order to enrich the retrieval model with new information. This manual process is different from the SATC validation process, because it does not imply an improvement in effectiveness; non-relevant documents have to be filtered out by the system, even if they can be useful once annotated.

One recall-oriented task is *e-discovery* [62, 63], in which the goal is to retrieve documents from large collections that are relevant to a request for production in civil litigation. The documents are usually reviewed by a team of experts, so examining an entire collection is an expensive manual process. Information retrieval systems help to select only a subset of documents for manual review, but a primary requisite in e-discovery is that manual review has to cover all relevant documents. High recall is then mandatory in e-discovery.A measure like F_2 balances well the requisites of an e-discovery retrieval application. We think that, for the SATC task, $F_{0.5}$ and F_2 are more interesting measures than precision and recall. Especially the recall function has a limitation: correcting a false positive does not bring about a gain in recall, because false positives are not covered by this measure, thus $G_\infty(d_i, fp_j)$ would be undefined.

6.2.1 Results and discussion

Experiments are conducted by adopting the same protocol of Chapter 3. Regarding the evaluation measures for SATC, the only modification is in the definition of error.



(a) (b)

Figure 6.2: Results obtained on REUTERS-21578 with MP-BOOST for the measures $ER^M_{0.5}$ (Figure 6.2a) and ER^M_2 (Figure 6.2b).

We define $ENER^M_{0.5}$ and $ENER^M_2$ as the macro-averaged expected normalized error reductions, where the errors are defined as $E^M_{0.5} \equiv 1 - F^M_{0.5}$ and $E^M_2 \equiv 1 - F^M_2$ respectively. The same applies for the micro-averaged version $ENER^\mu_{0.5}$ and $ENER^\mu_2$. For simplicity of notation we omit the subscript ρ. In Table 6.2 the errors on the test sets are reported; in this experimental setting we show the results on the datasets OHSUMED, OHSUMED-S and REUTERS-21578.

We present a fraction of the results, in order to analyze only the most interesting aspects of these experiments. In Figure 6.2 we plot the curves using the same configuration of Figure 3.3 in Chapter 3, so we can observe the differences between different measures. The error reductions grow as in the results for F_1, the same happens in the results on the other datasets and learners, that we leave out for reasons of space. Utility-theoretic methods are thus effective also with these accuracy measures.

We would like to understand how the learners and ranking methods behave when the classification accuracy is more precision- or recall-oriented. In Table 6.2 we observe that the learners are more effective on $F_{0.5}$, while on F_2 they perform worse than both $F_{0.5}$ and F_1 (see also Table 3.3 in Chapter 3).

For $F^M_{0.5}$ the baselines are generally higher than for F^M_1 and F^M_2. This is because of the improved effectiveness of the classifiers; we do not report the results for $F^M_{0.5}$, which follow the trend seen for F^M_1, and we concentrate instead on $F^\mu_{0.5}$. In Table 6.3 we show the evaluations for $F^\mu_{0.5}$ (we leave out U-Theoretic(d) because of the similar performance to U-Theoretic(s)). Regarding the results with MP-BOOST, we

		$\xi = 0.05$		$\xi = 0.10$		$\xi = 0.20$	
		MP-Boost					
Reuters-21578	Baseline	.107		.167		.223	
	U-Theoretic(s)	.128	(+20%)	.185	(+11%)	.235	(+5%)
	Oracle1(s)	.128	(+20%)	.185	(+11%)	.235	(+5%)
	Oracle2(s)	.385	(+260%)	.506	(+203%)	.559	(+151%)
OHSUMED	Baseline	.475		.583		.606	
	U-Theoretic(s)	.487	(+3%)	.594	(+2%)	.615	(+1%)
	Oracle1(s)	.486	(+2%)	.594	(+2%)	.615	(+1%)
	Oracle2(s)	.680	(+43%)	.747	(+28%)	.719	(+19%)
OHSUMED-S	Baseline	.048		.072		.097	
	U-Theoretic(s)	.057	(+19%)	.085	(+18%)	.113	(+16%)
	Oracle1(s)	.056	(+19%)	.085	(+17%)	.113	(+16%)
	Oracle2(s)	.158	(+233%)	.243	(+238%)	.318	(+227%)
		SVMs					
Reuters-21578	Baseline	.242		.329		.392	
	U-Theoretic(s)	.206	(-15%)	.296	(-10%)	.366	(-7%)
	Oracle1(s)	.206	(-15%)	.296	(-10%)	.366	(-7%)
	Oracle2(s)	.435	(+80%)	.550	(+67%)	.590	(+51%)
OHSUMED	Baseline	.537		.638		.647	
	U-Theoretic(s)	.542	(+1%)	.642	(+1%)	.650	(+0%)
	Oracle1(s)	.542	(+1%)	.642	(+1%)	.650	(+%)
	Oracle2(s)	.734	(+37%)	.780	(+22%)	.738	(+14%)
OHSUMED-S	Baseline	.076		.120		.163	
	U-Theoretic(s)	.071	(-7%)	.115	(-4%)	.160	(-2%)
	Oracle1(s)	.071	(-6%)	.116	(-3%)	.161	(-1%)
	Oracle2(s)	.206	(+172%)	.306	(+155%)	.381	(+134%)

Table 6.3: Results of ranking methods applied to MP-Boost and SVMs, on Reuters-21578, OHSUMED and OHSUMED-S, in terms of $ENER^{\mu}_{0.5}(\xi)$, for $\xi \in \{0.05, 0.10, 0.20\}$. Improvements listed for the various methods are relative to the baseline.

observe a significant improvement of the utility-theoretic methods with respect to the results of Section 4.2.1, considering that the baselines are higher too. The same results

do not hold for SVMs, where we see a failure of the utility-based methods. This is actually due to the gain pairs, whose difference is minimum for SVMs, i.e., the gain of correcting false positives is very similar to the gain of correcting false negatives in these experiments. This condition seems to produce ineffective utilities.

Regarding the recall-oriented measure F_2, we obtain very good results on macro-averaged effectiveness, despite the lower performance of the baselines, which are affected by the worse effectiveness of the classifiers; utility-based methods always gain a consistent improvement over the baselines, very similar to the improvement obtained for F_1^M. We leave out the data of these results, for reasons of space, and visualize the evaluations for F_2^μ. This is the case in which our methods succeed the most with micro-averaged gains, in both MP-Boost and SVMs (even if OHSUMED remains a tough collection in this type of evaluation); in Table 6.4 we can read the results.

Document utilities are strongly affected by the value of β, because the higher the value of β, the bigger the difference between $G_\beta(d_i, fn_j)$ and $G_\beta(d_i, fp_j)$. Thus, correcting a false negative of an automatic classification, when recall is a priority, has a heavy influence on error reduction. When the two gains are more unbalanced, the micro-averaged ranking methods become effective; in fact, we can note a considerable difference with respect to the results in Section 4.2.1. This is one of the main advantages of utility-theoretic methods, which exploits particular needs of the user; the methods are completely adaptable to heterogeneous scenarios of application.

