A THOROUGH EXPLOITATION OF DISTANCE-BASED META-FEATURES FOR AUTOMATED TEXT CLASSIFICATION

SERGIO CANUTO

A THOROUGH EXPLOITATION OF DISTANCE-BASED META-FEATURES FOR AUTOMATED TEXT CLASSIFICATION

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

ORIENTADOR: MARCOS ANDRÉ GONÇALVES
COORIENTADOR: THIERSON COUTO ROSA

Belo Horizonte - MG

Novembro de 2019

SERGIO CANUTO

A THOROUGH EXPLOITATION OF DISTANCE-BASED META-FEATURES FOR AUTOMATED TEXT CLASSIFICATION

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

ADVISOR: MARCOS ANDRÉ GONÇALVES
CO-ADVISOR: THIERSON COUTO ROSA

Belo Horizonte - MG
November 2019

Ficha catalográfica elaborada pela bibliotecária Belkiz Inez Rezende Costa
CRB 6ª Região nº 1510

Canuto, Sérgio Daniel Carvalho.

C235t A thorough exploitation of distance based meta-features for automated text classification/ Sérgio Daniel Carvalho Canuto — Belo Horizonte, 2019.
xxiii, 112 f. il.; 29 cm.

Tese (doutorado) - Universidade Federal de Minas Gerais – Departamento de Ciência da Computação;
Orientador: Marcos André Gonçalves.
Coorientador: Thierson Couto Rosa.

1. Computação – Teses. 2.Apredizado supervisionado. 3. Classificação de textos. 4.Meta caracteristicas. 5 Aprendizado de máquina – Teses. I. Orientador. II. Coorientador. III. Título.

CDU 519.6*82.10 (043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

A Thorough Exploitation of Distance-Based Meta-Features for Automated Text Classification

SÉRGIO DANIEL CARVALHO CANUTO

Tese defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. MARCOS ANDRÉ GONÇALVES - Orientador
Departamento de Ciência da Computação - UFMG

PROF. THIERSON COUTO ROSA - Coorientador
Instituto de Informática - UFG

PROFA. GISELE LOBO PAPPA
Departamento de Ciência da Computação - UFMG

PROF. RODRYGO LUIS TEODORO SANTOS
Departamento de Ciência da Computação - UFMG

PROF. PÁVEL PEREIRA CALADO
Departamento de Engenharia Informática - ULisboa

PROF. ALEXANDRE PLASTINO DE CARVALHO
Instituto de Computação - UFF

Belo Horizonte, 22 de Novembro de 2019.

Abstract

Automated Text Classification (ATC) has become substantially important for a variety of tasks, such as categorizing news, organizing digital libraries, building web directories, analyzing sentiment of user-generated content and detecting spam, to name a few. Given a set of training documents classified into one or more predefined categories, the task of ATC is to automatically learn how to classify new (unclassified) documents, using a combination of features of these documents that associates them with categories. Due to the fact that the ATC problem occurs in a number of different applications, diverse machine learning algorithms have been proposed to deal with ATC.

Although the classification algorithm itself plays an important role in ATC, the features that represent documents may be equally important to determine effectiveness. In particular, representing documents in a feature space is a prerequisite work for ATC, since these classification algorithms are designed to discover discriminative patterns on these features. In this sense, a relevant challenge relies on efficiently manipulating the feature space to address ATC from a data engineering viewpoint. In this context, we address the problem of automatically learning to classify texts by exploiting information derived from meta-features, i.e., features engineered from the original (bag-of-words) representation. Particularly, the exploited meta-features rely on distance measures to summarize complex relationships between documents and present discriminative information for classification.

We here not only propose new meta-features that provide discriminative evidence for classification, but also new mechanisms to analyze and select meta-features using multi-objective strategies. These strategies are capable of reducing the number of meta-features while maximizing the classification effectiveness, when considering the adequacy of the selected meta-features to a particular dataset or classification method. Moreover, we provide additional contributions to improve the efficiency and effectiveness of meta-features. Particularly, we propose: (i) the use of commodity GPUs to reduce the computational time to generate meta-features; (ii) the use of supervised learning to enrich distance relationships with labeled information; and (iii) the design of new specific meta-features for the sentiment analysis context.

Our experimental results on five traditional benchmarks for topic classification show that with the appropriate selection techniques, our distance-based meta-features can achieve remarkable classification results considering the results of original feature space and other recently proposed distance-based meta-features. We further explain our results with the identification and discussion about meta-features that, when combined, provide core information to classify documents. Our improvements on core meta-features using labeled information to enrich distance relationships provide additional gains over our best results in topic datasets. We also evaluate meta-features on nineteen sentiment analysis datasets. In this context, our proposals for sentiment classification produced remarkable results considering the effectiveness of previous meta-features that do not take sentiment analysis idiosyncrasies into account.

Keywords: supervised learning; text classification; meta-features; machine learning

Resumo

Classificação Automática de Texto (CAT) têm adquirido notória importância em uma variedade de tarefas, como a categorização de notícias, organização de bibliotecas digitais, criação de diretórios da web, análise de sentimentos em conteúdos gerados por usuários e detecção de spam. Dado um conjunto de documentos de treinamento classificados em uma ou mais categorias predefinidas, a tarefa do CAT é aprender automaticamente como classificar novos documentos (não classificados), usando uma combinação de atributos desses documentos que os associam a categorias. Devido ao fato de o problema do CAt ocorrer em vários contextos, diversos algoritmos de aprendizado de máquina foram propostos para lidar com CAT.

Embora o próprio algoritmo de classificação tenha um papel importante na CAT, os atributos que representam documentos podem ser igualmente importantes para determinar a eficácia da classificação. Especificamente, representar documentos em um espaço de atributos é um trabalho que precede a CAT, pois esses algoritmos de classificação são projetados para descobrir padrões discriminativos usando esses atributos. Nesse sentido, uma tarefa importante consiste em promover a manipulação espaço de atributos para abordar a CAT do ponto de vista da engenharia de dados. Nesse contexto, abordamos o problema de aprender a classificar textos de forma automática, explorando informações derivadas de meta-atributos, ou seja, atributos criados a partir da representação original dos documentos (bag of words). Particularmente, os meta-atributos explorados contam com medidas de distância capazes de sumarizar relacionamentos potencialmente complexos entre documentos e apresentar informações relevantes para classificação.

Neste trabalho, não apenas propomos novos meta-atributos que fornecem evidências discriminativas para classificação, mas também novos mecanismos para analisar e selecionar meta-atributos. Nesse sentido, utilizamos estratégias multiobjetivo capazes de minimizar o número de meta-atributos e maximizar a eficácia da classificação, considerando a adequação dos meta-atributos selecionados a uma coleção de dados ou método de classificação específico. Além disso, fornecemos contribuições adicionais para aprimorar a eficiência e a eficácia da utilização de meta-atributos. Em particular, propomos o uso de GPUs (Graphical Process-

ing Units) para reduzir o tempo computacional da geração de meta-atributos, o uso de aprendizado supervisionado para o enriquecimento dos relacionamentos de distância com dados rotulados, e a construção de novos meta-atributos específicos para o contexto da análise de sentimento.

Nossos resultados experimentais em cinco coleções tradicionalmente usadas na classificação em tópicos mostram que, com as técnicas de seleção apropriadas, nossos meta-atributos baseados em distância podem alcançar excelentes resultados de classificação considerando os resultados previamente obtidos no espaço de atributos original ou outros meta-atributos baseados em distância recentemente propostos. Além disso, avançamos nossa análise experimental com a identificação e discussão de meta-atributos que, quando combinados, fornecem informações centrais para a classificação de documentos. Aprimoramentos adicionais nesses meta-atributos a partir do enriquecimento dos relacionamentos de distância com informações de rotulação proporcionaram ganhos adicionais sobre nossos melhores resultados obtidos em coleções de classificação em tópicos. Também avaliamos meta-atributos em dezenove coleções de análise de sentimento. Nesse contexto, nossas propostas para classificação de sentimento apresentaram excelentes resultados quando comparados aos meta-atributos anteriores que não levam em consideração as idiossincrasias da tarefa de análise de sentimento.

Palavras-Chave: apredizado supervisionado; classificação de textos; meta características; aprendizado de máquina

List of Figures

List of Tables

List of Algorithms

Contents

Chapter 1

Introduction

The rapid growth of online information has led to an increasing need for handling and organizing textual data. Automatic Text Classification (ATC) plays a significant role in this scenario, due to a wide range of problems that can be solved with this approach, particularly when simple heuristics (that disregard data analysis) fail to provide effective results. In fact, ATC methods have become substantially important for a variety of tasks, such as categorizing news, organizing digital libraries, building web directories, analyzing sentiment of user-generated content and detecting spam, to name a few. In all such cases, there might be some discriminative patterns that relate a document to its respective category. For example, consider the problem of categorizing news articles. In this context, the occurrence of the word "democrats" in a document is a discriminative evidence that relates the document to the category politics. However, given the huge number of words and the potentially complex relationships among them, the task of manually describing such patterns may become infeasible.

To address such problems, ATC uses a set of training documents classified into one or more predefined categories, and automatically learns how to classify new (unclassified) documents, using a combination of features of these documents that associate them with categories (Sebastiani, 2002). More formally, a learning algorithm uses a given a set of pairs $(\vec{x}_i, c_i)$ known as a training set, where $\vec{x}_i \in \mathscr{X}$ denotes the vector representation of the document i as a point in a feature space $\mathscr{X}$ and $c_i \in \mathscr{C}$ a categorical attribute (or response variable) indicating i's class (C is a finite set composed of all the possible classes). The main goal of a supervised learning algorithm is to infer a classification function $f : \mathscr{X} \to \mathscr{C}$ from the training set, which can be used to map a new example $\vec{x}_n$ to its respective category c_n.

Although supervised learning techniques play a fundamental role in ATC, the features that represent documents may be equally important to determine classification effectiveness. In particular, representing documents in a feature space is a prerequisite work for supervised

classification methods, since these methods are designed to discover discriminative patterns based on these features. Usually, each dimension of the feature space $\mathscr{X} \subseteq \mathbb{R}^d$ corresponds to a word of the dictionary with d words. Therefore, a document can be seen as a point in this feature space, where each dimension corresponds to a word weighted according to its relative importance in the document (Hastie et al., 2003). In this sense, a trend that has emerged in ATC that works on the data engineering level instead of on the algorithmic level, is the introduction of meta-level features that can replace or work in conjunction with the original (bag-of-words-based) feature space (Canuto et al., 2015, 2014; Gopal and Yang, 2010; Kyriakopoulou and Kalamboukis, 2007, 2008; Pang et al., 2015; Raskutti et al., 2002; Yang and Gopal, 2012). These meta-features can capture insightful new information about the unknown underlying data distribution that relates the observed patterns with the associated category.

Particularly, we focus on distance-based meta-features, which rely on the fact that distance measures are capable of summarizing potentially complex relationships between text documents. This summarized information can then be exploited in different ways to generate robust and informative meta-features for text classification. For example, let us consider the meta-features driven from the projection of highly-dimensional documents into a low-dimensional space spanned by category centroids. More specifically, in this scenario, the meta-features are the distance scores between the document and each category centroid. This specific group of meta-features, taken from category centroids, is capable of mitigating issues related to imbalanced class distributions and irrelevant or noisy term features (Pang et al., 2015). Indeed recent works (Canuto et al., 2015, 2014, 2019, 2018; Pang et al., 2015; Yang and Gopal, 2012) represented discriminative patterns for text classification with several different groups of meta-features. These groups are manually designed to extract complementary information primarily from the distribution of distances between documents. Each group is designed to exploit a particular aspect, such as the similarity scores, the class distribution, the entropy, and the within-class cohesion observed in the k nearest neighbors of a given test document. The combination of these meta-features provides a more compact and informative feature space that reportedly improves classification effectiveness.

1.1 Problem Statement

In general terms, the problem we address in this dissertation can be defined as the task of transforming the original (bag-of-words) feature space $\mathscr{X}$ into a new meta-feature space $\mathscr{M}$ which is potentially smaller and more informative for text classification. Such meta-feature space is built upon the distances between an arbitrary document and training documents from

each particular category, providing explicit information about labeled data. Therefore, the problem consists in using the distances that summarize the relationship among documents from different categories as the primary source of information to build new meta-features for text classification.

More specifically, given a training set $\mathbb{T}_{train} = \{(\vec{x}_i, c_i) \in \mathscr{X} \times \mathscr{C}\}|_{i=1}^{m}$ and an arbitrary document $\vec{x}_n \in \mathscr{X}$, we define a meta-feature vector $\vec{m}_n \in \mathscr{M}$ for each document $\vec{x}_n \in \mathscr{X}$. The distance-based meta-features are dimensions of $\vec{m}_n$, which are driven primarily from the distances between a document $\vec{x}_n$ and training examples. The meta-feature vector $\vec{m}_n$ is divided into r sub-vectors $[m_n^1 m_n^2, ..., m_n^r]$, and each sub-vector is called a meta-feature group. Each meta-feature group is a manually defined sub-vector of meta-features that uses a particular strategy to exploits distances between documents and provides information for classification methods. In this scenario, the engineering of different meta-feature groups can be seen as different strategies that primarily exploit the contiguity hypothesis (Manning et al., 2008), which expects $\vec{x}_n$ to have the same label as the training documents located close to it.

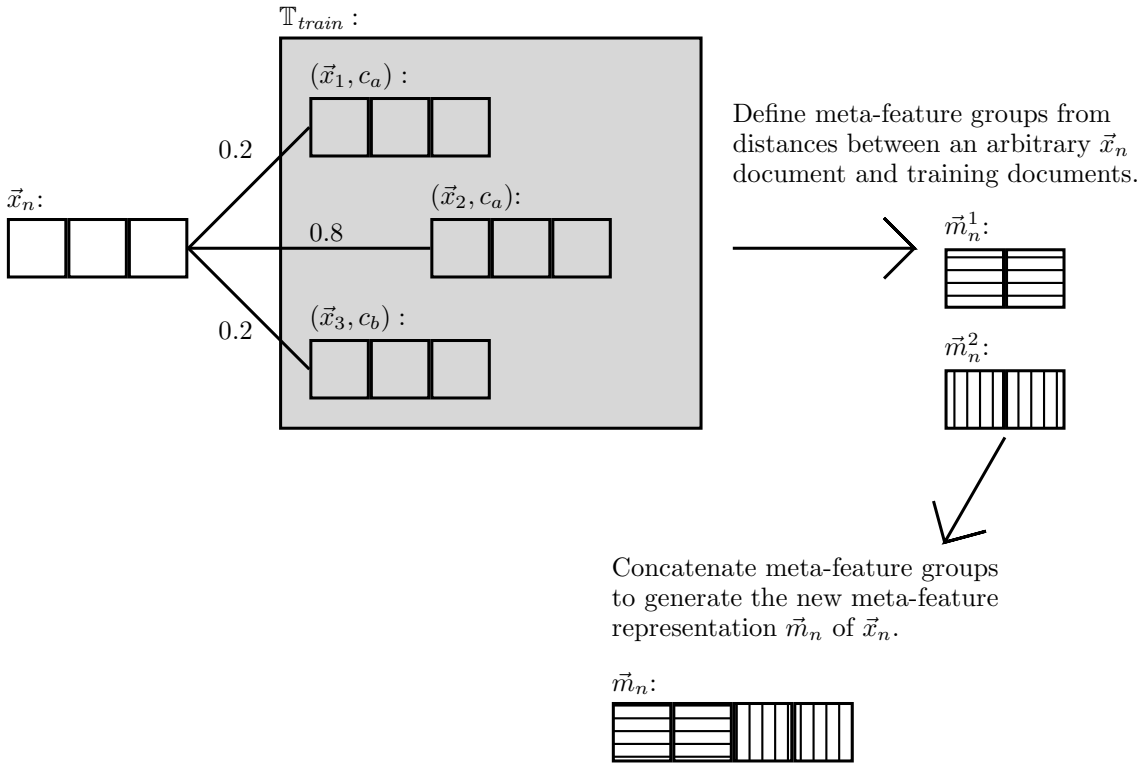

Figure 1.1: Generation of distance-based meta-features for the document x_n.