6.3 A new measure of error reduction

In Chapter 3 we defined for ranking method ρ the notion of normalized error reduction at rank n as $NER_\rho(n) = ER_\rho(n) - \frac{n}{|Te|}$. We removed the random factor from the definition of error reduction, because we expected that any ranking method can be better than a random ranking, when the method uses some smart, basic principle. In this measure we set a lower bound for the evaluation, but we still miss an upper bound, the maximum error reduction that a method can achieve.

In the experimental phase of Chapter 3 we implemented some methods called "oracles". One of these methods, Oracle2, is the best possible ranking we can obtain, but it uses information that is not accessible. The definition of such upper bounds can be extended to any method (we already designed the methods Oracle2(s) and Oracle2(d)) and, moreover, it can be used as the theoretic limit which a ranking method cannot exceed. We can exploit this upper bound in an evaluation measure, which takes into account the error reduction relative to the best ranking obtainable.

We thus define a function $S : \mathrm{P} \to \mathrm{P}$ (where P is the set of ranking methods) which returns the oracle $S(\rho)$ relative to the ranking method ρ, namely another ranking method which uses the actual contingency tables and the true labels of a test set. We design a new evaluation measure, that we call *doubly-normalized error reduction*:

		$\xi = 0.05$	$\xi = 0.10$	$\xi = 0.20$
		MP-Boost		
Reuters-21578	Baseline	.108	.167	.223
	U-Theoretic(s)	.123 (+14%)	.189 (+13%)	.246 (+10%)
	Oracle1(s)	.123 (+14%)	.189 (+13%)	.246 (+10%)
	Oracle2(s)	.349 (+223%)	.469 (+181%)	.531 (+138%)
OHSUMED	Baseline	.415	.527	.563
	U-Theoretic(s)	.415 (+0%)	.520 (-1%)	.555 (-1%)
	Oracle1(s)	.415 (+0%)	.520 (-1%)	.554 (-2%)
	Oracle2(s)	.690 (+66%)	.753 (+43%)	.723 (+28%)
OHSUMED-S	Baseline	.038	.062	.089
	U-Theoretic(s)	.053 (+40%)	.086 (+39%)	.116 (+30%)
	Oracle1(s)	.053 (+40%)	.086 (+39%)	.116 (+30%)
	Oracle2(s)	.168 (+341%)	.247 (+301%)	.317 (+255%)
		SVMs		
Reuters-21578	Baseline	.236	.320	.385
	U-Theoretic(s)	.266 (+13%)	.367 (+15%)	.433 (+12%)
	Oracle1(s)	.266 (+13%)	.367 (+15%)	.433 (+12%)
	Oracle2(s)	.407 (+72%)	.520 (+63%)	.567 (+47%)
OHSUMED	Baseline	.461	.573	.600
	U-Theoretic(s)	.485 (+5%)	.589 (+3%)	.608 (+1%)
	Oracle1(s)	.485 (+5%)	.589 (+3%)	.609 (+2%)
	Oracle2(s)	.717 (+56%)	.770 (+34%)	.732 (+22%)
OHSUMED-S	Baseline	.049	.082	.119
	U-Theoretic(s)	.074 (+51%)	.116 (+42%)	.157 (+32%)
	Oracle1(s)	.074 (+52%)	.116 (+42%)	.157 (+32%)
	Oracle2(s)	.180 (+267%)	.266 (+225%)	.339 (+185%)

Table 6.4: Results of ranking methods applied to MP-Boost and SVMs, on Reuters-21578, OHSUMED and OHSUMED-S, in terms of $ENER_2^{\mu}(\xi)$, for $\xi \in \{0.05, 0.10, 0.20\}$. Improvements listed for the various methods are relative to the baseline.

$$DNER_{\rho}(n) = \frac{NER_{\rho}(n)}{ER_{S(\rho)}(n) - \frac{n}{|Te|}}$$

The function is undefined when $ER_{S(\rho)}(n) - \frac{n}{|Te|} = 0$, namely when the lower and upper bounds coincide; in this case we take $DNER_\rho(n) = 1$, since the method ρ coincides with the oracle $S(\rho)$. The "double normalization" stands for the contributions of both NER_ρ and $ER_{S(\rho)}$. The normalized error reduction is re-normalized in order to squeeze the range of possible error reductions, which is limited by the ER of the two bounds. This measure delineates the ratio between the effective error reduction and the loss in respect to the perfect ranker; this loss is inversely proportional to the ability of the ranking method to estimate values (probabilities and gains) close to the oracle values.

$DNER_\rho$ is a new instrument for the evaluation of SATC methods, that exploits the quality of the ranking methods applied on specific learning algorithms and data. In fact, in this dissertation we have seen that the oracles behave differently when data and predictions change. The upper bounds give us a reference for comparison, so, for example, if a method seems to perform poorly, looking at its oracle makes us understand if there is real room for improvement; this aspect is exploited by $DNER_\rho$. For each method ρ and combination of learner and dataset, we decide to use as $S(\rho)$ the Oracle2 which performs the best on that combination. We could specify a single oracle for each ranking method, but then we could not compare them on the same setting, because of the different normalization of each method. This measure is however more interesting when we match the evaluations on different learners and datasets; we cannot compare these evaluations directly, because the normalization is different from one setting to another, but we can obtain a sort of *qualitative* judgement of the methods.

As for the previous measures, we define $DNER^M_\rho$ and $DNER^\mu_\rho$ for macro- and micro-averaged error reduction, respectively, and we extend the $ENER_\rho$ measure to

$$EDNER_\rho = \sum_{n=1}^{|Te|} P_s(n) DNER_\rho(n)$$

So, equivalently we have $EDNER^M_\rho$ and $EDNER^\mu_\rho$. In order to compute $EDNER_\rho(\xi)$ we have to select the best oracle in terms of $ENER_\rho(\xi)$ to use as normalization parameter of $EDNER_\rho$. We therefore formulate the oracle selection function $S : \mathrm{P}, \mathbb{R} \to \mathrm{P}$, plugged in $EDNER_\rho$, as

$$S(\rho, \xi) = \max_\rho ENER_\rho(\xi)$$

which is computed for each dataset, learner, and value of ξ; in our experiments we set $\xi \in \{0.05, 0.10, 0.20\}$. The function comes in two versions, like for any error reduction measure: $S^M(\rho, \xi)$ and $S^\mu(\rho, \xi)$.

6.3.1 Results and discussion

We show a selection of the results with the ranking methods, learners and datasets used in this dissertation, evaluated with $EDNER_\rho$; the format is the same of the

results for $ENER_\rho$. We omit the Oracle2 methods, because they obviously have an $EDNER_\rho$ equal to 1, or very close to 1 for the worst of the two Oracle2s.

In Table 6.5 we compare the evaluations of the rankings of the three versions of Reuters-21578 with MP-Boost and SVMs as learners. We can notice that the differences between the evaluations computed with $EDNER_\rho^M$ are not proportional to the corresponding differences with $ENER_\rho^M$ (see Table 3.1 in Chapter 3). In $EDNER_\rho$ the error reduction grows differently, the normalization is computed at each step of the simulated validation, and it depends of the error reduction of the oracle, thus this is not a simple re-normalization of $ENER_\rho$.

In these evaluations we point out an interesting result: on Reuters-21578/10 and Reuters-21578/100 we can judge the utility-theoretic methods to be more effective than they turned out to be in Chapter 3. In Table 6.5 we see that the expected error reduction on these two datasets (especially on the tiny datasets) is superior than on Reuters-21578, while we saw the opposite for $ENER_\rho^M$. In Chapter 3 we argued that, when test sets are small, smoothing affects the rankings to a larger degree, and that the effect of smoothing is to perturb the contingency tables, thus the difference with the oracles is larger. From these evaluations we understand that also the best rankers do not perform well on small datasets, so our ranking methods are more effective than they had proven to be, despite the more inaccurate estimation of the contingency tables.

When comparing different learners, the $EDNER_\rho^M$ measure does not give new insights with respect to the previous evaluations; the Oracle2 methods produced from MP-Boost and SVMs behave very similarly, the normalizations have thus an analogous contribution in the evaluations performed on the two learners (give a fixed a dataset). The results confirm the better performance of SVMs.

In Table 6.6 we report the evaluations of the rankings for the three versions of Reuters-21578 with the SVM learner (the evaluation measure is $EDNER_\rho^\mu$). We observe high values of $EDNER_\rho^\mu$, for both baselines and utility-based methods, that suggest a high effectiveness of the ranking methods, close to that of the best ranker. Here we also confirm the low performances of Oracle2 methods, which show the evident limitation of the manual validation in improving micro-averaged effectiveness, as pointed out in Section 4.2. We can understand that the room for improvement of the ranking methods is constrained by the nature of this type of classification accuracy: the improvements in accuracy brought about by different document corrections are homogeneous, and also with the best ranking method we can not achieve the impressive performance typical of macro-averaged effectiveness.

The $EDNER_\rho$ evaluation measure proves to be a useful tool for several analyses of SATC methods, therefore it can be used together with $ENER_\rho$ in the evaluation of the ranking methods. One possible drawback of this measure is that it requires the

		$\xi = 0.05$	$\xi = 0.10$	$\xi = 0.20$
		MP-Boost		
Reuters-21578	Baseline	0.091	0.142	0.248
	U-Theoretic(s)	0.210 (+130%)	0.294 (+107%)	0.419 (+69%)
	U-Theoretic(d)	0.206 (+126%)	0.292 (+106%)	0.417 (+68%)
	Oracle1(s)	0.196 (+115%)	0.287 (+102%)	0.417 (+68%)
	Oracle1(d)	0.192 (+110%)	0.283 (+99%)	0.411 (+66%)
R-21578/10	Baseline	0.104	0.142	0.225
	U-Theoretic(s)	0.227 (+118%)	0.290 (+104%)	0.380 (+69%)
	U-Theoretic(d)	0.220 (+112%)	0.282 (+99%)	0.371 (+65%)
	Oracle1(s)	0.270 (+160%)	0.307 (+116%)	0.386 (+72%)
	Oracle1(d)	0.268 (+158%)	0.315 (+122%)	0.400 (+78%)
R-21578/100	Baseline	0.148	0.206	0.284
	U-Theoretic(s)	0.267 (+80%)	0.307 (+49%)	0.375 (+32%)
	U-Theoretic(d)	0.267 (+80%)	0.312 (+51%)	0.379 (+33%)
	Oracle1(s)	0.447 (+202%)	0.458 (+122%)	0.492 (+73%)
	Oracle1(d)	0.455 (+207%)	0.487 (+136%)	0.554 (+95%)
		SVMs		
Reuters-21578	Baseline	0.345	0.455	0.607
	U-Theoretic(s)	0.575 (+67%)	0.677 (+49%)	0.782 (+29%)
	U-Theoretic(d)	0.558 (+62%)	0.665 (+46%)	0.774 (+28%)
	Oracle1(s)	0.624 (+81%)	0.719 (+58%)	0.811 (+34%)
	Oracle1(d)	0.621 (+80%)	0.723 (+59%)	0.819 (+35%)
R-21578/10	Baseline	0.391	0.463	0.585
	U-Theoretic(s)	0.532 (+36%)	0.598 (+29%)	0.699 (+19%)
	U-Theoretic(d)	0.539 (+38%)	0.605 (+31%)	0.705 (+21%)
	Oracle1(s)	0.635 (+62%)	0.695 (+50%)	0.777 (+33%)
	Oracle1(d)	0.639 (+63%)	0.704 (+52%)	0.791 (+35%)
R-21578/100	Baseline	0.493	0.525	0.612
	U-Theoretic(s)	0.644 (+31%)	0.645 (+23%)	0.702 (+15%)
	U-Theoretic(d)	0.638 (+29%)	0.646 (+23%)	0.705 (+15%)
	Oracle1(s)	0.690 (+40%)	0.730 (+39%)	0.795 (+30%)
	Oracle1(d)	0.688 (+40%)	0.737 (+40%)	0.809 (+32%)

Table 6.5: Results of ranking methods applied to MP-Boost and SVMs, on Reuters-21578, Reuters-21578/10 and Reuters-21578/100, in terms of $EDNER_{\rho}^{M}(\xi)$, for $\xi \in \{0.05, 0.10, 0.20\}$. Improvements listed for the various methods are relative to the baseline.