For example, lets consider the case illustrated in Figure 1.1 that presents the trans-

formation of an arbitrary document $\vec{x}_n$ into its new representation $\vec{m}_n$. The transformation uses the following training set $\mathbb{T}_{train}$ composed by three training documents i.e., $\mathbb{T}_{train} = \{(\vec{x}_1, c_a), (\vec{x}_2, c_a), (\vec{x}_3, c_b)\}$. The distances between $\vec{x}_n$ and the three training documents $\vec{x}_1$, $\vec{x}_2$ and $\vec{x}_3$ are 0.8, 0.2 and 0.2, respectively. First, we obtain the meta-feature groups for x_n, which represent statistics from distances between x_n and the training examples. In our toy example, lets consider two meta-feature groups m_n^1 and m_n^2. Assuming m_n^1 as the meta-feature group that represents the average distance from documents of the same category (e.g., the average distances between $\vec{x}_n$ and all documents categorized as c_a is $\frac{0.8+0.2}{2} = 0.5$), we obtain $m_n^1 = [0.5, 0.2]$. Then, we generate the meta-feature group m_n^2, which in our case is the distance between $\vec{x}_n$ and the nearest neighbor of each category (i.e., $m_n^2 = [0.2, 0.2]$). In this scenario the meta-feature representation of $\vec{x}_n$ is $\vec{m}_n = [m_n^1 m_n^2] = [0.5, 0.2, 0.2, 0.2]$.

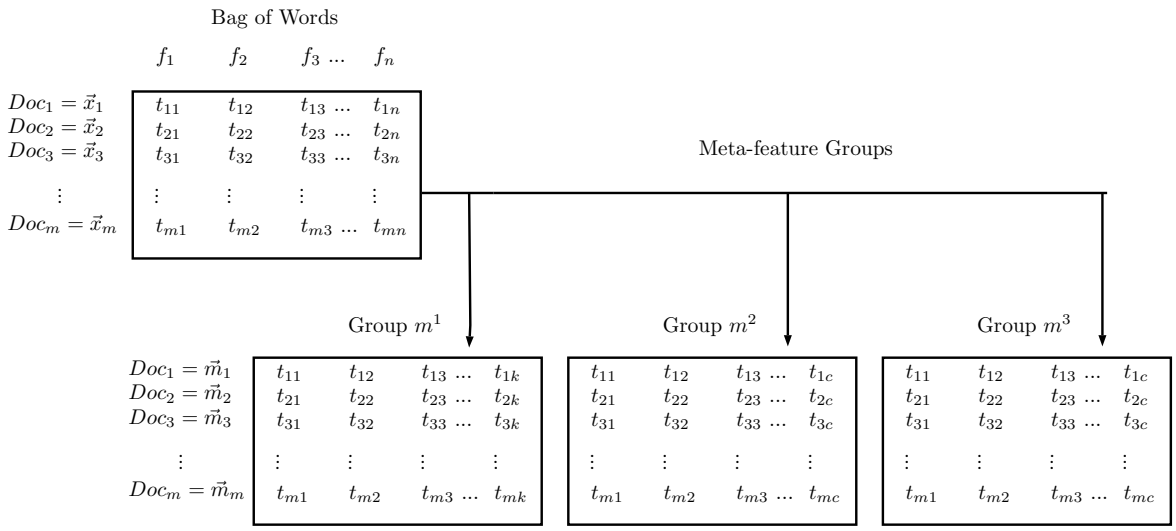

Figure 1.2: Transforming a dataset represented with bag-of-words into its respective meta-feature representation showed as the concatenation of meta-features from different groups.

Figure 1.2 presents the documents of a dataset represented as the traditional bag-of-words. In this example, after generating three meta-feature groups for each document, the new meta-feature representation can be seen as the concatenation of such groups.

Though discriminative by themselves, the combined use may sometimes degrade the effectiveness of classification algorithms (Canuto et al., 2014). In fact, the resulting meta-feature space may be unnecessarily complex and highly dimensional, increasing the tendency of overfitting of the classification models. Moreover, meta-feature groups may include noise in the meta-feature space because of specific characteristics of the dataset. For example, meta-features that exploit the neighborhood of documents on unbalanced data might incorrectly suggest the relationship between documents and the majority class, since the neighbors

of an arbitrary document tend to be from the majority class. Additionally, a large number of different meta-features might also be computationally expensive to generate, also increasing the computational costs to build classification models.

To use as few groups of meta-features as possible, it is necessary to consider the adequacy of the selected groups to a particular dataset or classification method, which is not trivial due to the high number of possible combinations. A related problem consists in finding a "common" reduced core of meta-features capable of providing effective results for different datasets. The solution to these problems can produce insights about the unknown behavior of different combinations of meta-feature groups, providing a starting point for new meta-features and their application in different contexts.

Throughout this dissertation, we shall study the effects of using different meta-feature groups for text documents considering the idiosyncrasies of each dataset. In fact, we provide methods that aim at solving the problems related to engineering efficient and effective distance-based meta-feature groups, as well as selecting and analyzing meta-features that improve the classification effectiveness on text categorization.

1.2 Dissertation Hypotheses and Proposals

In the following, we state the main hypotheses that we defend in this doctoral dissertation. Such hypotheses follow from the main idea that the exploitation of distances between documents can provide an informative document representation capable of improving document classification. Our proposals and empirical evidence that provide support for the hypotheses are discussed below in the form of answers for explicit research questions.

Hypothesis 1. *The use of distance-based meta-features can improve the classification effectiveness through the exploitation of discriminative patterns obtained from different statistics drawn from distances between documents.*

Despite the evidence for this hypothesis provided by previous works (Gopal and Yang, 2010; Pang et al., 2015; Yang and Gopal, 2012), the few distance-based meta-features previously proposed have not been extensively studied in multi-class ATC tasks. Specifically, they were not evaluated with the most traditional topic benchmarks adopted to compare algorithms for this task (e.g., 20NewsGroups, Reuters, WebKB, Medline, and ACM). In this work, we propose not only the use of such benchmarks, but also the use of new meta-features proposed in this dissertation, which exploit sophisticated statistics from distances between documents to improve classification effectiveness. We also propose additional experimental analyses to support our hypothesis by designing experiments that answer the following research questions:

RQ1 – How effective is the combination of the meta-feature groups proposed in different works? To answer this question, we propose to combine the meta-feature groups of different works and compare the effectiveness of the combination against the original proposals. As we shall see, in most situations, the combination of all meta-feature groups is more effective, showing their complementarity. This analysis not only supports our initial hypothesis, but also motivates our next research questions.

Hypothesis 2. *Multi-objective optimization techniques can be used as effective and efficient strategies for meta-feature selection and analysis.*

Despite their complementarity, the inclusion of a new group in the meta-feature space may introduce noise depending on the dataset characteristics (Canuto et al., 2014). We investigate this hypothesis by evaluating various feature selection mechanisms on meta-features, also considering an in-depth analysis of meta-feature selection on different classification methods using our proposed multi-objective methods. More precisely, we design experiments and methods guided by the following research questions:

Q2 – How effective and efficient are different strategies for feature selection on the meta-feature space?

As far as we know, there is no previous study that analyzes the potential of standard feature selection strategies for distance-based meta-features. The only preliminary study in this direction uses a greedy strategy to remove meta-feature groups that harm classification effectiveness (Canuto et al., 2014). In this work, we extend this analysis with selection strategies such as exhaustive search, best-first and genetic algorithms to search for effective combinations of meta-feature groups. Different from the previously mentioned methods, our proposed multi-objective genetic algorithms explicitly optimize the objectives that guides the algorithm to the most promising regions of the search space. Particularly, considering the objectives of maximizing the classification effectiveness and minimizing the number of meta-feature groups, our proposal can achieve remarkable classification effectiveness, with substantial meta-feature reduction by up to 89%.

Q3 – How effective are different classification strategies on distance-based meta-features considering meta-feature selection? Despite extensive use of the SVM classifier with meta-features, there is no comparative study that evaluates the effectiveness of different classification approaches on distance-based meta-features. We evaluate the effectiveness of different classification paradigms considering not only the combination of various meta-feature groups but also using our *meta-feature selection proposal (SPEA2fast),* which is capable of efficient selection considering multiple classification methods. SVM is consistently among

the best classification approaches, even considering potential improvements enjoined by the other methods after feature selection.

Hypothesis 3. *Despite the idiosyncrasies of each dataset, there is a "common set" of meta-feature groups capable of providing core information to classify documents over multiple datasets.*

In order to investigate this hypothesis, we propose to answer the following research question considering datasets both individually and collectively:

RQ4 –Which combinations of meta-feature groups provide the core information to classify documents? Each meta-feature group provides some sort of (potentially complementary) discriminative information. However, there might be combinations of just a few "core" meta-feature groups that are as effective as the combination of all of them, due to redundancy and noise. Thus, we propose to evaluate the conflicting objectives of maximizing the classification effectiveness while minimizing the number of meta-features groups to identify the most informative combinations.

Thus, we search for different combinations of meta-features groups that optimize the tradeoff between the two conflicting objectives. In this scenario, our meta-feature selection proposal (*SPEA2SVM*) aims at finding the Pareto frontier (Zitzler et al., 2001) of our objectives, instead of focusing only on finding the most effective combination of meta-features. This constitutes a major difference between our proposal and other traditional feature selection methods. Specifically, we focus our analysis on the individuals of the Pareto frontier. This frontier is constituted of a set of different combinations of meta-feature groups in which it is impossible to either include any other combination containing fewer groups or is more effective than any other combination in the Pareto frontier set. For instance, if a combination obtains low classification effectiveness but contains only a few meta-features groups, it cannot be discarded from the Pareto Frontier if there is no other combination that is both more effective and uses at least the same number of groups. Figure 1.3 presents an example of the Pareto frontier. The Pareto frontier provides evidence to evaluate good combinations while considering a restricted number of meta-feature groups.

In fact, we evaluate the specificities of the combinations in the Pareto frontier considering the characteristics of each dataset. For example, if we analyze the Pareto frontier of Figure 1.3, we can see that the combination using only four meta-feature groups achieves very effective results (close to the combination of all 7 groups).

Based on these results, we also analyze the core information present in this combination by verifying the characteristics of each meta-feature group present in the final selection,

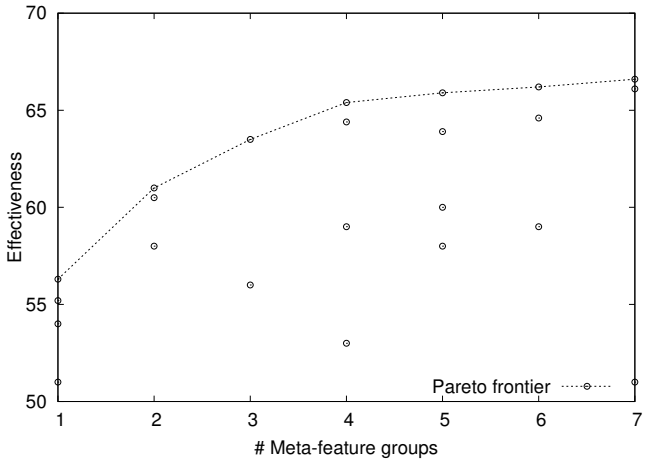

Figure 1.3: Representation of the Pareto frontier (points in the dashed line), where each point in the plane corresponds to a combination of meta-features.

considering the dataset specificities. Summarizing, the Pareto frontier provides evidence for an in-depth analysis of the discriminative information of different combinations.

Besides analyzing the core discriminative information considering each individual dataset, we also analyze specific combinations of meta-feature groups that provide the core information to classify documents accross various datasets. Given the objectives of minimizing the number of meta-feature groups and the maximization of the average effectiveness on the datasets as objectives, the proposed multi-objective optimization strategy searches for compact combinations that are effective in all datasets. As a result, we found that the combination of only two meta-feature groups provides most of the core discriminative information, and a specific combination of five meta-feature groups is capable of providing almost all the discriminative information to classify documents in our evaluations.

Hypothesis 4. *Enriching distance relationships with labeled information can improve the meta-feature space.*

Despite the previous success of distance-based meta-features, their underlying distance relationships rely on traditional distance measures among documents. These distances aim at summarizing discriminative evidence based on simple manipulations of term weights (such as TF-IDF), which might thwart the importance of relevant discriminative terms in the similarity computation. Also, distance measures such as Cosine, Euclidean and Manhattan are not designed to capture whether two documents belong to the same class and thus do not directly associate similarity with class information. In this context, we argue that enriching distance relationships with label information can improve the meta-feature space. Particularly, given a set of labeled training examples and similarity evidence (e.g., common words

among documents and similarity measures) that are manifested in distance relationships, we aim at answering the following research question:

RQ5 – Is it possible to improve the effectiveness of distance-based meta-features by enriching distance relationships with label information? To answer this question, we focus on extending the underlying distance relationships between a document and its neighbors with supervised strategies that evaluate the relevance of similarity evidence. Particularly, we propose meta-features capable of correlating a set of similarity evidence of a pair of documents with the likelihood of these documents belonging to the same class. We use these likelihoods, predicted with SVM hyperplane distances, to build new meta-features. We also estimate the level of error introduced by these newly proposed meta-features, specially for hard-to-classify regions of the meta-feature space, We do so by estimating the prediction errors in the neighborhood of each document. Whithout relying on feature selection strategies to improve the meta-feature space, this approach is able to exploit a deeper correlation of small pieces of evidence shared by documents and its labeled neighbors.

Hypothesis 5. *Meta-features can improve the classification effectiveness on other applications through the combined exploitation of context-specific information and different statistics drawn from distances between documents.*

In this dissertation, we propose to study the application of meta-features for the specific context of sentiment analysis. The classification of sentiments in short documents (i.e., short messages/reviews) poses new challenges for effective classification due to the shortage of information in small messages, the potentially limited number of training samples and noisy texts (e.g., inconsistent shortened words and slang terms) (Kiritchenko et al., 2014). These characteristics might impair the classification effectiveness and meta-feature representation. In this context, we evaluate the application of meta-features by answering the following research question:

RQ6 – How to exploit meta-features to provide effective results in sentiment analysis? To tackle the new challenges of sentiment analysis, we make use of BM25 (Manning et al., 2008) as similarity score in kNN, since it is an useful measure to rank documents with short messages as queries. We also exploit the neighborhood of a test example in both the training set and in a dataset containing 1.6 million tweets automatically labeled by its users with emoticons (Go et al., 2009). This methodology allows us to exploit discriminative information from different domains, even in noisy ones, like the large twitter dataset with emoticons. The last additional evidence we exploit is taken from the weighted sentiment polarity of the nearest neighbors by using lexicon-based methods to infer the message's

polarity towards a sentiment (Baccianella et al., 2010; Hutto and Gilbert, 2014; Thelwall et al., 2010).

Hypothesis 6. *Given the intrinsically paralyzable nature of the generation of distance-based meta-features, it is possible to considerably reduce the excecution time by taking advantage of the modern manycore GPU architectures.*

The prediction of documents using distance-based meta-features consumes a potentially high computational time, since it is necessary to find the neighborhood of each text document before the prediction task. In this scenario, we propose to exploit the intrinsically parallel nature of this problem (since distances between documents can be computed in parallel) by considering the high dimensional and sparse text data. More specifically, we take advantage of the current manycore GPU architecture and present a massively parallel strategy to compute distances between documents and find their closest neighbors. Our proposal aims at answering the following research question:

RQ7 – How to exploit the modern manycore GPU architectures to reduce the computational time to find the neighborhood of documents in text data?

Previously proposed methods to generate meta-features use serial CPU implementations to generate meta-features from text data. We propose to exploit inverted indexes along with an efficient implementation to cope with GPU memory limitations and compute distances efficiently.

1.3 Contributions

To summarize, the main contributions of this work are 6-fold:

- The proposition and thorough evaluation of new distance-based meta-features.
- The proposal of efficient and effective strategies to evaluate different combinations of meta-feature groups.
- The identification and discussion about the meta-feature groups that, when combined, provide the core information to classify documents.
- The proposal of effective strategies to enrich distance relationships with labeled data for meta-feature generation.
- The exploitation of meta-features specifically designed to take into account the idiosyncrasies of sentiment analysis.

- A GPU implementation of kNN for high dimensional and sparse data to reduce the computational time of predictions using distance-based meta-features.

1.4 Publications

The main results described in this dissertation are also presented in the following papers:

1. Canuto, S., Sousa, D., Gonçalves, M. A., Rosa, T. C. (2018). A Thorough Evaluation of Distance-Based Meta-Features for Automated Text Classification. *IEEE Transactions on Knowledge and Data Engineering*, 30(12):2242–2256. *(Qualis A1, h5-index: 77, IF. 3.85).*

2. Canuto, S., Salles, T., Couto, T., Gonçalves, M. A. (2019). Similarity-Based Synthetic Document Representations for Meta-Feature Generation in Text Classification. In *Proceedings of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, p. 355–364. *(Qualis A1, h5-index: 55).*

3. Canuto, S., Gonçalves, M. A., Benevenuto, F. (2016) Exploiting new sentiment-based meta-level features for effective sentiment analysis. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining – WSDM*, p. 53–62. *(Qualis A1, h5-index: 51).*

4. Canuto, S., Martins, W. S., Couto, T., Gonçalves, M. A. (2015). Efficient and Scalable MetaFeature-based Document Classification using Massively Parallel Computing. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, p. 333–342. *(Qualis A1, h5-index: 55).*

5. Canuto, S., Salles, T., Gonçalves, M. A., Rocha, L., Ramos, G., Gonçalves, G., Rosa, T., Martins W. (2014). On Efficient Meta-Level Features for Effective Text Classification. In *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management – CIKM*, p. 1709–1718. *(Qualis A1, h5-index: 48).*

6. Siqueira, G., Canuto, S., Gonçalves, M. A., Laender, A. F. (2017). Automatic Hierarchical Categorization of Research Expertise Using Minimum Information. In *Proceedings of the 21st International Conference on Theory and Practice of Digital Libraries – TPDL*. p. 103-115. *(Qualis B1, h5-index: 11).*

7. Sousa, D. X., Canuto, S., Rosa, T., Martins, W., Goncalves, M. A. (2016). Incorporating Risk-Sensitiveness into Feature Selection for Learning to Rank. In *Proceedings of*

the 25th ACM International on Conference on Information and Knowledge Management – CIKM. p. 257–266. *(Qualis A1, h5-index: 48).*

8. Sousa, D. X., Canuto, S., Gonçalves, M. A., Rosa, T., Martins, W. (2019). Risk-Sensitive Learning to Rank with Evolutionary Multi-Objective Feature Selection. ACM Transactions on Information Systems,37(2):1–34. *(Qualis A1, IF. 2.31, h5-index: 24).*

9. Viegas, F., Canuto S., Gomes, C., Luiz, W., Rosa, T., Ribas, S., Rocha, L., Gonçalves, M. A. (2019). CluWords: Exploiting Semantic Word Clustering Representation for Enhanced Topic Modeling. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining – WSDM*. p. 753-761. *(Qualis A1, h5-index: 51).*

(Canuto et al., 2014) presents evidence towards our first hypothesis with experiments using our proposed meta-feature groups and sheds light on the effects of selecting meta-feature groups with a simple meta-feature selection method. Our initial results motivated both our second and third hypotheses, which were thoroughly explored in (Canuto et al., 2018) with strategies to select, evaluate, and identify core meta-feature groups. (Canuto et al., 2019) exploits the core information provided by the similarity between a document and its neighbors using supervised learning to evaluate the similarity information, with experimental results that provide evidence to support our fourth hypothesis.

(Canuto et al., 2015) presents a method to efficiently generate meta-features using GPUs, with experimental results related to our sixth hypothesis. (Canuto et al., 2016) also takes advantage of GPUs to make the generation of meta-features from big sentiment analysis datasets feasible. Moreover, this publication presents additional evidence towards our fifth hypothesis by the application of meta-features designed for the context of sentiment analysis in short texts.

The remaining mentioned publications are works developed toegther with other members of the LBD (Laboratório de Banco de Dados) that use elements of this dissertation. In more details, (de Siqueira et al., 2017) proposes the application of distance-based meta-features in the context of categorizing research expertise and (Sousa et al., 2019, 2016) proposed the use of similar genetic algorithms presented in this dissertation to solve related multi-objective machine learning problems, taking inspiration in our solutions. Finally, (Viegas et al., 2019) takes inspiration on some ideas of this dissertation to exploit distances among word representations for topic modeling.

These additional results provide even more evidence towards the potential benefits of our proposals.

1.5 Outline of this Dissertation

The rest of this dissertation is organized as follows. In Chapter 2 we discuss related work on distance-based meta-features. Chapter 3 describes both, the proposed and the literature meta-features that explicitly exploit traditional distance measures. Our proposed strategies to evaluate and select such meta-feature provide empirical evidence for RQ1, RQ2, RQ3, and RQ4. Chapter 4 exploits new meta-features based on supervised strategies to evaluate similarity evidence, with experiments that support our hypothesis 4 and answer RQ5. Chapter 5 presents our strategies based on meta-features for sentiment analysis, helping to answer RQ6. Chapter 6 presents our parallel kNN implementation to generate meta-features efficiently, answering RQ7. Finally, in Chapter 8 we conclude this dissertation with a summarization of our main findings and propose some directions for further investigation.

Chapter 2

Related Work

In this chapter, we review work related to distance-based meta-features. First, we present previous work proposed in the literature and how our contributions advance the state-of-the-art for text classification. Next, we review sentiment analysis works that relate to our proposed meta-features. Finally, we discuss work related to GPU-based kNN parallelism that can potentially help in meta-feature generation.

2.1 Meta-features

Several meta-level features have been proposed in order to improve the effectiveness of machine learning methods. They can be derived from clustering methods (Kyriakopoulou and Kalamboukis, 2006, 2007, 2008; Raskutti et al., 2002), from kNN (Canuto et al., 2016, 2015, 2014, 2018; Gopal and Yang, 2010; Yang and Gopal, 2012), or from category centroids (Kim et al., 2005; Pang et al., 2015).

The use of clustering (Kyriakopoulou and Kalamboukis, 2006, 2007, 2008; Raskutti et al., 2002) is among the earliest strategies to generate distance-based meta-features. Particularly, such strategies use the idea of aggregating the information of similar documents on clusters using both labeled and unlabeled data (Kyriakopoulou and Kalamboukis, 2007, 2008; Raskutti et al., 2002). These clusters produce meta-features that indicate the similarity of each example to potentially informative groups of documents. In (Raskutti et al., 2002) the largest n clusters are chosen as the most informative ones. Each cluster c contributes with a set of meta-features, for example, an indicator function that captures whether c is the closest of the n clusters to the example or not; the similarity of the example to the cluster's centroid, among others. In (Kyriakopoulou and Kalamboukis, 2006, 2007, 2008) the number of clusters is chosen to be equal to the predefined number of classes and each cluster corresponds to an additional meta-feature.

More recently, several works (Canuto et al., 2016, 2015, 2014, 2018; Gopal and Yang, 2010; Pang et al., 2015; Yang and Gopal, 2012) proposed to use class centroids or kNN as the main tools to generate meta-features. They differ from the previously described cluster-based meta-features as they partition the training set for each class and exploit statistics from each training subset, explicitly capturing labeling information from training data. In more details, Gopal and Yang (2010) reported good results by designing meta-features that make a combined use of local information (through kNN-based features) and global information (through category centroids) in the training set. This work was extended by the same authors (Yang and Gopal, 2012) by leveraging successful learning-to-rank retrieval algorithms over the meta-feature space for the multi-label classification problem. The same strategy was successfully exploited on the problem of hierarchical categorization of research expertise using minimum information (de Siqueira et al., 2017).

Despite substantial improvements, the experiments of the previously described works (Gopal and Yang, 2010; Yang and Gopal, 2012) are limited to small datasets, since it is computationally expensive to generate meta-features for textual data. This happens because, to find the neighborhood of a given document, there is the need to constantly call the kNN algorithm. Particularly, they generate meta-features using a general purpose kNN implementation[1], which does not take into account the high dimensional and sparse characteristics of text data. Therefore, to make the meta-feature generation feasible for massive datasets, we propose our own parallel version of the kNN algorithm for highly dimensional and sparse text datasets (Canuto et al., 2015). This implementation enabled the application of meta-features on large datasets (Canuto et al., 2016).

Without using any additional special-purpose hardware, an alternative strategy to efficiently generate meta-features for highly dimensional and sparse data was the proposal of a compact meta-feature space derived only from category centroids (Kim et al., 2005; Pang et al., 2015). The use of these meta-features brings benefits in terms of both efficiency and classification effectiveness. This is especially the case for highly unbalanced datasets, since minority classes are not underrepresented by class centroids Pang et al. (2015). Moreover, Kim et al. (2005) experiments with centroid-based meta-features were capable of preserving the class structure from the full dimensional representation, achieving more effective and efficient results than other strategies aimed at dimension reduction, such as LSI (Latent Semantic Indexing) and GSVD (Generalized Singular Value Decomposition).

All the distance-based meta-features proposed in the previously discussed works (Canuto et al., 2015; Gopal and Yang, 2010; Pang et al., 2015; Yang and Gopal, 2012) are based on similarity scores (e.g., similarity between a document and a category

[1] `http://www.cs.umd.edu/~mount/ANN/`

centroid). We recently proposed the use of more elaborated statistics beyond the raw similarity scores (Canuto et al., 2014). In that work, different meta-feature groups were derived from the information provided by class distribution, the entropy and the within-class cohesion observed in the k nearest neighbors of a given test document x. The exploitation of various new statistics relied on a simple greedy feature selection strategy for removing meta-features that do not improve the classification effectiveness. In a follow-up, (Canuto et al., 2018) advanced the analysis and selection of these meta-feature groups exploiting the SPEA2 (Zitzler et al., 2001) genetic algorithm, which were successfully applied on related machine learning tasks (Dalip et al., 2014; Li et al., 2015; Sousa et al., 2016). Our extensions of the SPEA2 algorithm not only provided effective selection of meta-features for a particular dataset, but also enabled a thorough evaluation of core information exploited by the meta-features, as described in Chapter 3.

Without relying on feature selection, our most recent works (Canuto et al., 2016, 2019) focused on extending the exploitation of core meta-features that achieved the best effectiveness in most datasets. Particularly, in (Canuto et al., 2019) we focus on extending underlying cosine similarity relationships between a document and its neighbors with supervised strategies that evaluate the relevance of similarity information. Finally, in (Canuto et al., 2016), we enrich meta-features considering sentiment analysis idiosyncrasies, as described in the next section.

2.2 Sentiment Analysis

Unlike topic classification, which traditionally aims at classifying documents into topics, current sentiment extraction tasks differ according to the types of classes predicted (positive or negative, subjective or objective), the classification techniques used, and the considered classification levels, which comprehend sentence, aspect or document level (Feldman, 2013). Each level depends on the granularity of the sentiment scope, which may be predicted for whole documents (document-level), for individual sentences (sentence-level) or for specific aspects of entities and its properties (aspect-level). In this dissertation, we focus on the sentiment detection of messages composed of one or a few short sentences, such as tweets, comments and (micro-)reviews.

Regarding the document classes, there is an active research on subjectivity classification (Banea et al., 2008; Biyani et al., 2014; Su and Markert, 2008; Wiebe and Riloff, 2005), in which the goal is to discriminate objective messages from subjective ones. Some works separate subjective sentences from objective ones and then classify the polarity (positive or negative) of the opinions expressed in the subjective sentences (Barbosa and Feng,

2010; Pang and Lee, 2004; Wang et al., 2015). These efforts present empirical evidence that performing a simultaneous classification between positive, negative and objective (neutral) messages is worst than a two-step approach that filters the objective messages first.

Regarding the classification techniques, most methods can be categorized into one of two main groups: supervised approaches (Gonçalves et al., 2016), which use a wide range of features and labeled data for training sentiment classifiers, and lexicon-based approaches (Baccianella et al., 2010; Hutto and Gilbert, 2014; Thelwall et al., 2010), which make use of pre-built lexicons of words weighted with their sentiment orientations to determine the overall sentiment of a given document. Specifically, recent methods such as Vader (Hutto and Gilbert, 2014) and SentiStrength (Thelwall et al., 2010) rely on linguistic clues (e.g., punctuation and words that intensify or invert polarity) and on a dictionary of words provided with their sentiment scores, manually labeled [2]. Their dictionaries differ, since Vader's lexicons include most internet slang terms and SentiStrength rely on the combination of diverse available lexicon dictionaries. They also differ on their scoring scheme, since SentiStrength (Thelwall et al., 2010) weights the sentiment of a message by attributing to the sentiment score of the most positively (or negatively) classified word in the message and Vader sums the scores in of the words in the message. Instead of relying on a dictionary of lexicons, SentiWordNet (Baccianella et al., 2010) explores relationships between unambiguous polarized (or neutral) "concepts" in order to find the average polarity of close concepts associated to a message. Therefore, terms like dog and bite (both representing neutral concepts in SentiWordNet) appearing in the same message could eventually be expanded with a more emotional term like hurt, which holds a negative polarity. We choose to use these three lexicons as the basis to construct our meta-level features as they showed to be effective for sentiment detection in recent benchmark comparison studies (Araújo et al., 2014; Gonçalves et al., 2013; Ribeiro et al., 2016).

Overall, unlike the above efforts, most supervised approaches are direct applications of the traditional classification techniques. Thus, to the best of our knowledge, our effort described in details on Chapter 5 is the first to propose specific meta-level features for text classification in the sentiment analysis domain.

Since such meta-features are based on the neighborhood of documents, they can take advantage of parallel kNN implementations, discussed next.

[2] the sentiment of each word was manually adjusted. In Vader, for example, each word was manually scored with 9 degrees of valence (between extremely positive to extremely negative).

2.3 GPU-based kNN

Recently, some proposals have been presented to accelerate the kNN algorithm via highly mutithreaded GPU-based approaches (Garcia et al., 2008; Kuang and Zhao, 2009; Sismanis et al., 2012). However, they do not propose data structures capable of taking advantage of highly dimensional and sparse text datasets.

The first, and most cited, GPU-based kNN implementation was proposed by (Garcia et al., 2008). They used the brute force approach and reported speedups of up to two orders of magnitude when compared to a brute force CPU-based implementation, which is expected due to hundreds of cores in the GPU hardware. Their implementation assumes that multiple queries are performed. They also compute and store a complete distance matrix, what makes their approach impracticable for large datasets (over 65,536 documents).

Following works (Garcia et al., 2008) and (Kuang and Zhao, 2009) implemented their own optimized matrix operations for calculating distances, and used radix sort to find the top-k elements. Liang et al. (2009) took advantage of CUDA Streams to overlap computation and communication (CPU/GPU) when dealing with several queries, decreasing GPU memory requirements. The distances were computed in blocks and later merged first locally and then globally to find the top-k elements. However, such works can still be considered brute-force. (Sismanis et al., 2012) concentrated on the sorting phase of the brute-force kNN and provided an extensive comparison among parallel truncated sorts. They conclude that the truncated bitonic sort (TBiS) produces the best results.