		SVMs		
		$\xi = 0.05$	$\xi = 0.10$	$\xi = 0.20$
Reuters-21578	Baseline	0.604	0.631	0.701
	U-Theoretic(s)	0.618 (+2%)	0.645 (+2%)	0.711 (+1%)
	U-Theoretic(d)	0.619 (+2%)	0.644 (+2%)	0.710 (+1%)
	Oracle1(s)	0.618 (+2%)	0.645 (+2%)	0.711 (+1%)
	Oracle1(d)	0.619 (+2%)	0.645 (+2%)	0.710 (+1%)
R-21578/10	Baseline	0.593	0.624	0.692
	U-Theoretic(s)	0.606 (+2%)	0.637 (+2%)	0.702 (+1%)
	U-Theoretic(d)	0.611 (+3%)	0.639 (+2%)	0.703 (+2%)
	Oracle1(s)	0.605 (+2%)	0.637 (+2%)	0.703 (+2%)
	Oracle1(d)	0.608 (+3%)	0.638 (+2%)	0.703 (+2%)
R-21578/100	Baseline	0.610	0.621	0.677
	U-Theoretic(s)	0.611 (+%)	0.628 (+1%)	0.685 (+1%)
	U-Theoretic(d)	0.627 (+3%)	0.637 (+3%)	0.690 (+2%)
	Oracle1(s)	0.628 (+3%)	0.650 (+5%)	0.705 (+4%)
	Oracle1(d)	0.639 (+5%)	0.660 (+6%)	0.716 (+6%)

Table 6.6: Results of ranking methods applied to SVMs, on Reuters-21578, Reuters-21578/10 and Reuters-21578/100, in terms of $EDNER^{\mu}_{\rho}(\xi)$, for $\xi \in \{0.05, 0.10, 0.20\}$. Improvements listed for the various methods are relative to the baseline.

computation of the Oracle2 methods, a requisite that can become tedious when the methods have to be validated on multiple, large datasets.

6.4 Experiments with automated verbatim coding

In the field of market research a fundamental instrument for obtaining informative data from customer surveys is the use of open-ended questions. This is valid in many other fields in industry, as customer relationship, opinion research in political sciences, etc. In this area a lot of human effort is spent for the analysis of data, which is usually classified according to a *codeframe* (i.e., a classification scheme); the codeframe is developed along with the questions. Applications for the automatic coding of open-ended *verbatim*[1] responses have been developed in recent years [21]. These

[1] A verbatim is a piece of text that is an exact transcription of the user's answer, which could have been originally produced using various media (e.g., telephone interview, handwritten text, SMS answer, email, web form, etc.).

Dataset	$\|Tr\|$	$\|Te\|$	$\|\mathcal{C}\|$	AVC	AVL	E_1^M		E_1^μ	
						MP-B	SVMs	MP-B	SVMs
LL-A	140	61	17	15.35	1.21	.093	.090	.063	.035
LL-B	350	151	31	20.48	1.62	.185	.246	.080	.089
LL-C	140	61	17	8.24	1.59	.102	.059	.068	.018
LL-D	350	151	27	31.30	2.05	.506	.701	.148	.161
LL-E	140	61	36	6.53	2.59	.173	.221	.118	.139
LL-F	350	151	56	26.45	3.96	.334	.477	.178	.192
LL-G	350	151	100	15.68	3.87	.377	.518	.230	.242
LL-H	350	151	84	21.73	4.83	.404	.620	.218	.269
LL-I	350	151	67	23.81	4.60	.398	.486	.219	.238
LL-L	350	151	65	20.55	3.15	.554	.574	.259	.255
Egg-A1	700	300	16	86.56	1.98	.691	.787	.468	.464
Egg-A2	700	300	16	86.56	1.98	.685	.728	.498	.479
Egg-B1	653	273	21	50.38	1.62	.610	.614	.403	.387
Egg-B2	653	273	21	50.38	1.62	.612	.685	.437	.414
ANES L/D	1865	800	1	969.00	0.52	.156	.137	.156	.137

Table 6.7: Characteristics of the ten market research datasets (LL-A to LL-L), four customer satisfaction datasets (Egg-A1 to Egg-B2), and one social science dataset (ANES L/D), that we have used here for experimentation. The columns represent the name of the dataset, the number of training verbatims ($|Tr|$) and the number of test verbatims ($|Te|$), the number of codes in the codeframe ($|\mathcal{C}|$), the average number of positive training verbatims per code (AVC), the average training verbatim length (AVL), and the initial error (both E_1^M and E_1^μ) generated by the MP-Boost and SVMs classifiers.

applications rely on a supervised learning approach, which means that training data is indispensable, and is commonly available from the already coded verbatims of a company. Automatic verbatim coders allow companies to reduce the manual human work, which only becomes necessary for the training phase or for successive phases of retraining (i.e., active learning).

In this field semi-automated text classification can play a productive role, giving the human coders a further tool for improving the classification of verbatims. In order to test our SATC methods on this domain we can experiments on datasets of real survey data.

Table 6.7 lists the main characteristics of the 15 datasets we have used. The first 10 datasets (LL-A to LL-L) consist of verbatims from market research surveys and were provided by Language Logic LLC. The LL-B, LL-D, and LL-F to LL-L datasets are from a large consumer packaged-good study, with both open-ended and brand-list questions. The LL-A, LL-C, and LL-E datasets are instead from one wave of a continuous ("tracking") survey that Language Logic LLC codes 12 times a year, which consists of "semi-open" brand questions (i.e., questions – such as "What is your favourite soft drink?" – that, although in principle eliciting a textual response, usually generate many responses consisting of only the name of a product or brand, with this name coming from a small set of such names). The next 4 datasets consist of verbatims from customer satisfaction surveys and were provided by Egg PLC; for both datasets, which were collected in the context of two different surveys, respondents were answering the question "Have we done anything recently that has especially disappointed you?". Actually, the Egg-A1 and Egg-A2 datasets contain the same verbatims, but the test verbatims differ in the codes applied to them, since they were coded independently by two different human coders, so as to provide data for an intercoder agreement study (see e.g., [11]); so, we treat them as separate datasets. The same goes for the Egg-B1 and Egg-B2 datasets. The last dataset (ANES L/D) consists of verbatims from a political survey run in 1992 and were obtained from the American National Election Studies (ANES) committee. Two sets of verbatims were used: the first were returned in answer to the question "Is there anything in particular about Mr. Clinton that might make you want to vote for him? If so, what is that?" while the second were returned in answer to the question "Is there anything in particular about Mr. Clinton that might make you want to vote against him? What is that?". Our coding task consisted in guessing whether the verbatim belongs to the former or to the latter set. For all these 15 datasets, see [21, Table I] for more details.

It is apparent from the last column of Table 6.7 that the LL-A to LL-L and ANES L/D datasets appear "easier" (i.e., they give rise to error values lower than .269 for E_1^{μ}), while the Egg datasets appear "harder" (with error values higher than .387 for E_1^{μ}). We might wonder whether the datasets characterized by lower error are also characterized by higher $ENER_{\rho}$ values. To investigate this, instead of visualizing the expected normalized error reduction for each single dataset, in Tables 6.8 and 6.9 we represent the results of the methods presented in Chapter 3 for macro- and micro-averaged effectiveness, separately averaged across the eleven "easier" datasets (LL and ANES) and across the four "harder" ones (Egg). Furthermore it is clear from Table 6.7 that the size of these datasets, and the average sizes of the verbatims, are considerably smaller than the datasets we have analyzed so far.