Our proposal, described in Chapter 6.1, differs from the above mentioned work in many aspects. First, it exploits a very efficient GPU implementation of inverted indexes which supports an exact kNN solution without relying on brute-force. This also allows our solution to save memory space since the inverted index corresponds to a sparse representation of the data. In the distance calculation step, we resort to a smart load balancing among threads to increase the parallelism. And in the sorting step, we exploit a GPU-based sorting procedure, which was shown to be superior to other partial sorting algorithms (Sismanis et al., 2012), in combination with a CPU merge operation based on a priority queue. This implementation enabled the application of meta-features on large datasets for topic classification (Canuto et al., 2015) and sentiment analysis tasks (Canuto et al., 2016).

Most of the recently proposed distance-based meta-feature groups rely on our fast kNN implementation to find the neighborhood of documents efficiently. In the next chapter, we provide a detailed description of such meta-features.

Chapter 3

Meta-Features: Proposition, Evaluation and Selection

In this chapter, we present meta-features previously proposed in literature as well as our new proposals (Canuto et al., 2014). We analyze combinations of meta-feature groups that provide the core information to classify documents by exploiting traditional distance measures such as Euclidean, Manhattan, and the Cosine similarity. We start by providing a detailed description of each meta-feature group in Section 3.1. Next (Section 3.2), we describe the used meta-feature selection methods showing that multi-objective methods are not only capable of selecting effective meta-features, but also evaluating core combinations of groups. Finally, in Section 3.3, we describe our experimental results using the previously mentioned meta-features and meta-feature selection strategies .

3.1 Meta-Features Based on Traditional Distance Measures

Let $\mathscr{X}$ and $\mathscr{C}$ denote the input (feature) and output (class) spaces, respectively. Let $\mathbb{D}_{train} = \{(x_i, c_i) \in \mathscr{X} \times \mathscr{C}\}|_{i=1}^{n}$ be the training set. Recall that the main goal of supervised classification is to learn a mapping function $h : \mathscr{X} \mapsto \mathscr{C}$ which is general enough to accurately classify examples $x' \notin \mathbb{D}_{train}$.

The distance-based meta-features proposed in this dissertation, as well as the already existing ones (from the literature) (Gopal and Yang, 2010; Pang et al., 2015), are designed to replace the original input space $\mathscr{X}$ with a new informative and compact one $\mathscr{M}$.

A vector of meta-level features $m_f \in \mathscr{M}$ is expressed as the concatenation of some of the sub-vectors below, which are defined for each example $x_f \in \mathscr{X}$ and category $c_j \in \mathscr{C}$ for

$j = 1, 2, \ldots, |\mathscr{C}|$ as:

- $\vec{v}_{\vec{x}_f}^{cos} = [\cos(\vec{x}_{ij}, \vec{x}_f)]$: A k-dimensional vector produced by considering the k nearest neighbors of class c_j to the target vector x_f. More specifically, $\vec{x}_{ij}$ is the i^{th} ($i \leq k$) nearest neighbor to $\vec{x}_f$, and $\cos(\vec{x}_{ij}, \vec{x}_f)$ is the cosine similarity between them. Thus, a k-dimensional meta-level feature vector is generated for each class, given a document x_f.

- $\vec{v}_{\vec{x}_f}^{L_1} = [d_1(\vec{x}_{ij}, \vec{x}_f)]$: A k-dimensional vector whose elements $d_1(\vec{x}_{ij}, \vec{x}_f)$ denote the L_1 distance between $\vec{x}_f$ and the i^{th} nearest class c_j neighbor of $\vec{x}_f$ (i.e., $d_1(\vec{x}_{ij}, \vec{x}_f) = ||\vec{x}_{ij} - \vec{x}_f||_1$).

- $\vec{v}_{\vec{x}_f}^{L_2} = [d_2(\vec{x}_{ij}, \vec{x}_f)]$: A k-dimensional vector whose elements $d_2(\vec{x}_{ij}, \vec{x}_f)$ denote the L_2 distance between $\vec{x}_f$ and the i^{th} nearest class c_j neighbor of $\vec{x}_f$ (i.e., $d_2(\vec{x}_{ij}, \vec{x}_f) = ||\vec{x}_{ij} - \vec{x}_f||_2$).

- $\vec{v}_{\vec{x}_f}^{cos\_cent} = [cos(\vec{x}_j, \vec{x}_f)]$: A 1-dimensional vector where $\vec{x}_j$ is the category c_j centroid (i.e., vector average of all training examples of the class c_j).Thus, for each category c_j there is a corresponding meta-feature which its value is the raw cosine similarity score between the document x_f and the category c_j centroid.

- $\vec{v}_{\vec{x}_f}^{L_2\_cent} = [d_2(\vec{x}_j, \vec{x}_f)]$: A 1-dimensional vector where $\vec{x}_j$ is the category c_j centroid. This vector contains the L_2 distance between the document x_f and the category c_j centroid.

- $\vec{v}_{\vec{x}_f}^{cnt} = \left[n_j\right]$: A vector of dimension 1 produced by counting the number n_j of neighbors (among the k neighbors) of $\vec{x}_f$ which are positive training examples in category c_j.

- $\vec{v}_{\vec{x}_f}^{mcnt} = \left[\frac{s_j}{s_{max}}\right]$: A vector of dimension 1 produced by the sum s_j of the neighbor's cosine similarity (among the k neighbors) of $\vec{x}_f$ which are positive training examples in category c_j. Such a sum is then normalized by s_{max}, which corresponds to the largest s_j among all categories.

- $\vec{v}_{\vec{x}_f}^{qrt} = \left[cos(\vec{x}_{ej}, \vec{x}_f)\right]$: A vector of dimension 5 produced by considering five points that characterize the distribution of similarities of $\vec{x}_f$ to its category j neighbors x_{ij} ($1 \leq i \leq k$), where k is the neighborhood size, as measured by the cosine similarity between both. Among all computed similarity values, we pick five representative points: the lowest similarity, the highest similarity, the median similarity, the lower quartile (that splits the lowest 25% of points) and the upper quartile (that splits the highest 25% of points).

- $\vec{v}_{\vec{x}_f}^{cqrt} = \left[cos(q_{ej}, c_{ej})\right]$: A vector of dimension 5 containing the similarity value ($cos(q_{ej}, c_{ej})$ between each previously described quartile q_{ej} and the centroid c_{ej} computed among quartiles of the same kind of q_{ej} but obtained from the training examples of class c_j. For example, suppose the quartile q_{ej} is the lowest similarity quartile regarding $\vec{x}_f$. We obtain the lowest similarity quartile regarding each training element of class c_j and

obtain the centroid c_{ej} of these quartiles by computing the average quartile among them. Next, we compute the cosine similarity between q_{ej} and c_{ej}. We repeat this process for the four other quartiles, obtaining the 5-dimensional vector.

- $\vec{v}_{\vec{x}_f}^{sum\_cent} = \left[\frac{1}{|Neig_{c_j}(\vec{x}_f)|}\sum cos(\vec{x}_{ej},\vec{x}_j)\right]$: A vector of dimension $|C|$ containing the mean similarity between the class c_j centroid $\vec{x}_j$ and all the $\vec{x_{ej}} \in Neig_{c_j}(x_f)$, where $Neig_{c_j}(x_f)$ is the set of neighbors (regarding the cosine measure) of the $\vec{x_f}$ belonging to category c_j.

- $\vec{v}_{\vec{x}_f}^{IG} = \left[IG(cos(\vec{x}_{ij},\vec{x_f})), split, cos(\vec{x}_{split},\vec{x_f})\right]$: According to the Information Gain (Breiman et al., 1984) measure, a 3 - dimensional vector is produced, considering the point which best splits the neighbors of $\vec{x_f}$ belonging to category c_j. Specifically, these neighbors are ranked according to their similarity to $\vec{x_f}$ and, similarly to decision trees' splitting step, we select the point that better splits the ranking into two parts. The best split point is the one that maximizes the information gain for the class c_j. For example, suppose $\vec{x_f}$ belongs to category c_j. It is expected, in this case, that the top ranked neighbors documents also belong to c_j, and the best splitting point to be selected is the least similar neighbor of $\vec{x_f}$ which belongs to c_j (with the low ranked neighbors probably not belonging to c_j)—a (quasi-)zero-entropy case. The three meta-features selected in this process are: (i) the maximized information gain value, (ii) the position of the split point in the ranking and (iii) the similarity between the splitting point and $\vec{x_f}$.

- $\vec{v}_{\vec{x}_f}^{fisher} = \left[\frac{|\mu_i - \mu_j|}{\sqrt{\mathbb{V}_i + \mathbb{V}_j}}\right]$: A 1-dimensional vector that captures the correlation between the neighbors of $\vec{x_f}$ which belong to category c_j and the remaining neighbors that do not belong to c_j, considering a neighborhood of size k. Let μ_j be the mean similarity of the positive neighborhood (i.e., those belonging to c_j), given by $\frac{1}{k}\sum cos(\vec{x}_{ij},\vec{x_f})$, and μ_i the mean similarity of the negative neighborhood (i.e., the neighbors not in c_j). Also, let $\mathbb{V}_j$ and $\mathbb{V}_i$ be the variances computed from each positive and negative similarities, respectively. The Fisher's correlation measure (Fisher, 1925) is then computed, as $\frac{|\mu_i - \mu_j|}{\sqrt{\mathbb{V}_i + \mathbb{V}_j}}$.

Table 3.1 presents the names given to each group of meta-features and the authors who proposed them. More specifically, the most compact meta-feature space was proposed by Pang et al. (2015). It transforms the feature space using only the cosine similarity scores between examples and category centroids as meta-features. In fact, the proposed *cos_cent* group contains only $|\mathscr{C}|$ meta-features, as illustrated in Figure 3.1.

A different meta-feature space was proposed by Gopal and Yang (2010) by using *cos_cent* along with four other groups of meta-features. This new meta-feature space is based only on distance and similarity scores that make a combined use of local (through *cos_knn*, *l*1*_knn* and *l*2*_knn*) and global information (through *l*2*_cent* and *cos_cent*). More

Group Name	Sub-Vector	Reference
cos_knn	$\vec{v}_{\vec{x}_f}^{cos}$	Gopal and Yang (2010)
cos_cent	$\vec{v}_{\vec{x}_f}^{cos\_cent}$	Gopal and Yang (2010) and Pang et al. (2015)
l1_knn	$\vec{v}_{\vec{x}_f}^{L_1}$	Gopal and Yang (2010)
l2_knn	$\vec{v}_{\vec{x}_f}^{L_2}$	Gopal and Yang (2010)
l2_cent	$\vec{v}_{\vec{x}_f}^{L_2\_cent}$	Gopal and Yang (2010)
cnt	$\vec{v}_{\vec{x}_f}^{cnt}$	Canuto et al. (2014)
ncnt	$\vec{v}_{\vec{x}_f}^{ncnt}$	Canuto et al. (2014)
qrt	$\vec{v}_{\vec{x}_f}^{qrt}$	Canuto et al. (2014)
fisher	$\vec{v}_{\vec{x}_f}^{fisher}$	Canuto et al. (2014)
ig	$\vec{v}_{\vec{x}_f}^{IG}$	Canuto et al. (2014)
cqrt	$\vec{v}_{\vec{x}_f}^{cqrt}$	Canuto et al. (2014)
sum_cent	$\vec{v}_{\vec{x}_f}^{sum\_cent}$	Canuto et al. (2014)

Table 3.1: Given name for each group of meta-feature.

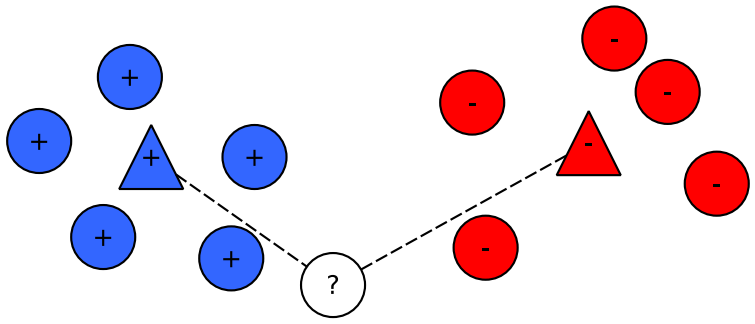

Figure 3.1: The dashed lines represent cos_cent meta-features generated for the (white circle) document. Each meta-feature is the cosine similarity between the document and a category centroid (triangle).

specifically, each test example is directly compared to a set of nearest labeled examples and category centroids, which are assumed to be enough to effectively characterize and discriminate categories, as illustrated in Figure 3.2. The intuition behind these meta-features consists in the assumption that if the distances between an example to the nearest neighbors belonging to the category c (and its corresponding centroid) are small, then the example is likely to belong to c.

Our meta-features (Canuto et al., 2014) exploit the neighborhood of a document extracting sophisticated statistics from them, instead of simply using the similarity scores as source of information. Specifically, the group of meta-features *sum_cent* consider the prox-

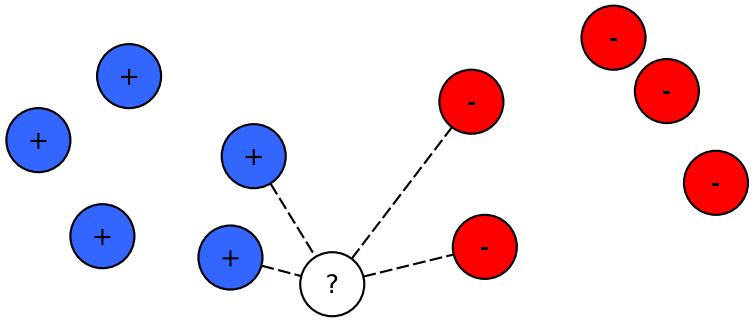

Figure 3.2: The dashed lines represent cos_knn meta-features generated for the (white circle) document. Each meta-feature is the cosine similarity between the document and its nearest neighbors from each category.

imities of the neighbors of a test example belonging to some class c_i to the centroid of c_j. This directly evaluates the class cohesion in the neighborhood of a test example, being an important information regarding the uncertainty level in such region of the input space, as illustrated in Figure 3.3.

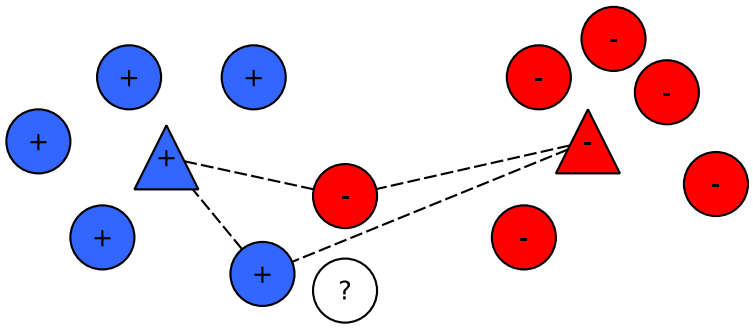

Figure 3.3: Given only the two nearest neighbors of the white circle, the dashed lines between the centroids and these neighbors represent the proposed sum_cent meta-features.

The group of meta-features *cnt* exploits the continuity hypothesis which guarantees the kNN classifier's success: the existence of a mode in the class distribution of the neighborhood of a test example usually determines its category. The neighborhood of a test example is further exploited by the *ncnt* group, which includes the information provided by the similarity scores and their relationship with the majority class.

The raw similarity scores between a test example and the k nearest neighbors from each class are explored by the *qrt* meta-feature group, which considers the three quartiles along with the minimum and maximum similarity value, exploring the within-class cohesion tied together with the locality information. The use of quartiles instead of the full distribution

(exploited by *cos_knn*) reduces considerably the number of dimensions, preventing overfitting in small datasets. These quartiles (together with min-max values) are further explored by the *cqrt* meta-feature group, by evaluating their deviance to the expected values, which provides global pieces of information related to each class.

Another key aspect exploited by these meta-features refers to the entropy observed in the neighborhood of a test example. The *IG* meta-feature group exploit the boundary, in the neighborhood of a test example, which maximizes the information gain of class c_j, similarly to the splitting strategy of decision trees. The situations where the splitting point is low ranked among the nearest neighbors highlight low entropy scenarios where the majority of the most similar examples belong to the same class. Finally, the $fisher$ group assesses the correlation between the similarities among the positive examples (those in c_j) and the similarities among the negative examples (those not in c_j) inside the neighborhood of $\vec{x_f}$. Again, low correlation values underline low entropy regions in the neighborhood.

Despite the previous efforts to design effective groups of meta-features, some combinations of meta-feature groups produce complex and high dimensional feature spaces, which induces overfitting on classification approaches. The next section presents adaptive strategies capable of selecting effective combinations of meta-features for distinct datasets.

3.2 Meta-feature Selection Strategies

3.2.1 Brute-Force

The most simple but computationally expensive strategy to evaluate combinations of meta-feature groups is the exploitation of the entire search space of 2^n possible combinations, where n is the number of groups. Specifically, the brute-force method just evaluates the effectiveness of each possible combination of meta-feature groups according to a cross-validation classification experiment on the training set. After evaluating all combinations, the method selects the most effective combination of them.

Besides selecting the most effective combination, brute-force can also be used as a tool to analyze different combinations of meta-features. In fact, brute-force is an effective strategy to obtain the Pareto Frontier that presents the trade-offs between maximizing effectiveness and minimizing the number of features in combinations of meta-feature groups.

3.2.2 GreedyMF

The recently proposed GreedyMF strategy (Canuto et al., 2014) is a simple proposal to evaluate only a few combinations of meta-feature groups by guiding the search for combinations

according to a greedy heuristic. Specifically, it removes the meta-feature group that is most harmful to the effectiveness at each iteration. The strategy can be outlined by the following steps:

1. Let $\mathscr{M}_{sub} = \mathscr{M}$, set formed by all meta-feature groups.
2. Find a meta-feature group $\mathscr{M}_{bad} \subset \mathscr{M}_{sub}$ which, when removed, achieves the highest effectiveness improvement. If none is found, stop. Otherwise, go to step 3.
3. Do $\mathscr{M}_{sub} = \mathscr{M}_{sub} \setminus \mathscr{M}_{bad}$ and go to step 2.

For example, let's assume that $\mathscr{M}_{sub}$ initially corresponds to all the meta-feature groups described in Table 3.1 and each one of these groups corresponds to a candidate for $\mathscr{M}_{bad}$. In each iteration, GreedyMF strategy removes the group $\mathscr{M}_{bad}$ which causes the most harm to the effectiveness according to a cross-validation experiment on the training set. When there is no statistically significant effectiveness improvements with the removals, the method stops.

3.2.3 Best-FirstMF

Best-FirstMF is a greedy strategy that instead of removing meta-feature groups from the set of all meta-features like the previously described strategy, it recursively includes the meta-feature group that is most beneficial to the effectiveness at each iteration.

It starts with n cross-validation experiments, each of them using one of the n meta-feature groups. The group that yields the experiment with the best effectiveness is selected. In the next iteration, it creates another set of $n-1$ experiments using two meta-feature groups: the one selected in the previous iteration and another of the $n-1$ remaining groups. Again, the combination of features that gives the best performance is selected. The script stops when there is no meta-feature group that can improve the classification effectiveness.

Despite simple and intuitive, the greedy strategies are capable of evaluating only a restrict region of the search space of possible combinations. In the next section, we propose an alternative capable of performing a robust search considering more diverse and promising combinations without evaluating all of them.

3.2.4 SPEA2SVM and SPEA2fast

We propose two multi-objective optimization strategies for meta-feature selection and evaluation, which are capable of searching for combinations of meta-feature groups considering different objectives without evaluating all possible combinations of meta-feature groups.

We argue that these strategies are not only capable of finding good combinations of meta-features, but are also important tools for analyzing different combinations considering different objectives. Both strategies correspond to evolutionary multi-objective proposals to meta-feature selection and are based on the SPEA2 (et al., 2001) algorithm, which presented successful results in related machine learning tasks (Dalip et al., 2014; Li et al., 2015; Sousa et al., 2019, 2016). The first strategy, named SPEA2SVM, is an effective approach to select and analyze combinations of meta-feature groups considering the effects of reducing the number of groups in the combination. The second approach, called SPEA2fast, is an efficient method for effective meta-feature selection on different classification strategies. We describe the original SPEA2 algorithm in Section 3.2.4.1. The specification of the initial population for our meta-feature selection strategies is described in Section 3.2.4.1. SPEA2SVM and SPEA2fast are described respectively, in Sections 3.2.4.2 and 3.2.4.4. We also describe a single-objective genetic algorithm strategy (SingleGA) that optimizes only classification effectiveness for the meta-feature selection task in Section 3.2.4.3.

3.2.4.1 SPEA2

SPEA2 is based on Genetic Algorithms (Srinivas and Patnaik, 1994) and thus uses an evolutionary approach to explore the solution space for a multi-objective problem. In this process, each solution (also referred to as an *individual*) receives a *fitness value* that scores its worth based on its likelihood of surviving in the next generation. Once the fitness values have been computed for each individual in one generation, the best individuals are selected to take part in the breeding of the next generation. These selected individuals are kept in an archive A_g during generation g. Thus, along with the process, the archives work as buckets to keep the best individuals over the generations. On the other hand, the unfit individuals are eliminated during this evolutionary process. After many generations, surviving individuals tend to be better than the eliminated ones, according to the fitness criteria.

Algorithm 2 describes the original SPEA2. The algorithm takes as input the size n of the population, the size a of the archive, and the number ng of generations. A population, $P_g = \{i_0, ..., i_n\}$, is the set of individuals in a generation g. In our case, each individual corresponds to a binary array (aka, a chromosome). A position in the array is defined as a *gene*. It is 0 when the feature group is absent, and 1 otherwise. The algorithm first creates an empty archive A_1 and a population P_1 with n individuals in Lines 1 and 2, respectively.

Once all individuals have been created, the fitness score for each one is computed (Line 3 and 22). When assigning scores to features, SPEA2 must consider the optimization of multiple objectives. Thus, the algorithm uses the *dominance relationship* among individuals to provide the fitness values. Considering the individuals i and j, we say that i dominates j

(denoted as $i \succ j$), if i is better than j in one objective, and i is not worse than j in any other one. Formally, considering $o_i^1, o_i^2 \ldots o_i^m$ as the values for the m objectives of an individual i, Equation 3.1 describes the dominance relationship.

$$i \succ j \iff (o_i^1 \geq o_j^1 \wedge o_i^2 \geq o_j^2 \wedge \ldots o_i^m \geq o_j^m) \wedge (o_i^1 > o_j^1 \vee o_i^2 > o_j^2 \vee \ldots o_i^m > o_j^m) \tag{3.1}$$

Given the dominance relationship, we say that an individual i is in the Pareto frontier when there is no other individual j that dominates i. In this case, i is called a *nondominated* individual. The *strength* $S(i)$ of an individual i is defined as the number of individuals who are dominated by i, as described in Equation 3.2.

$$S(i) = \mid \{j \mid j \in P_g \cup A_g \wedge i \succ j\} \mid \tag{3.2}$$

where $\mid . \mid$ is the cardinality of a set. Finally, the fitness score of i is computed by Eq. 3.3.

$$fitness(i) = R(i) + D(i) \tag{3.3}$$

where,

$$R(i) = \sum_{j \in (P_g \cup A_g) \wedge j \succ i} S(j) \tag{3.4}$$

$R(i)$ sums the strength of the individuals who dominate i. Note that $R(i) > R(j)$ means that the individual i is worse than individual j, as the individuals that dominate i are stronger than those who dominate j. Thus, the value of $fitness(i)$ is optimized by minimizing $R(i)$. When $R(i) = 0$, no individual dominates i, meaning that all individuals with $R(i) = 0$ are the best solutions, i.e., they belong to the Pareto frontier.

Term $D(i)$ is assigned to promote a large variety of solutions, as it decreases when i is farther from a dense region. In this sense, a higher priority is given to the more distinct individual, stopping the search process from being trapped in a local optimal solution. In addition, individuals with tied $R(i)$ values but in a sparse region will have more chance of surviving to the next generation. Term $D(i)$ in Equation 3.3 is referred to as *density estimate*, and it is calculated according to Equation 3.5:

$$D(i) = \frac{1}{\sigma_i^k + 2} \tag{3.5}$$

The value 2 is used to ensure that $D(i)$ is less than 1 and to keep the denominator greater than zero (et al., 2001). Also, σ_i^k is the distance (in objective space) between individual i and the k^{th} nearest individual using the K-nearest neighbor algorithm with the Euclidean distance. The parameter k is defined as $\sqrt{\mid A_g \mid + \mid P_g \mid}$.

Algorithm 1 The Original SPEA2 Algorithm.

Require: Population size n
Require: Size a of Archive (A_g)
Require: Number of generation ng
Require: Gene mutation probability p
Ensure: A_g close to Pareto frontier
Let P_g = pop. of individuals $\{i_0, ..., i_n\}$ of generation g
Let A_g = the best individuals of all generations until g
Let D_g = dominated individuals of P_g and A_g
Let N_g = non-dominated individuals of P_g and A_g
1: $A_1 \leftarrow \emptyset$
2: **Initialize** P_1 with random individuals
3: **Compute** fitness(i), $i \in P_1$
4: **for** $g = 1$ to ng **do**
5: with $i \in P_g \cup A_g$ do:
6: **Assign** i to D_g if fitness(i) ≥ 1
7: **Assign** i to N_g if fitness(i) < 1
8: **Add** N_g to A_g
9: **if** $\mid A_g \mid > a$ **then**
10: truncate(A_g)
11: **else if** $A_g < a$ **then**
12: $k = a - \mid A_g \mid$
13: Fill A_g with the k best individuals in D_g
14: $P_{g+1} \leftarrow \emptyset$
15: $A_{g+1} \leftarrow A_g$
16: **while** $\mid P_{g+1} \mid -1 < n$ **do**
17: **Select** two individuals i_x and i_y from A_g.
18: (new_i_x, new_i_y) = *crossover*(i_x, i_y)
19: **Add** new_i_x and new_i_y to P_{g+1}
20: **fol all** $i \in P_{g+1}$
21: *random_mutate(i,p)*
22: **Compute** fitness(i), $i \in P_{g+1} \cup A_{g+1}$

After computing the fitness for each individual, the algorithm defines the D_g and N_g sets, putting inside D_g the individuals which are dominated by other individuals (Line 6), and in N_g all *nondominated* individuals (Line7).

Lines 8-13 of Algorithm 2 define the elitism process, saving in the archive (A_g) all the *nondominated* individuals of the population. If there is more individuals in the archive than the limit a (Line 9), the algorithm keeps only the a individuals with the best fitness in the archive. If the archive is not full (Line 13) the algorithm fills the archive with the best individuals in D_g (i.e. individuals that despite being dominated have small fitness).

After A_g is full, the algorithm initializes the next generation archive (A_{g+1}) with A_g (Line 15). Next, it creates a new population P_{g+1}, performing crossover and mutation on individuals of the current archive (A_g). Crossover is performed by using the Tournament

Selection method (Srinivas and Patnaik, 1994) (Line 17), which selects the individuals with highest fitness values, among a few set of individuals chosen at random from A_g. Using the *Two Point Crossover* (Srinivas and Patnaik, 1994) method, the uniform crossover is performed (Line 18) exchanging a random genes on two selected individuals.

In Lines 21 and 22, the algorithm applies a random selection for mutation to each individual. The $random\_mutate(i, p)$ method flips a coin for each gene in the chromosome corresponding to i, changing its content with probability p.

After Algorithm 2 is completely executed, we can analyze the trade-offs between the considered objectives using the nondominated individuals in the last archive, or just select one of the individuals from the archive according to some heuristic. In the next sections we present details about objectives, selection strategies and adaptations in the original SPEA2 method for effective meta-feature selection and analysis.

Initial Population Instead of using only randomly generated individuals for the initial population, we include individuals to guide the genetic algorithm towards the most promising regions of the search space faster. We include individuals that minimize the number of meta-feature groups as well as potentially effective individuals. Considering the initial individuals that minimize the number of groups, each meta-feature group in Table 3.1 corresponds to an individual that contains only one active gene (that represents its corresponding meta-feature group).

We also include four potentially effective combinations of meta-feature groups as initial individuals. The first one is the combination of all meta-feature groups (i.e, all genes are active). The other three individuals correspond to the literature combinations described in Table 3.1 and proposed by Canuto et al. (2014), Gopal and Yang (2010) and Pang et al. (2015).

3.2.4.2 SPEA2 with SVM (SPEA2SVM)

Since SVM is the state-of-the-art classification method in various domains and the best performing classifier for meta-features (as shown in Section 3.3.4), we focus on searching for the Pareto frontier considering the maximization of the SVM model effectiveness on a 5-fold cross validation experiment as objective along with the minimization of the number of meta-feature groups of the individuals. The values of these objectives are derived from an individual represented by a binary vector, where each position corresponds to the presence or absence of a meta-feature group.

The benefits of using this SPEA2SVM for meta-feature selection are two-fold: (i) it is capable of searching for individuals that maximize the classification effectiveness with

as few meta-feature groups as possible and (ii) we can interpret the impact of our objectives by analyzing the non-dominated individuals found by SPEA2SVM. For example, let us consider two non-dominated individuals (i.e., individuals on the Pareto frontier) i and j. Considering effectiveness as the objective function, suppose that i is a little higher than j, using the MacroF1 as effectiveness measure. Otherwise, taking into account the minimization of meta-feature groups, j can use substantially less meta-feature groups than i. In this case, it is interesting to analyze the hypothesis that meta-feature groups in j provide the most discriminative information to classify documents and the reasons why i is not able to substantially improve the effectiveness of j.

SPEA2SVM provides a feasible alternative to the brute-force for the task of finding the Pareto frontier when considering various meta-feature groups or big datasets. The combinations of meta-feature groups in the Pareto frontier provide information to analyze the core information to classify documents, which is essential to answer our fourth question (Q4). We also use SPEA2SVM as a new feature selection approach by selecting the most effective non-dominated individual on a 5-fold cross validation experiment. The effectiveness of SPEA2SVM as a meta-feature selection strategy and other feature selection approaches will be evaluated to answer our second research question (Q2).

3.2.4.3 Single Objective GA (SingleGA)

SingleGA is the single objective version of the previously described SPEA2SVM genetic strategy. Unlike SPEA2SVM, SingleGA does not aim at finding the Pareto frontier among objectives, and therefore it is not useful as an evaluation tool to analyze core combinations of meta-feature groups. Instead, it only maximizes classification effectiveness as goal, being an important baseline to evaluate the effects of exploiting the search space guided only by effectiveness on genetic algorithms.

3.2.4.4 Efficient SPEA2 for Multiple Classifiers (SPEA2fast)

Despite the good effectiveness of the SVM classifier, it is sometimes very slow to compute, specially with a large number of categories, since it is a binary classifier that separates the feature space into two subspaces using a hyperplane, requiring costly solutions for multiclass tasks (Hastie et al., 2003). Consequently, evaluating the effectiveness of each individual with SVM can be computationally expensive. Moreover, the evaluation of individuals might be biased to those that work effectively with the SVM classifier, which limits the evaluation of the individuals on other classification approaches.

In this section, we propose an efficient meta-feature selection method that does not rely on the SVM classifier to find effective combinations for SVM and other classification

approaches. Specifically, we evaluate the effectiveness of individuals without relying on expensive classifiers (such as SVM). Instead, we propose the maximization of the effectiveness on the following three classification methods as objectives of the multi-objective strategy SPEA2: Naive Bayes, Nearest Centroid, and Extreme Randomized Trees. These classification methods were chosen because they are very fast to execute and they are based on different classification paradigms as objectives[1]. In other words, the classification effectiveness of each chosen fast classifier is one objective of our multi-objective strategy. We argue that maximizing the effectiveness of diverse and fast classification methods as objectives can direct the search of effective combinations of meta-feature groups, and the most promising combinations can be found on the Pareto frontier.