When looking at the results for $ENER_{\rho}^{M}$, we immediately notice that the baselines of the easier datasets are higher. This is because the confidence estimates on these datasets are more reliable, due to the higher effectiveness of the classifiers. On the

		$\xi = 0.05$		$\xi = 0.10$		$\xi = 0.20$	
		MP-Boost					
easier	Baseline	.093		.153		.193	
	U-Theoretic(s)	.097	(+4%)	.155	(+1%)	.190	(-1%)
	Oracle1(s)	.092	(-1%)	.151	(-1%)	.187	(-3%)
	Oracle2(s)	.317	(+240%)	.420	(+175%)	.474	(+145%)
harder	Baseline	.008		.008		.010	
	U-Theoretic(s)	.023	(+204%)	.039	(+413%)	.051	(+394%)
	Oracle1(s)	.037	(+386%)	.045	(+485%)	.050	(+382%)
	Oracle2(s)	.229	(+2893%)	.299	(+3800%)	.350	(+3282%)
		SVMs					
easier	Baseline	.121		.207		.273	
	U-Theoretic(s)	.174	(+43%)	.261	(+26%)	.322	(+18%)
	Oracle1(s)	.222	(+83%)	.304	(+47%)	.354	(+30%)
	Oracle2(s)	.368	(+203%)	.462	(+123%)	.503	(+85%)
harder	Baseline	.075		.114		.151	
	U-Theoretic(s)	.089	(+19%)	.135	(+18%)	.178	(+18%)
	Oracle1(s)	.112	(+50%)	.159	(+39%)	.200	(+32%)
	Oracle2(s)	.286	(+281%)	.369	(+222%)	.420	(+177%)

Table 6.8: Results of SATC ranking methods on survey datasets, in terms of $ENER_\rho^M(\xi)$, for $\xi \in \{0.05, 0.10, 0.20\}$. Improvements listed for the various methods are relative to the baseline.

harder datasets, with MP-Boost the Baseline method performs poorly, close to a random ranker. In this case we see a substantial contribution of the gains in improving the expected error reduction, even if the values of $ENER_\rho^M$ are still low; the Oracle2 methods give us the idea of how much room there is for improvement, which means that we should firstly enhance the classifier effectiveness, then the ranking methods can consequently perform better. Except for this case, on the four harder datasets we notice a lower improvement of the utility-based methods with respect to the baselines. The likely reason for this fact is not that the latter four datasets are easier, but that the average imbalance between positive and negative examples (i.e., the ratio between the value in the AVC column and the number of samples in the datasets of Table 6.7) happens to be smaller for these four datasets than for the other eleven datasets; a smaller imbalance means that the difference in gain between correcting a false positive

		$\xi = 0.05$	$\xi = 0.10$	$\xi = 0.20$
		MP-Boost		
easier	Baseline	.110	.171	.210
	U-Theoretic(s)	.097 (-12%)	.153 (-10%)	.198 (-6%)
	Oracle1(s)	.097 (-12%)	.154 (-10%)	.197 (-6%)
	Oracle2(s)	.244 (+122%)	.352 (+106%)	.420 (+100%)
harder	Baseline	.013	.024	.036
	U-Theoretic(s)	.014 (+4%)	.022 (-6%)	.031 (-15%)
	Oracle1(s)	.013 (+3%)	.022 (-6%)	.031 (-14%)
	Oracle2(s)	.103 (+697%)	.157 (+566%)	.208 (+471%)
		SVMs		
easier	Baseline	.111	.187	.255
	U-Theoretic(s)	.116 (+4%)	.193 (+3%)	.260 (+2%)
	Oracle1(s)	.160 (+44%)	.231 (+23%)	.287 (+13%)
	Oracle2(s)	.280 (+153%)	.374 (+100%)	.429 (+69%)
harder	Baseline	.047	.074	.102
	U-Theoretic(s)	.048 (+2%)	.074 (+0%)	.103 (+1%)
	Oracle1(s)	.049 (+3%)	.075 (+2%)	.104 (+2%)
	Oracle2(s)	.135 (+185%)	.201 (+172%)	.259 (+155%)

Table 6.9: Results of SATC ranking methods on survey datasets, in terms of $ENER^{\mu}_{\rho}(\xi)$, for $\xi \in \{0.05, 0.10, 0.20\}$. Improvements listed for the various methods are relative to the baseline.

and correcting a false negative is smaller, which makes the utility-theoretic method more similar to the baseline (probabilistic) method.

U-Theoretic(s) rankings produced from MP-Boost classifications, applied on the easier datasets, have an evident difficulty in improving over the Baseline (especially for $ENER^{\mu}_{\rho}$). This happens also for the Oracle1 methods, which indicates that the estimation of the contingency table cannot be one of the causes. Looking at the results on the single datasets (not shown here) it seems that the poor performance of U-Theoretic(s) is correlated with the performance of the baselines, but in an opposite way: the higher the Baseline error reduction, the lower the U-Theoretic(s) error reduction. This is then due to a bad combination of the probability and gain values in the computation of utility; confidence estimates of MP-Boost have an irregular distribution, which can impact negatively on the utility function, especially when the datasets are small and the features are limited, as in the case of these datasets.

Results are different for SVMs, which seems a preferable learner for this task. U-Theoretic(s) always achieves the best performance, also in micro-averaged error reduction, even if it is less notable. Both harder and easier datasets present characteristics specific to the survey data (e.g., the average size of the verbatims), which make the application of learning algorithms, and SATC methods, a compelling process. This scenario of application deserves further studies, in order to better combine supervised learning algorithms with ranking methods for manual validation.

6.5 Conclusions

In this chapter we have worked at building a wider picture of the effectiveness of our ranking methods for semi-automated text classification. We have studied some development and extensions of SATC applications, in order to examine the feasibility of the ranking methods in different scenarios.

We have found that estimating the classification accuracy after a manual validation is a difficult task, and it is not trivial to certify a specific level of effectiveness. We can however refine the accuracy estimation through annotations, and acknowledge, with some degree of confidence, the minimum level of effectiveness of the automatic classification. In the future we could study and quantify the confidence of these estimations, by working on the data with statistical tools.

The F_β measure allows the customer to evaluate a classification according to particular requisites. We have shown that utility-theoretic methods allow to drastically improve the classification effectiveness when a manual validation is performed. Our ranking methods identify the documents that, once corrected, better satisfy predefined requirements, which can be oriented to precision or on recall.

We have defined a new measure for the evaluation of ranking methods, namely $EDNER_\rho$. The results evaluated with this measure show how ranking methods behave in comparison with the perfect rankers. We have seen that, when our methods are close to the oracles, or when the oracles determine a below-average upper bound, this measure allows interesting analyses to be carried out.

Finally, we have conducted experiments with data from market research surveys, achieving good results, especially with SVMs.

It is clear that the potential of SATC applications is extensive, and the development of new instruments in the SATC framework can be further explored in the future.