The expected Pareto frontier considering these objectives may contain high quality individuals, which in most cases[2] To illustrate the intuition about why the Pareto frontier obtained in SPEA2fast contains high quality individuals for arbitrary classification methods, let us consider an example of three non-dominated individuals, i, j and z. Suppose that individuals i and j are the most effective individuals considering respectively, the Nearest Centroid and NB classifiers. Controversially, i and j are also the worst performers considering the objectives of maximizing the effectiveness on NB and Nearest Centroid, respectively. In this scenario, i and j are biased to optimize the performance on specific classifiers due to specificities of different classification paradigms. The third individual z achieves good results in both objectives, despite not achieving the best results in any of them. Besides z, there may be other non-dominated individuals with potentially good results in both objectives, but more biased towards one of them. We argue that the subset of non-dominated individuals in this Pareto frontier are high quality individuals that can be efficiently found and most of these individuals are not entirely biased to any classification paradigm. Therefore, we expect that some of these individuals can produce effective results for an arbitrary classification approach. Considering our previous example, the SVM classifier might not produce good results for using the individuals i or j, since each one is biased to optimize the performance of a specific classification method different from SVM. However, other non-dominated individuals such z that can produce good results for these methods are more likely to produce effective results for SVM.

In order to select the best individual to a specific classifier, we test every non-dominated individual found in the last archive produced by SPEA2fast using a target classification

[1]We do not include the minimization of the number of meta-feature groups as an objective, since it would considerably increase the size of the Pareto frontier, also increasing the computational cost of SPEA2fast. Moreover, fast classifiers usually rely on only a few meta-features to obtain their best effectiveness, which indirectly reduce the number of meta-features in individuals.

[2]With the exception of the limits of the Pareto frontier. are not biased to one specific fast classifier. In other words, SPEA2fast aim at finding individuals that can perform relatively well in all three classifiers.

method (e.g., SVM). We select the most effective individual based on a 5-fold cross-validation experiment for the target classification approach. This efficient and effective method for meta-feature selection considering diverse classification methods is an important tool to provide evidence for our third research question (Q3).

3.3 Experimental Evaluation

In this section, we provide experimental evidence to answer the proposed research questions using five datasets from different contexts. We first present the experimental setup and then we present results of experiments designed for providing evidences for the answers to our research questions.

3.3.1 Experimental Setup

3.3.1.1 Textual Datasets

In order to evaluate our research questions, we consider five real-world textual datasets, namely, 20 Newsgroups, Four Universities, Reuters, ACM Digital Library and MEDLINE. For all datasets, we performed a traditional preprocessing task: we removed stopwords, using the standard SMART list, and applied a simple feature selection by removing terms with low "document frequency (DF)"[3] and use the traditional TFIDF term weighting. Next, we give a brief description of each dataset.

4 Universities (4UNI), a.k.a, WebKB this dataset contains Web pages collected from Computer Science departments of four universities by the Carnegie Mellon University (CMU) text learning group. There is a total of $8,277$ web pages, classified in 7 categories (such as student, faculty, course and project web pages).

20 Newsgroups (20NG) this dataset contains $18,805$ newsgroup documents, partitioned almost evenly across 20 different newsgroups categories. 20NG has become a popular dataset for experiments in text applications of machine learning techniques, such as text classification and text clustering.

ACM-DL (ACM) a subset of the ACM Digital Library with $24,897$ documents containing articles related to Computer Science. We considered only the first level of the taxonomy adopted by ACM, whereas each document is assigned to one of 11 classes.

[3] We removed all terms that occur in less than six documents (i.e., $DF < 6$).

Reuters (REUT90) this is a classical text dataset, composed by news articles collected and annotated by Carnegie Group, Inc. and Reuters, Ltd. We consider here a set of 13,327 articles, classified into 90 categories.

MEDLINE (MED) a subset of the MedLine dataset, with 861,454 documents classified into 7 distinct classes related to Medicine. This dataset was obtained from Rocha et al. (2008). In that work, the authors considered the first level of the taxonomy so that each document article is classified under only one category, avoiding dealing with multilabel cases.

3.3.1.2 Evaluation, Algorithms and Procedures

The different combinations of meta-features were compared using two standard text categorization measures: micro averaged F_1 (MicroF_1) and macro averaged F_1 (MacroF_1) (Lewis et al., 2004; Yang, 1999). While the MicroF_1 measures the classification effectiveness over all decisions (i.e., the pooled contingency tables of all classes), the MacroF_1 measures the classification effectiveness for each individual class and averages them. All experiments were executed using a 5-fold cross-validation procedure (Breiman and Spector, 1992). This procedure involves partitioning a sample of data into five complementary subsets, where four of them are considered as training data to generate the classification model considering a combination of meta-features, and the remaining subset is considered as test set to measure the effectiveness of the classification model. We report the average effectiveness of the five possible different training/test partitions using this scheme.

In this work, we use the training set for two different tasks besides the usual tasks of building a classification model with the training examples and setting the parameters via cross-validation on the training set (i.e., nested cross-validation). The first task is the transformation of the original feature space into the meta-feature space, which is done by generating the meta-feature groups described in Section 3.1 for each document in the dataset. If document d is in the test partition, we use all the training samples to generate meta-features for d, and in case document d is a training example, we use all training examples except d to generate the meta-features to d.

After the generation of meta-features for the training and test partitions, we use each training partition for the task of finding effective combinations of meta-features using various feature selection strategies. Besides the methods described in Section 3, we evaluate seven other widespread general-purpose feature selection strategies, namely InfoGain (Yang and Pedersen, 1997), Chi2 (Liu and Setiono, 1995), RELIEF (Kira and Rendell, 1992), Pearson-Correlation(Hall, 2000), FCBF (Yu and Liu, 2003), CFS (Hall, 2000), and Consistency (Liu and Setiono, 1996).

Most feature selection strategies and classification algorithms rely on the selection of parameters to achieve their best results. The feature selection strategies InfoGain (Yang and Pedersen, 1997), Chi2 (Liu and Setiono, 1995), RELIEF (Kira and Rendell, 1992), PearsonCorrelation (Hall, 2000), and FCBF (Yu and Liu, 2003) rely on the the parameter that controls the number of selected features. This parameter was chosen among 5%, [10%, 90%] with steps of 10% and 95% of the total number of features by using a nested 5-fold cross-validation within the training datasets. The parameters necessary to execute our SPEA2-based meta-feature selection algorithms were chosen according to the general guidelines provided in previous works (Chen et al., 2012; Laumanns et al., 2001) and the specificities of machine learning tasks (Dalip et al., 2014; Sousa et al., 2016).We evaluate our parameterization using the brute-force results as the gold standard for the Pareto frontier. Particularly, we search for the lowest number of generations and population size (always keeping identical population and archive sizes) that achieve comparable results with the gold standard. As a result, we set the following parameters in Algorithm 2: number of generations $ng = 30$, population size $n = 20$, archive size $a = 20$ and mutation probability $p = 0.03$. In fact, using the this parameterization, SPEA2SVM can produce similar effectiveness to the optimal solution provided by the brute-force method on a fraction of its execution time.

In order to evaluate the performance of different combinations of meta-features, we adopted the LIBLINEAR (Fan et al., 2008) implementation of the SVM classifier, and the scikit-learn (Pedregosa et al., 2011) implementation of the classifiers random forests (RF), extreme randomized trees (XF), nearest centroid (CENT) and Gaussian naive Bayes (NB). The regularization parameter of SVM was chosen among eleven values from 2^{-5} to 2^{15} by using 5-fold cross-validation on each training dataset. Likewise, the percentage of features to consider when looking for the best split of a tree from RF and XF was chosen among six values from 10% to 60%. Each tree is grown without pruning, as suggested by Breiman (2001), and since there are no statistically significant differences on results obtained with 500, 1000, and 2000 trees, we adopted 500 trees due to the lower cost. The size of neighborhood used in knn-based meta-features is chosen among ten values from 10 to 100 according to the number of neighbors that maximizes the kNN effectiveness in the cross-validation on each training dataset.

In order to consider the cost of all processing power of each approach, we conduct our experiments on a Supermicro server with two E5-2620 Intel Xeon processors and one GeForce GTX TITAN Black consumer GPU. We report the wall time consumed by the process execution in all experiments regarding execution time.

To compare the average results of our cross-validation experiments, we assess the statistical significance of our results by means of a paired 2-tailed t-test with 95% confidence and Holm correction to account for multiple tests. T-test is strongly recommend over sign and

Wilcoxon tests for hypothesis testing on mean effectiveness (Urbano et al., 2019), which is arguably robust to violations of the normality assumption (Hull, 1993). Either way, the bag-of-words and the set of all meta-features follow the normal distribution under the Shapiro-Wilk test (Luengo et al., 2009). We mark the best results and results that are not statistically inferior to the best in **bold**. The obtained results (and their 95% confidence intervals) are described in the following sections.

We would like to point out that some of the results obtained in some datasets with and without the meta-features may differ from the ones reported in other works for the same datasets (Godbole and Sarawagi, 2004; Kim et al., 2006; Lan et al., 2006). Such discrepancies may be due to several factors such as differences in dataset preparation[4], the use of different splits of the datasets (e.g., some datasets have "default splits"[5] such as 20NG and REUT), the application of some score thresholding, such as SCUT, PCUT, etc., which, besides being an important step for multilabel problems, also affects classification performance by minimizing class imbalance effects, among other factors. We would like to stress that we ran all alternatives under the same conditions in all datasets, using the best traditional feature weighting scheme (TFIDF), using standardized and well-accepted cross-validation procedures that optimize parameters for each of alternatives, and applying the proper statistical tools for the analysis of the results. All our datasets are available for others to replicate our results and test different configurations.

3.3.2 Effectiveness of Meta-features (Q1)

We start by analyzing the effectiveness of the SVM classifier on different combinations of meta-feature groups. We aim at evaluating how effective is the combination of all the meta-feature groups proposed in different works, which corresponds to our first research question. Table 3.2 shows the results of various combinations of meta-feature groups and the original bag-of-words feature space. More specifically, we present the combination of all meta-feature groups (AllMF), described in Table 3.1, and the combination of meta-feature groups proposed in each literature work (e.g., the combination of meta-feature groups proposed by Gopal and Yang (2010), as can be seen in in Table 3.1). The combination AllMF presents better results than the combination of the meta-feature groups proposed by Canuto et al. (2014), Gopal and Yang (2010), Pang et al. (2015) and the bag-of-words on most datasets, improving the results of Gopal and Yang (2010), Pang et al. (2015) and Canuto et al. (2014) by up to 9%, 40% and 4%, respectively. The fact that AllMF achieves better results than other

[4]For instance, some works do exploit complex feature weighting schemes or feature selection mechanisms that do favor some algorithms in detriment to others.

[5]In fact, we do believe that running experiments only in the default splits is not the best experimental procedure as it does not allow a proper statistical treatment of the results.

combinations of meta-feature groups on MED, ACM and 4UNI provides evidence of complementary discriminative information among the different meta-feature groups proposed in the literature.

Conversely, AllMF produces worse results than Gopal et al on REUT and 20NG, indicating that the additional information in AllMF can cause overfitting to the classification model on these datasets. This hypothesis is also supported by the fact that these datasets provide only a few training documents per class, which may not be enough to cope with the complexity and the large number of meta-features provided by AllMF. Since the other considered datasets do not suffer from a shortage of training information, AllMF is always better than Gopal et al on them, with gains up to 10% and 6% on MacroF1 and MicroF1, respectively.

AllMF also presents substantial gains ranging from 11% to 40% over Pang et al for most datasets, except REUT. The gains are expected, since the Pang et al's meta-features only provide global information about the distribution of distances between examples and their category centroids. However, the use of centroid information was enough to achieve good MacroF1 results on REUT. In this scenario, the centroid information is important to prevent errors considering the imbalanced and small number of training examples available for some categories.

AllMF presents statistically superior results compared with the Bag of Words (Bow) for most datasets, with gains up to 9% and 12% respectively, in MicroF1 and MacroF1. The only statistically loss of AllMF over the BoW representation is the MicroF1 result on MEDLINE, which is only 1% lower. One possible justification for this fact is that the number of documents in this dataset is one order of magnitude higher than in the other datasets. As a consequence, the training sets have sufficient examples to deal with the large space of textual features. Since some MEDLINE classes do not have many training examples, the AllMF's MacroF1 is as good as BoW's MacroF1 on this dataset.

Despite the overall good effectiveness of AllMF and the fact that the number of dimensions in AllMF is usually much smaller than the number of dimensions in the BoW representation, AllMF is still highly dimensional and complex (dense) feature space. Complementary information from different meta-feature groups brings benefits to the effectiveness for most cases, but also harms the effectiveness in some scenarios due to its complexity. This situation motivates our work to further exploit the selection and analysis of meta-features.

3.3.3 Meta-Feature Selection Results (Q2)

In this section, we answer our second research question showing how effective and efficient are different strategies for feature selection on the meta-feature space. Specifically, we eval-

		20NG	4UNI	REUT	ACM	MEDLINE
Canuto et al. (2014)	MacF1	88.3(0.6)↓	**66.1(2.6)**↕	32.4(2.6)↓	**64.1(1.1)**↕	72.7(0.5)↓
	MicF1	88.5(0.6)↓	78.9(1.6)↓	71.5(0.9)↕	75.5(0.8)↓	82.5(0.2)↓
Gopal and Yang (2010)	MacF1	**89.5(0.5)**↑	60.6(2.7)↓	**41.7(2.8)**↑	62.7(1.4)↓	74.9(0.2)↓
	MicF1	**89.8(0.6)**↑	75.6(0.7)↓	**77.9(1.2)**↑	75.6(0.4)↓	84.2(0.1)↓
Pang et al. (2015)	MacF1	77.4(0.6)↓	56.4(1.8)↓	37.2(1.6)↑	52.1(1.6)↓	46.3(1.0)↓
	MicF1	78.3(0.7)↓	67.6(1.1)↓	71.8(0.8)↕	65.0(0.9)↓	66.3(1.0)↓
Bag of Words	MacF1	87.8(0.2)↓	60.4(1.0)↓	29.5(2.1)↓	61.6(0.4)↓	**76.0(0.2)**↕
	MicF1	87.6(0.2)↓	70.7(0.8)↓	65.7(0.7)↓	72.1(0.5)↓	**85.6(0.5)**↑
allMF	MacF1	88.8(0.6)	**66.3(1.4)**	33.5(2.6)	**64.2(1.0)**	**76.0(0.4)**
	MicF1	89.0(0.6)	**80.3(1.2)**	71.9(1.3)	**76.2(0.7)**	84.6(0.2)

Table 3.2: Comparison between the meta-features proposed in different literature works and the combination of all meta-features allMF, using the SVM classifier. ↑, ↓ and ↕ correspond, respectively, to statistically significant gains, losses and no evidence over the set of all meta-features allMF.

uate the effectiveness and efficiency of different meta-feature selection methods designed to remove groups of meta-features (Brute-force, GreedyMF, SPEA2fast, SPEA2SVM, BestFirstMF, and SingleGA) and various general feature selection methods on the meta-feature space of all meta-features (AllMF).

We start by noting that, according to Table 3.3 the proposed methods SPEA2SVM and SingleGA are the only methods that always improve classification effectiveness. They also are capable of significantly reduce the number of meta-features from AllMF. Moreover, our proposals SPEA2SVM, SPEA2fast, and SingleGA are never statistically inferior to the brute-force approach on 4UNI, 20NG, and ACM without the need to perform the computationally expensive evaluation of all possible combinations of meta-feature groups. This extensive evaluation, makes brute-force infeasible on datasets containing a high number of categories or training examples[6], which is the case of MED and REUT datasets.

The meta-feature selection methods SPEA2SVM, SingleGA, and SPEA2fast are capable of obtaining the best results on both REUT and 20NG, with significant gains over AllMF – up to 24% and 7.6% respectively, in Macro and Micro F1 on REUT. As discussed in the previous section, REUT and 20NG have only a few training examples for each category. Therefore, considering complex or highly dimensional feature spaces to represent training examples of these datasets might lead to overfitting. SPEA2SVM, SPEA2fast and SingleGA were the methods capable of automatically choosing the most effective meta-feature groups considering the peculiarities of these datasets.

GreedyMF is the most conservative approach tested to remove meta-feature groups. Despite keeping the results of AllMF on most datasets, it was not able to provide statistically significant improvements on 20NG and it was not capable of achieving the same effec-

[6]The SVM method is less efficient to compute individuals with various meta-feature groups (which consumes most of the execution time on the brute-force algorithm), and datasets with various categories (since the SVM method needs to generate one classifier per category on the adopted one-versus-all approach).

		20NG	4UNI	REUT	ACM	MED
SPEA2SVM	MacF1	**89.7(0.6)**↑	**66.5(1.4)**↕	**41.5(3.1)**↑	**64.9(1.4)**↕	**75.7(0.6)**↕
	MicF1	**90.0(0.7)**↑	**79.9(1.4)**↕	**77.4(1.5)**↑	**76.3(0.7)**↕	**84.4(0.5)**↕
SingleGA	MacF1	**89.6(0.6)**↑	**66.3(1.5)**↕	**41.3(2.8)**↑	**64.4(1.1)**↕	**75.9(0.5)**↕
	MicF1	**90.0(0.6)**↑	**80.1(1.3)**↕	**77.3(1.7)**↑	**76.3(1.2)**↕	**84.5(0.3)**↕
SPEA2fast	MacF1	**89.6(0.5)**↑	**67.0(2.5)**↕	**41.2(3.2)**↑	**64.4(1.4)**↕	75.0(0.9)↓
	MicF1	**89.9(0.6)**↑	**79.6(0.9)**↕	**77.0(1.2)**↑	**76.1(0.6)**↕	83.9(0.6)↓
Brute-force	MacF1	**89.7(0.5)**↑	**66.3(1.0)**↕	*	**64.6(1.2)**↕	*
	MicF1	**90.1(0.7)**↑	**79.8(1.2)**↕	*	**76.4(0.9)**↕	*
Best-FirstMF	MacF1	**89.5(0.4)**↑	**66.7(1.3)**↕	**41.5(3.1)**↑	**64.8(1.5)**↕	75.2(0.6)↓
	MicF1	89.6(0.5)↑	**79.9(1.5)**↕	**77.4(1.5)**↑	**76.1(0.5)**↕	84.0(0.5)↓
GreedyMF (Canuto et al., 2014)	MacF1	89.0(0.9)↕	**66.3(1.4)**↕	37.1(3.3)↑	**63.9(0.9)**↕	**76.0(0.4)**↕
	MicF1	89.3(0.9)↕	**80.3(1.2)**↕	73.1(1.4)↑	**76.3(0.9)**↕	**84.6(0.2)**↕
CFS (Hall, 2000)	MacF1	87.2(0.8)↓	59.2(2.6)↓	29.3(2.6)↓	57.5(1.8)↓	71.0(0.4)↓
	MicF1	87.4(0.8)↓	73.2(2.8)↓	73.3(0.7)↑	72.3(0.5)↓	81.7(0.1)↓
Consistency (Liu and Setiono, 1996)	MacF1	47.7(3.2)↓	52.5(3.6)↓	23.5(3.7)↓	49.1(1.8)↓	58.3(6.7)↓
	MicF1	54.0(2.5)↓	70.7(2.7)↓	68.7(1.3)↓	68.8(0.7)↓	79.7(1.3)↓
InfoGain (Yang and Pedersen, 1997)	MacF1	89.0(0.7)↕	**67.4(1.6)**↕	37.4(2.6)↑	**63.9(1.3)**↕	**75.9(0.4)**↕
	MicF1	89.2(0.8)↕	**79.5(1.4)**↕	75.2(1.9)↑	**75.8(0.9)**↓	**84.5(0.3)**↕
Chi2 (Liu and Setiono, 1995)	MacF1	89.1(0.7)↕	**66.9(1.1)**↕	36.6(3.2)↕	**64.1(1.0)**↕	**75.7(0.3)**↕
	MicF1	89.3(0.8)↕	**79.2(1.0)**↕	74.4(1.9)↑	**76.0(0.5)**↕	**84.3(0.4)**↕
RELIEF (Kira and Rendell, 1992)	MacF1	88.7(0.7)↕	**67.0(1.8)**↕	33.3(4.6)↕	**64.0(1.2)**↕	**75.8(0.5)**↕
	MicF1	89.0(0.8)↕	**79.2(1.9)**↕	71.0(2.3)↕	**76.2(0.6)**↓	**84.4(0.2)**↕
PearsonCorrelation (Hall, 2000)	MacF1	89.0(0.7)↕	**67.3(2.9)**↕	35.3(3.3)↕	**64.9(1.8)**↕	**75.8(0.4)**↕
	MicF1	89.2(0.7)↕	**79.5(1.3)**↕	72.7(1.6)↕	**76.2(0.6)**↕	**84.4(0.4)**↕
FCBF (Yu and Liu, 2003)	MacF1	89.0(0.6)↑	**67.4(1.3)**↕	36.6(2.0)↑	**63.8(1.3)**↕	**75.8(0.4)**↕
	MicF1	89.2(0.6)↑	**79.4(1.4)**↕	74.2(1.8)↑	**76.1(0.6)**↕	**84.3(0.3)**↕
allMF	MacF1	88.8(0.6)	**66.3(1.4)**	33.5(2.6)	**64.2(1.0)**	**76.0(0.4)**
	MicF1	89.0(0.6)	**80.3(1.2)**	71.9(1.3)	**76.2(0.7)**	**84.6(0.2)**

Table 3.3: Average MicroF$_1$ and MacroF$_1$ of different feature selection strategies on the SVM as classifier. ↑, ↓ and ↕ correspond, respectively, to statistically significant gains, losses and without significances over the set of all meta-features allMF.

tiveness improvements of SPEA2SVM, SingleGA, Best-FirstMF, and SPEA2fast on REUT. Despite following the same paradigm as GreedyMF, the Best-FirstMF method is superior to GreedyMF in most datasets, except MED. This provides evidence for the benefits of the recursive inclusion of the best meta-features to build a solution, instead the removal of the worst meta-features from the full set of meta-features. Both GreedyMF and Best-FirstMF are not adequate for exploiting the search space of possible combinations, since the removal or inclusion of only one meta-feature group at time restricts the search for good combinations. This is not the case for genetic algorithms, since they are designed to exploit diverse regions of the search space in a non-linear way.

We now turn our attention to the the seven general purpose feature selection strategies capable of evaluating the "worthiness" of a subset of features or of individual features. The strategies CFS and Consistency are both based on finding a compact subset of features by filtering irrelevant and redundant information regardless of a specific classification method. Since various meta-features are highly correlated, these methods consider most meta-features as redundant, filtering them. As a consequence, CFS and Consistency present the worst results on most datasets.

The methods InfoGain, Chi2, RELIEF, PearsonCorrelation and FCBF provide scores that measure the discriminative value of each individual feature. Therefore, they provide a ranking of features, where only the top performers are selected. Most of these methods are not capable of significantly reducing the feature space or improving the effectiveness results in most datasets, since they are not designed to keep less discriminative features that provide complementary evidence to classification, which is the case of various meta-features. The only exception is the FCBF approach, which considers both relevance and redundancy among the top features. As a result, FCBF is the only method among the baselines capable of obtaining small but statistically significant improvements on both REUT and 20NG. Despite its improvements, FCBF is statistically worse than SPEA2SVM, SPEA2fast, Best-FirstMF, and SingleGA on both datasets and removes significantly fewer features than SPEA2SVM and SVM2fast, as we shall see next.

We show the proportion of meta-features removed (from the total number of generated meta-features for AllMF described in details in Table 3.5) using all the feature selection approaches in Table 3.4. Consistency and CFS always remove more than 95% of the meta-features, but present significant effectiveness losses on most datasets. The other standard methods – InfoGain, Chi2, RELIEF, PearsonCorrelation and FCBF – remove only a few meta-features (mostly up to 10%) from 4UNI, ACM and MEDLINE. By removing just a small number of features, these methods can keep the effectiveness of AllMF in most cases, with the expense of not being able to remove harmful features or produce significant dimensionality reductions.

	20NG	4UNI	REUT	ACM	MEDLINE
SPEA2SVM	85%	65%	89%	51%	28%
SingleGA	62%	58%	68%	39%	13%
SPEA2fast	82%	58%	87%	48%	47%
Brute-force	87%	62%	*	53%	*
Best-FirstMF	78%	62%	89%	45%	36%
GreedyMF (Canuto et al., 2014)	20%	0%	65%	0%	0%
CFS (Hall, 2000)	93%	97%	98%	96%	95%
Consistency (Liu and Setiono, 1996)	98%	98%	99%	98%	95%
InfoGain (Yang and Pedersen, 1997)	40%	10%	80%	5%	5%
Chi2 (Liu and Setiono, 1995)	60%	10%	80%	5%	5%
RELIEF (Kira and Rendell, 1992)	20%	40%	60%	5%	5%
PearsonCorrelation (Hall, 2000)	40%	10%	40%	20%	5%
FCBF (Yu and Liu, 2003)	40%	20%	60%	5%	5%
Total Number of Meta-Features	1360	805	12420	979	385

Table 3.4: Reduction of the number of features on different feature selection strategies

The most conservative approach is GreedyMF, which usually does not remove any meta-feature to keep or improve the effectiveness of AllMF in all situations. On the other side, Best-FirstMF is able do provide significantly more feature reduction following the same greedy paradigm because of the fact that Best-FirstMF tries to include only meta-features that

	#Meta-Features	20NG	4UNI	REUT	ACM	MEDLINE
cos_knn	|C|*k	200	210	900	220	70
cos_cent	|C|	20	7	90	11	7
l1_knn	|C|*k	200	210	900	220	70
l2_knn	|C|*k	200	210	900	220	70
l2_cent	|C|	20	7	90	11	7
cnt	|C|	20	7	90	11	7
ncnt	|C|	20	7	90	11	7
qrt	|C|*5	100	35	450	55	35
fisher	|C|	20	7	90	11	7
ig	|C|*3	60	21	270	33	21
sum_cent	$\|C\|^2$	400	49	8100	121	49
cqrt	|C|*3	100	35	450	55	35
total		1360	805	12420	979	385

Table 3.5: Number of meta-features generated for each group on each dataset according to the number of classes |C| and the number of neighbors k.

improve the results. Because of this inclusion strategy, Best-FirstMF was not able to keep the AllMF results in the MED dataset as GreedyMF, which does not remove any meta-features from MED.

SPEA2SVM is the only method capable of obtaining a substantial reduction of the number of features while improving or keeping the effectiveness of AllMF considering all datasets. SPEA2SVM and SingleGA achieved the best effectiveness results among all strategies, but SPEA2SVM can produce more compacted representations. These results provide evidence for the benefits of explicitly optimizing the minimization of the number of meta-features in SPEA2SVM.

SPEA2fast provides a different strategy to achieve close effectiveness and feature reduction results when compared to SPEA2SVM in most datasets. This was possible because SPEA2fast optimizes effectiveness considering multiple classifiers, some of which (e.g. Naive Bayes and Nearest Centroid) rely on a reduced number of meta-features to achieve their best effectiveness. This tends to guide the SPEA2fast genetic algorithm towards the most promising regions of the search space indirectly producing a reduced number of meta-features.

The high effectiveness of SPEA2SVM, SingleGA and Brute-force are due to the extensive use of classification approaches to evaluate possible combinations of meta-feature groups (i.e., wrapper strategies). As a consequence, they are among the most computationally expensive approaches, as shown in Table 3.6. Brute-force is the most expensive approach, since it builds SVM classification models to evaluate every possible combination of meta-feature groups. SPEA2SVM also builds various SVM classification models to evaluate the worthiness of the individuals. However, SPEASVM only evaluates the most promising combinations of meta-feature groups in diverse regions of the search space. This guided evaluation drives the search to smaller and relatively fast-to-compute individuals,

usually requiring only a few interactions to converge. Consequently, SPEA2SVM is capable of providing effectiveness and meta-feature reduction results as good as the ones provided by brute-force being about 100 times faster than the latter. SPEA2SVM is also up to 4 times more efficient than SingleGA, as it usually evaluates individuals with less meta-features than SingleGA.

Similarly to SPEA2SVM, SPEA2fast also exploits diverse promising combinations of meta-feature groups. However, it only uses fast classification approaches to evaluate individuals, which improves its efficiency. In fact, it is one of the most efficient methods among the meta-feature selection approaches, being up to 31 times faster than SPEA2SVM. Even the restrict search of GreedyMF and Best-FirstMF, which allows them to build only a few SVM models, is not as efficient as SPEA2fast in most evaluated datasets. As we can see, GreedyMF can be relatively slow to converge as it relies on the SVM classifier to build models usually using most meta-feature groups. Conversely, Best-first evaluates combinations with only a few meta-features and it is significantly faster than GreedyMF even though it requires more iterations to converge.

The general-purpose feature selection approaches are usually fast to compute, since they do not make extensive use of classification approaches to evaluate meta-features. CFS and Consistency both search for a subset of uncorrelated and informative features. CFS uses a fast heuristic based on the Pearson correlation, while Consistency measures inconsistencies among the subset of candidates and the full set of features. CFS is always significantly faster than Consistency due to its fast heuristic to compute correlations. InfoGain, Chi2, RELIEF, PearsonCorrelation and FCBF compute scores to measure the worthiness of individual features, and then use SVM a few times to select a cutoff value that excludes the worst evaluated features. The fast evaluation of individual features and the limited use of the classification method are the reasons for the short execution time of these feature selection approaches.

	20NG	4UNI	REUT	ACM	MEDLINE
SPEA2SVM	53873	11505	1564117	84494	1360846
SPEA2fast	17756	3504	50817	17903	44873
Best-FirstMF	9697	4211	198319	28627	532854
SingeGA	66382	25421	4914201	120439	3726032
Brute-force	3932160	505560	*	4157440	*
GreedyMF (Canuto et al., 2014)	25964	5804	301614	55641	458549
CFS (Hall, 2000)	1161	74	19428	451	4625
Consistency (Liu and Setiono, 1996)	2551	349	85844	2031	32240
PearsonCorrelation (Hall, 2000)	1262	364	22851	3340	29199
InfoGain (Yang and Pedersen, 1997)	1572	381	25514	3438	26124
Chi2 (Liu and Setiono, 1995)	1528	376	25179	3412	25482
RELIEF (Kira and Rendell, 1992)	1356	363	23791	3335	22968
FCBF (Yu and Liu, 2003)	1536	377	25528	3419	25923

Table 3.6: Execution time (in seconds) of different feature selection strategies.

Despite the short execution times of the general-purpose feature selection approaches,

none of them provide substantial reduction of meta-features, which is essential for our next evaluations. Since we provide evidence that the proposed approaches SPEA2SVM and SPEA2fast are capable of achieving good results in terms of both - effectiveness and feature reduction – and considering that they are designed to evaluate different combinations of meta-features groups, we choose them as tools for analyzing the meta-features described in Section 3.1.

3.3.4 Effectiveness on Different Classifiers (Q3)

In this section, we aim to answer our third research question: *how effective are different classification approaches when using the distance-based meta-features?* We evaluate the use of different classification paradigms in two scenarios. The first scenario consists in executing different classifiers considering all meta-features. In the second scenario, we adopt feature selection in our evaluations since each classification approach may achieve better results with a specific combination of meta-features.