7

Conclusions

We have presented a range of methods, all based on utility theory, for ranking the documents labelled by an automatic classifier. The documents are ranked in order to maximize the expected improvement in classification accuracy brought about by a human annotator who validates some of the documents, starting from the top-ranked ones. We call this task *semi-automated text classification*. We have also proposed an evaluation measure for such ranking methods, based on the expectation of the reduction in error brought about by the human annotator's validation activity.

In Chapter 3 we have developed a utility-theoretic ranking method for macro-averaged effectiveness, and a first variant of the SATC evaluation measure (the *expected normalized error reduction*).

We have then introduced additional utility-theoretic ranking methods in Chapter 4: a "dynamic" version of the basic, "static" method of Chapter 3, and micro-averaged versions of both. Experiments have been carried out on standard datasets and standard learning algorithms for text classification. The results show that the intuition of using utility theory is correct. In particular, out of four methods evaluated, we have found that the two methods optimized for macro-averaged effectiveness deliver drastically improved performance with respect to the baseline, while the two methods optimized for micro-averaged effectiveness bring about only limited improvements for F_1. We have also found that the two static methods, while seemingly inferior to the dynamic ones on a purely intuitive basis, perform as well as the dynamic ones at a fraction of the computational cost.

In Chapter 5 we have studied analogies and differences between the tasks of SATC and active learning, in order to better understand the importance and the potential of SATC methods. We have conducted an empirical analysis of the feasibility of SATC methods in an AL scenario. We have looked at how AL methods behave in a SATC task. Aside from the evident difference of the two tasks, the results of utility-theoretic ranking methods, as applied to AL look promising.

In Chapter 6 we have analyzed a pragmatic scenario for the application of our SATC methods, with the aim of identifying the stopping criterion in the manual an-

notation process. We have then continued in the evaluation of our SATC methods, showing the good performance of micro-averaged ranking function for the F_2 evaluation function, introducing an additional variant of the SATC evaluation measure, and conducting an empirical evaluation of SATC methods on datasets specifically devised for the classification of surveys for market research.

To the best of our knowledge, the work presented in this dissertation is the first in the literature to address semi-automated text classification as a task in its own, and to present methods explicitly devised for optimizing it.

7.1 Future directions

Due to the novelty of the task of semi-automated text classification, the potential of new research studies is vast, as well as the opportunities of implementing ranking methods in TC applications (e.g., e-discovery).

We could test new learners, especially the ones based on supervised learning algorithms which output probability values as confidence estimates. By employing these algorithms we could do away with the costly phase of probability calibration, and directly obtain accurate probabilities of misclassification. Some examples of such algorithms are logistic regression [88] or Naive Bayes [44]. We also remark that our techniques are obviously not limited to *text* classification, but can be useful in any classification context in which class imbalance [30], or cost-sensitivity [1] in general [18], suggest using a measure (such as F_β) that caters for these characteristics.

The aspect of cost in classification has been differently tackled in active learning [55], where the cost is applied to the activity of labelling. In a SATC application the cost of validation (i.e., the cost of inspecting and eventually correcting a document) could be integrated in the ranking function; e.g., the cost could be proportional to the length of the document. In this case, for example, correcting a document of 100 words which brings about an improvement of 20% in accuracy is suboptimal with respect to correcting two documents of 15 words each which bring an improvement of 10% each. Also the evaluation measure should consider this cost. In [24] the author defines an evaluation measure for information retrieval, based on modelling user behaviour, in which the cost of certain user choices is summed to the product of the probability and the benefit of the choices. This approach could be also followed in the definition of our measures.

The *expected reciprocal rank* (ERR) measure [13] (see Section 3.6) can be a starting point for creating a more sophisticated evaluation measure, that models the behaviour of the user who pays attention to the work made. For example, a user is likely to continue a validation if she has not performed enough corrections, or she is likely to stop if the corrections are frequent. An evaluation measure can model this situation, that

[1] Cost-sensitive classification extends regular classification by charging different costs for different types of classification errors.

is closer to a concrete validation activity, in which the user determines a stopping criterion based on her estimation of the error reduction. Following this approach of providing a sort of estimation of the user "persistence", we could define additional methods for estimating the probability of the user to stop, independently of the ranking; for example we could compute some statistics on the average number of labels (and errors) that a document can have.

New ranking methods could be developed, leaving the utility-based strategy, and embracing standard techniques for document ranking. *Learning to rank* refers to machine learning techniques used for the activity of ranking; nowadays it is the standard approach to ranking in information retrieval [53]. A learning to rank method for SATC could learn, from the training set, the characteristics of the documents which make them prone to be misclassified. Besides learning to rank, we could continue to explore active learning strategies in order to find the correct methods to be adapted to SATC.

We will continue to carry out new experiments, using new datasets and learners, but also new methods for contingency table smoothing and probability calibration. We will aim at the statistical robustness of our evaluations, by using more data, with the final objective of achieving accurate estimations of the error reduction, independently of the datasets used.

References

1. IJsbrand J. Aalbersberg. Incremental relevance feedback. In *Proceedings of the 15th ACM International Conference on Research and Development in Information Retrieval (SIGIR 1992)*, pages 11–22, Copenhagen, DK, 1992.
2. Charu C. Aggarwal and Cheng Xiang Zhai. *Mining Text Data*. Springer, 2012.
3. Omar Alonso and Ricardo Baeza-Yates. Design and implementation of relevance assessments using crowdsourcing. In *Proceedings of the 33rd European Conference on Advances in Information Retrieval (ECIR 2011)*, pages 153–164, Dublin, IE, 2011.
4. Paul Anand. *Foundations of Rational Choice under Risk*. Oxford University Press, Oxford, UK, 1993.
5. Mossaab Bagdouri, William Webber, David D. Lewis, and Douglas W. Oard. Towards minimizing the annotation cost of certified text classification. In *Proceedings of the 22nd ACM international conference on Conference on information and knowledge management (CIKM 2013)*, pages 989–998, San Francisco, US, 2013.
6. Paul N. Bennett. Using asymmetric distributions to improve text classifier probability estimates. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval (SIGIR 2003)*, pages 111–118, Toronto, CA, 2003.
7. M. Bianchi, M. Draoli, and G. Gambosi. Towards a SLA-aware classification framework. In *Proceedings of the 5th Italian Information Retrieval Workshop (IIR 2014)*, Roma, IT, 2014.
8. Christina Brandt, Thorsten Joachims, Yisong Yue, and Jacob Bank. Dynamic ranked retrieval. In *Proceedings of the 4th International Conference on Web Search and Web Data Mining (WSDM 2011)*, pages 247–256, Hong Kong, CN, 2011.
9. Klaus Brinker. On active learning in multi-label classification. In Myra Spiliopoulou, Rudolf Kruse, Christian Borgelt, Andreas Nürnberger, and Wolfgang Gaul, editors, *From Data and Information Analysis to Knowledge Engineering*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 206–213. Springer Berlin Heidelberg, 2006.
10. Prabir Burman. Smoothing sparse contingency tables. *The Indian Journal of Statistics*, 49(1):24–36, 1987.
11. James W. Carey, Mark Morgan, and Margaret J. Oxtoby. Intercoder agreement in analysis of responses to open-ended interview questions: Examples from tuberculosis research. *Cultural Anthropology Methods*, 8(3):1–5, 2006.
12. Ben Carterette, Evangelos Kanoulas, and Emine Yilmaz. Simulating simple user behavior for system effectiveness evaluation. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM 2011)*, pages 611–620, Glasgow, UK, 2011.

13. Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM 2009)*, pages 621–630, Hong Kong, CN, 2009.
14. Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics (ACL 1996)*, pages 310–318, Santa Cruz, US, 1996.
15. Gordon V. Cormack. Email spam filtering: A systematic review. *Foundations and Trends in Information Retrieval*, 1(4):335–455, 2008.
16. Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
17. Susan T. Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the 7th ACM International Conference on Information and Knowledge Management (CIKM 1998)*, pages 148–155, Bethesda, US, 1998.
18. Charles Elkan. The foundations of cost-sensitive learning. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 973–978, Seattle, US, 2001.
19. Andrea Esuli, Tiziano Fagni, and Fabrizio Sebastiani. MP-Boost: A multiple-pivot boosting algorithm and its application to text categorization. In *Proceedings of the 13th International Symposium on String Processing and Information Retrieval (SPIRE 2006)*, pages 1–12, Glasgow, UK, 2006.
20. Andrea Esuli and Fabrizio Sebastiani. Active learning strategies for multi-label text classification. In *Proceedings of the 31st European Conference on Information Retrieval (ECIR 2009)*, pages 102–113, Toulouse, FR, 2009.
21. Andrea Esuli and Fabrizio Sebastiani. Machines that learn how to code open-ended survey data. *International Journal of Market Research*, 52(6):775–800, 2010.
22. Andrea Esuli and Fabrizio Sebastiani. Training data cleaning for text classification. *ACM Transactions on Information Systems*, 31(4), 2013.
23. George Forman and Martin Scholz. Apples-to-apples in cross-validation studies: Pitfalls in classifier performance measurement. *SIGKDD Explorations*, 12(1):49–57, 2010.
24. Norbert Fuhr. A probability ranking principle for interactive information retrieval. *Information Retrieval*, 11(3):251–265, June 2008.
25. Fumiyo Fukumoto and Yoshimi Suzuki. Correcting category errors in text classification. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, pages 868–874, Geneva, CH, 2004.
26. Fumiyo Fukumoto, Takeshi Yamamoto, Suguru Matsuyoshi, and Yoshimi Suzuki. Text classification with relatively small positive documents and unlabeled data. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM 2012)*, pages 2315–2318, Maui, US, 2012.
27. William A. Gale and Kenneth W. Church. What's wrong with adding one? In N. Oostdijk and P. de Haan, editors, *Corpus-Based Research into Language: In honour of Jan Aarts*, pages 189–200. Rodopi, Amsterdam, NL, 1994.
28. Shantanu Godbole, Abhay Harpale, Sunita Sarawagi, and Soumen Chakrabarti. Document classification through interactive supervision of document and term labels. In *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2004)*, pages 185–196, Pisa, IT, 2004.
29. Maura R Grossman, Gordon V Cormack, Bruce Hedin, and Douglas W Oard. Overview of the trec 2011 legal track. In *The 20th Text REtrieval Conference (TREC 2011)*, Gaithersburg, US, 2011.
30. Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

31. William Hersh, Christopher Buckley, T.J. Leone, and David Hickman. OHSUMED: An interactive retrieval evaluation and new large text collection for research. In *Proceedings of the 17th ACM International Conference on Research and Development in Information Retrieval (SIGIR 1994)*, pages 192–201, Dublin, IE, 1994.
32. Steven C. Hoi, Rong Jin, and Michael R. Lyu. Large-scale text categorization by batch mode active learning. In *Proceedings of the 15th International Conference on World Wide Web (WWW 2006)*, pages 633–642, Edinburgh, UK, 2006.
33. Chen-Wei Hung and Hsuan-Tien Lin. Multi-label active learning with auxiliary learner. In *Proceedings of the 3rd Asian Conference on Machine Learning (ACML 2011)*, pages 315–332, Taoyuan, TW, 2011.
34. PanagiotisG. Ipeirotis, Foster Provost, VictorS. Sheng, and Jing Wang. Repeated labeling using multiple noisy labelers. *Data Mining and Knowledge Discovery*, 28(2):402–441, 2014.
35. Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems*, 20(4):422–446, October 2002.
36. Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning (ECML 1998)*, pages 137–142, Chemnitz, DE, 1998.
37. Thorsten Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999)*, pages 200–209, San Francisco, US, 1999.
38. Maryam Karimzadehgan and ChengXiang Zhai. Exploration-exploitation tradeoff in interactive relevance feedback. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM 2010)*, pages 1397–1400, Toronto, CA, 2010.
39. Gabriella Kazai. In search of quality in crowdsourcing for search engine evaluation. In *Proceedings of the 33rd European Conference on Advances in Information Retrieval (ECIR 2011)*, pages 165–176, Dublin, IE, 2011.
40. Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI 1995)*, pages 1137–1143, Montreal, CA, 1995.
41. Trausti Kristjansson, Aron Culotta, Paul Viola, and Andrew McCallum. Interactive information extraction with constrained conditional random fields. In *Proceedings of the 19th National Conference on Artifical Intelligence (AAAI 2004)*, pages 412–418, San Jose, US, 2004.
42. Ken Lang. Newsweeder: Learning to filter netnews. In *In Proceedings of the Twelfth International Conference on Machine Learning (ICML 1995)*, pages 331–339, 1995.
43. Leah S. Larkey and W. Bruce Croft. Combining classifiers in text categorization. In *Proceedings of the 19th ACM International Conference on Research and Development in Information Retrieval (SIGIR 1996)*, pages 289–297, Zürich, CH, 1996.
44. David D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *Proceedings of the 10th European Conference on Machine Learning (ECML 1998)*, pages 4–15, London, UK, 1998.
45. David D. Lewis. *Reuters-21578 text categorization test collection Distribution 1.0 README file (v 1.3)*, 2004.
46. David D. Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of 11th International Conference on Machine Learning (ICML 1994)*, pages 148–156, New Brunswick, US, 1994.
47. David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th ACM International Conference on Research and Development in Information Retrieval (SIGIR 1994)*, pages 3–12, Dublin, IE, 1994.

48. David D. Lewis, Robert E. Schapire, James P. Callan, and Ron Papka. Training algorithms for linear text classifiers. In *Proceedings of the 19th ACM International Conference on Research and Development in Information Retrieval (SIGIR 1996)*, pages 298–306, Zürich, CH, 1996.
49. David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.
50. X. Li, L. Wang, and E. Sung. Multilabel svm active learning for image classification. In *International Conference on Image Processing (ICIP 2004)*, pages 2207–2210, Singapore, 2004.
51. Xin Li and Yuhong Guo. Active learning with multi-label svm classification. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2013)*, Beijing, CN, 2013.
52. Bing Liu. *Sentiment Analysis and Opinion Mining*. Morgan and Claypool Publishers, San Rafael, US, 2012.
53. Tie-Yan Liu. *Learning to Rank for Information Retrieval*. Now Publishers Inc., Hanover, US, 2009.
54. Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK, 2008.
55. Dragos D. Margineantu. Active cost-sensitive learning. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, pages 1622–1623, Edinburgh, UK, 2005.
56. Miguel Martinez-Alvarez, Alejandro Bellogin, and Thomas Roelleke. Document difficulty framework for semi-automatic text classification. In *Proceedings of the 15th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2013)*, Prague, CZ, 2013.
57. Andrew K. McCallum and Kamal Nigam. Employing EM in pool-based active learning for text classification. In *Proceedings of the 15th International Conference on Machine Learning (ICML 1998)*, pages 350–358, Madison, US, 1998.
58. Alistair Moffat and Justin Zobel. Rank-biased precision for measurement of retrieval effectiveness. *ACM Transactions on Information Systems*, 27(1), 2008.
59. Ramesh Nallapati, Mihai Surdeanu, and Christopher Manning. Corractive learning: Learning from noisy data through human interaction. In *Proceedings of the IJCAI 2009 Workshop on Intelligence and Interaction*, Pasadena, US, 2009.
60. Alexandru Niculescu-Mizil and Rich Caruana. Obtaining calibrated probabilities from boosting. In *Proceedings of the 21st Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI 2005)*, pages 413–420, Arlington, US, 2005.
61. Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22th International Conference on Machine Learning (ICML 2005)*, pages 625–632, 2005.
62. Douglas W. Oard, Jason R. Baron, Bruce Hedin, David D. Lewis, and Stephen Tomlinson. Evaluation of information retrieval for E-discovery. *Artificial Intelligence and Law*, 18(4):347—386, 2010.
63. Douglas W. Oard and William Webber. Information retrieval for e-discovery. *Foundations and Trends in Information Retrieval*, 7, 2013.
64. John C. Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In Alexander Smola, Peter Bartlett, Bernard Schölkopf, and Dale Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 61–74. The MIT Press, Cambridge, MA, 2000.
65. Hema Raghavan, Omid Madani, and Rosie Jones. Active learning with feedback on features and instances. *Journal of Machine Learning Research*, 7:1655–1686, 2006.

66. Stephen E. Robertson. A new interpretation of average precision. In *Proceedings of the 31st ACM International Conference on Research and Development in Information Retrieval (SIGIR 2008)*, pages 689–690, Singapore, SN, 2008.
67. Stephen E Robertson and Ian Soboroff. The trec 2002 filtering track report. In *The Eleventh Text REtrieval Conference (TREC 2002)*, volume 2002, page 5, Gaithersburg, US, 2002.
68. J. J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, Prentice-Hall Series in Automatic Computation, chapter 14, pages 313–323. Prentice-Hall, Englewood Cliffs NJ, 1971.
69. Nicholas Roy and Andrew Mccallum. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the 18th International Conference on Machine Learning (ICML 2001)*, pages 441–448, Williamstown, US, 2001.
70. Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.
71. Robert Schapire and Yoram Singer. Improved boosting using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.
72. Robert E. Schapire and Yoram Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.
73. Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.
74. Burr Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012.
75. Pannaga Shivaswamy and Thorsten Joachims. Online structured prediction via coactive learning. *CoRR*, abs/1205.4213, 2012.
76. Jeffrey S. Simonoff. A penalty function approach to smoothing large sparse contingency tables. *The Annals of Statistics*, 11(1):208–218, 1983.
77. Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2001.
78. Cornelis J. van Rijsbergen. *Information Retrieval*. Butterworths, London, UK, second edition, 1979.
79. John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, US, 1944.
80. William Webber, Mossaab Bagdouri, David D. Lewis, and Douglas W. Oard. Sequential testing in classifier evaluation yields biased estimates of effectiveness. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2013)*, pages 933–936, Dublin, IE, 2013.
81. William Webber and Jeremy Pickens. Assessor disagreement and text classifier accuracy. In *Proceedings of the 36th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2013)*, pages 929–932, Dublin, IE, 2013.
82. Bishan Yang, Jian-Tao Sun, Tengjiao Wang, and Zheng Chen. Effective multi-label active learning for text classification. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD 2009)*, pages 917–926, New York, US, 2009.
83. Yiming Yang and Xin Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd ACM International Conference on Research and Development in Information Retrieval (SIGIR 1999)*, pages 42–49, Berkeley, US, 1999.
84. Emine Yilmaz, Milad Shokouhi, Nick Craswell, and Stephen Robertson. Expected browsing utility for web search evaluation. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM 2010)*, pages 1561–1564, Toronto, CN, 2010.

85. Bianca Zadrozny and Charles Elkan. Learning and making decisions when costs and probabilities are both unknown. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD 2001)*, pages 204–213, San Francisco, US, 2001.
86. Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the Eight ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD 2002)*, pages 694–699, Edmonton CA, 2002.
87. ChengXiang Zhai and John Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 22(2):179—214, 2004.
88. Tong Zhang and Frank J. Oles. Text categorization based on regularized linear classification methods. *Information Retrieval*, 4(1):5–31, April 2001.
89. Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML 2003 workshop on the continuum from labeled to unlabeled data in machine learning and data mining*, pages 58–65, Washington DC, US, 2003.

Acknowledgments

First of all, I would like to thank my advisors, Fabrizio Sebastiani, Andrea Esuli and Luca Simoncini, who have offered me all the support possible during these three years of challenging work. They have been able to put my uncertainties at ease and incite me to do my best.

I also thank all my colleagues at CNR, in particular Diego Marcheggiani, Tiziano Fagni and Alejandro Moreo Fernandez, who are excellent coworkers and mates, I can never get bored with them. I am also grateful to all the people I have worked with, as Fabrizio Silvestri and all the researchers in Pisa (and around the world) I have shared my experiences with.

A special thanks goes to my friends, all around Italy. To my friends in Pisa, thank you for the time spent together, climbing, travelling, and drinking. To my friends from Perugia, who are my big family, thanks for being close to me, no matter the distance.

I am very grateful to my family, because they have always been the main supporters in my life. Last but non least, a huge thanks to Cristina, for her help, but above all for the immense, lovely patience.