We start by analyzing the effectiveness of different classification methods considering all meta-features. In this scenario, Table 3.7 shows that SVM results are consistently the best on all datasets. These results provide evidence towards the robustness of SVM on dealing with the diverse meta-features that use different distributions of numeric values.

SVM's effectiveness is always statistically superior to RF, with gains up to 27% and 8% in MacroF1 and MicroF1, respectively. XF, which is also based on an ensemble of decision trees, presents results close to RF. These results provide evidence for the shortcomings of exploiting meta-features using association rules provided by random forest or extremely randomized trees. More specifically, meta-features provide document representations based on direct evidence from similarity measures and statistics derived from them. We hypothesize that those dimensions are difficult to exploit with algorithms based on a recursive partitioning of the space, as explored in RFs , since they discretize meta-features. This may reduce the capability to exploit nuances of the general behavior found on numeric meta-features. We will investigate this hypothesis further in the future.

The classifiers NB and CENT are always significantly worse than SVM, RF, and XF. This provides strong evidence towards the limitations of NB and CENT to cope with the complex feature space produced by all meta-features. In fact, both methods are not capable of combining features that follow different distributions and do not have sophisticated internal mechanisms to evaluate the worthiness of the meta-features.

We now turn our attention to the reduced space of meta-features selected by SPEA2fast, which was specifically designed to perform an efficient search for effective combinations of meta-feature groups considering different classifiers. Table 3.8 shows that our

		20NG	4UNI	REUT90	ACM	MEDLINE
SVM	MacF1	**88.8(0.6)**	**66.3(1.4)**	**33.5(2.6)**	**64.2(1.1)**	**76.0(0.4)**
	MicF1	**89.0(0.6)**	**80.3(1.2)**	**71.9(1.3)**	**76.2(0.6)**	**84.6(0.2)**
RF	MacF1	86.4(0.8)	61.1(3.9)	26.9(1.0)	58.7(0.6)	74.0(0.4)
	MicF1	86.6(0.8)	75.3(2.1)	68.4(0.9)	72.7(0.6)	83.4(0.0)
XF	MacF1	86.1(0.4)	60.2(2.6)	27.4(1.7)	56.0(0.4)	72.3(0.5)
	MicF1	86.3(0.4)	74.2(1.8)	69.1(0.6)	70.8(0.7)	81.9(0.1)
NB	MacF1	56.6(1.4)	15.2(8.0)	15.2(0.5)	33.6(2.1)	38.8(1.4)
	MicF1	57.8(2.0)	16.0(8.7)	39.2(2.4)	37.7(3.6)	44.9(2.3)
CENT	MacF1	68.1(0.7)	55.8(1.7)	17.0(0.6)	51.8(1.0)	63.9(0.6)
	MicF1	64.8(1.0)	62.5(2.0)	44.1(1.3)	58.8(2.0)	78.6(0.4)

Table 3.7: Average MicroF$_1$ and MacroF$_1$ of different classifiers using all meta-features.

meta-feature selection approach significantly improves the results (previously presented in Table 3.7) for the classifiers NB and CENT on all datasets, with gains up to 73% and 50% in MacroF1, respectively. Moreover, after the selection CENT is capable obtaining competitive results with RF in most datasets. These results indicate that simple classifiers might benefit from combining specific groups of meta-features, as we analyze in details latter in this section.

The RF and XF results do not improve with meta-feature selection in any dataset. We argue that these results are primarily due to the capability of RF and XF to build models by evaluating the worthiness of each meta-feature, meaning that the most discriminative meta-features can contribute more for classification. This characteristic, combined with bagging of decision trees implemented in both approaches, already reduce the impact of noisy or irrelevant meta-features.

Despite providing consistently the best results, SVM is not capable of avoiding the impact of noisy or irrelevant meta-features in all datasets, since the meta-feature selection is capable of improving its results on 20NG and REUT. For both datasets, the shortage of information provided by just a few documents per class tends to harm the same mechanism that grants SVM its best results: its capability of finding nuances that provide discriminative information for classification.

In order to analyze the combinations of meta-feature groups found by SPEA2fast for each classification method, we present the most common among the best combinations found by the executions of SPEA2fast in all datasets. Each column in Table 3.9 corresponds to a combination, where the ✓cells indicate a selected group for the best individual.

As expected, each classifier achieves its best results with a different combination of meta-features, since each classification approach exploits the meta-feature space in a different way. Specifically, the meta-features l1_knn do not appear in the individual for the SVM classifier because they usually do not provide complementary discriminative evidence to the l2_knn and cosine similarity exploited by l2_knn and cos_knn, respectively. The NB classifier achieves its best results using only the ncnt group, probably because it is easy to

		20NG	4UNI	REUT90	ACM	MEDLINE
SVM	MacF1	**89.6(0.5)**↑	**66.5(1.4)**↕	**41.2(3.2)**↑	**64.4(1.4)**↕	**75.0(0.9)**↓
	MicF1	**89.9(0.6)**↑	**79.9(1.4)**↕	**77.0(1.2)**↑	**76.1(0.6)**↕	**83.9(0.6)**↓
RF	MacF1	86.9(1.0)↕	62.6(2.9)↕	26.9(1.7)↕	58.5(0.9)↕	73.9(0.3)↕
	MicF1	87.1(1.1)↕	74.8(3.1)↕	68.4(1.2)↕	72.9(0.7)↕	83.4(0.1)↕
XF	MacF1	86.2(0.3)↕	59.6(3.4)↕	27.3(1.6)↕	55.8(1.1)↕	72.2(0.3)↕
	MicF1	86.4(0.4)↕	72.2(3.6)↕	69.2(0.9)↕	70.6(0.7)↕	82.0(0.1)↕
NB	MacF1	83.7(1.0)↑	58.1(2.5)↑	17.2(1.1)↑	58.9(0.2)↑	65.7(0.3)↑
	MicF1	83.8(1.1)↑	65.4(1.8)↑	40.1(3.6)↕	69.7(0.7)↑	80.5(0.1)↑
CENT	MacF1	86.6(0.9)↑	59.6(2.4)↑	34.5(1.3)↑	60.2(1.5)↑	71.3(0.4)↑
	MicF1	86.8(1.0)↑	67.2(2.0)↑	66.4(1.0)↑	70.7(1.8)↑	81.3(0.1)↑

Table 3.8: Average MicroF$_1$ and MacroF$_1$ of different classifiers using the SPEA2fast feature selection strategy. ↑, ↓ and ↕ correspond, respectively, to statistically significant gains, losses and without significances over the set of all meta-features on the same classifier. Despite the effectiveness benefits of selecting meta-features for NB and CENT, SVM is consistently the best classifier for all datasets.

MF	SVM	NB	CENT	RF
cos_knn	✓			✓
cos_cent	✓		✓	✓
l1_knn				✓
l2_knn	✓			
l2_cent	✓		✓	✓
cnt	✓			✓
ncnt	✓	✓	✓	✓
qrt	✓			✓
fisher			✓	✓
ig				✓
sum_cent	✓			✓
cqrt	✓		✓	✓

Table 3.9: Individuals (columns) found by the SPEA2fast algorithm for each classifier. Each column represents the best individual found for a classifier, where the ✓cells indicate a selected group for this individual.

estimate Gaussian probabilities from this normalized meta-feature group. CENT uses all meta-features that include global class distributions, and RF is capable of exploiting information from most meta-features thanks to its internal selection mechanism that evaluates the best features to build decision trees.

To further evaluate the best combination of meta-feature groups for XF, NB and CENT, we use the brute-force method executed with these fast classifiers in each dataset. Since SVM is the best method to classify meta-features, we use the SVM method to classify documents represented with the combinations of meta-feature groups that correspond to each classifier in Table 3.10. As expected, the best meta-feature groups for the classifiers XF, NB and CENT do not achieve results as effective as SPEA2fast or Brute-force (SVM) in most datasets. This was somewhat expected as the best combination of meta-features found by each classification approach is biased towards their respective classification paradigm, as previously discussed. This provides evidence for the benefits of using the multi-objective SPEA2fast for the joint

exploitation of different classification methods.

		20NG	4UNI	REUT90	ACM	MEDLINE
SPEA2fast	MacF1	**89.6(0.5)**	**67.0(2.5)**	**41.2(3.2)**	**64.4(1.4)**	**75.0(0.9)**
	MicF1	**89.9(0.6)**	**79.6(0.9)**	**77.0(1.2)**	**76.1(0.6)**	**83.9(0.6)**
Brute-force (SVM)	MacF1	**89.7(0.5)**	**66.3(1.0)**	*	**64.6(1.2)**	*
	MicF1	**90.1(0.7)**	**79.8(1.2)**	*	**76.4(0.9)**	*
Brute-force (XF)	MacF1	88.9(0.7)	**66.7(1.5)**	35.3(2.4)	**64.2(0.9)**	**75.1(0.7)**
	MicF1	89.0(0.7)	**79.5(1.2)**	73.2(1.5)	**76.2(0.7)**	**84.0(0.6)**
Brute-force (NB)	MacF1	84.7(1.3)	55.3(1.4)	31.2(0.6)	60.8(2.1)	70.9(0.7)
	MicF1	84.9(3.3)	69.5(1.4)	72.9(2.3)	72.0(0.2)	80.4(0.3)
Brute-force (CENT)	MacF1	89.2(0.3)	57.3(1.9)	38.5(1.4)	60.8(2.2)	73.2(0.4)
	MicF1	89.3(0.4)	71.3(2.2)	72.2(2.7)	72.1(0.2)	82.4(0.2)

Table 3.10: Average MicroF$_1$ and MacroF$_1$ of the SVM classifier executed on meta-feature groups found by variations of the Brute-force executed with different classification methods and the SPEA2fast method. As expected, the best meta-feature groups for the classifiers XF, NB and CENT are not usually effective on SVM.

3.3.5 Core Combinations of Meta-Feature Groups (Q4)

In this section, we analyze the Pareto frontier produced by maximizing the classification effectiveness and minimizing the number of meta-feature groups in the task of finding combinations of meta-feature groups. As we shall see, this analysis is capable of providing information about the discriminative power of specific combinations of meta-feature groups considering the nuances of each dataset. Since SPEA2SVM achieved the best results in our meta-feature selection experiments, we start comparing Brute-force with SPEA2SVM to provide evidence that SPEA2SVM is an effective tool to find the Pareto frontier in Section 3.3.5.1. Then, we analyze the Pareto Frontier of each individual dataset in Section 3.3.5.2. Finally, Section 3.3.5.3 finishes our analysis showing core combinations that provide the most discriminative information considering multiple datasets at the same time.

3.3.5.1 SPEA2SVM Pareto Frontier

The Brute-force strategy is very slow and it is not feasible as a mechanism to find the Pareto frontier in datasets such as REUT and MEDLINE. Therefore, we start by providing evidence that SPEA2SVM might be as effective as the Brute-force method for the task of finding the Pareto frontier using our smaller/less complex datasets.

Instead of evaluating all possible combinations, SPEA2SVM aims at driving the search to non-dominated individuals and improving their results in successive generations. Figure 3.4 shows the distance[7] between the Pareto frontier found by Brute-force and the Pareto

[7]We use the average Euclidian distance in the objective space between each individual of the Brute-force Pareto frontier and its respective closest individuals (as previously suggested (Czyzak and Jaszkiewicz, 1998)) of a SPEA2SVM generation.

frontier from each generation of SPEA2SVM on 20NG, 4UNI and ACM. In fact, considering only a few generations (less than 30), the Pareto frontiers found by the SPEA2SVM are very close to the Pareto frontier found by the Brute-force method.

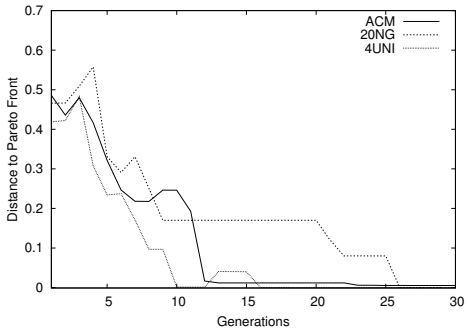

Figure 3.4: Distance between the Pareto frontier found on each generation of SPEA2SVM and the Pareto frontier found by the Brute-force method.

To be more specific, let us turn our attention to the Pareto frontier[8] found after executing all the iterations of the SPEA2SVM and the Pareto frontier found by the Brute-force method. Table 3.11 represents such Pareto frontier, in which the numbered columns represent the number of meta-feature groups in the non-dominated individual, and lines represent the individual's effectiveness. Table 3.11 shows that considering the same number of meta-feature groups, the effectiveness of all individuals in the Pareto frontier found by SPEA2SVM are equivalent (without statistically significant differences) with the individuals found by Brute-force. This indicates that despite the fact that SPEA2SVM does not evaluate all possible combinations, the individuals found by SPEA2SVM on 20NG, 4UNI and ACM are as good as the ones found by Brute-force. These results motivate the adoption of SPEA2SVM as an analysis framework.

3.3.5.2 Core Combinations of Meta-Feature Groups on each Dataset

In our next experiments, we provide a visualization of the individuals in the Pareto frontier. This visualization guides our analysis by showing the meta-feature groups in the individuals and by providing evidence to evaluate the core discriminative information in them. However, it is worth pointing out that one individual may generate different results on different samples of the same dataset due to the variance of the effectiveness scores predicted by the models being used. Consequently, there are different Pareto frontiers with different samples from the dataset. Since we execute SPEA2SVM five times for each dataset (due to the 5-fold cross

[8]Individuals with more meta-feature groups tend to be dominated by other individuals. Therefore, datasets such as 20NG and ACM contain only a few individuals with small number of meta-feature groups in their Pareto frontier.

		Number of Meta-Feature Groups						
		1	2	3	4	5	6	7
20NG	SPEA2SVM	88.5(0.7)	89.5(0.5)	89.7(0.6)	*	*	*	*
	Brute-force	88.5(0.8)	89.6(0.5)	89.8(0.5)	*	*	*	*
4UNI	SPEA2SVM	58.4(2.5)	61.1(1.7)	63.9(1.4)	65.8(1.7)	66.1(1.6)	66.4(1.6)	66.6(2.1)
	Brute-force	58.4(2.5)	61.2(1.1)	64.1(3.0)	66.0(2.8)	66.2(1.6)	66.3(1.8)	66.3(1.0)
ACM	SPEA2SVM	61.9(1.8)	62.7(1.9)	64.5(1.9)	64.9(1.4)	*	*	*
	Brute-force	62.1(1.4)	62.6(1.5)	64.6(1.9)	64.6(1.2)	*	*	*

Table 3.11: Pareto frontiers found by Brute-force and SPEA2SVM on 20NG, 4UNI and ACM. Each column corresponds to the individuals in the Pareto frontier sorted in ascending order by the number of meta-feature groups in each individual and their average MacroF1 effectiveness (with their 95% confidence intervals). *represents the absence of an individual in the Pareto frontier.

validation methodology), we show in the following tables only the most common Pareto frontier of different SPEA2SVM executions. This enables us to analyze a representative visualization of the Pareto frontier found by the SPEA2SVM method. In the following tables, each column corresponds to an individual in the Pareto frontier, where the ✓cells indicate a selected group for this individual, and the blank cells indicate the absence of a group. Since we consider as objectives the minimization of meta-feature groups and maximization of effectiveness, we order the columns according to the number of features, and present the average effectiveness results on MacroF1 and MicroF1 in the last two rows.

Table 3.12 represents the Pareto frontier on 20NG. The individual containing only the meta-feature group *qrt* is capable of achieving results comparable to the ones obtained with the combination containing all groups of meta-features. This indicates that the compact set of cosine similarities between an example and its neighbors provided by *qrt* is enough to provide the most discriminative information for this dataset. We argue that in balanced datasets, such as 20NG, the local information provided by the closest training examples from each category (exploited by *qrt*) is a reliable source of discriminative patterns, since there is enough training examples from each category to provide low variance statistics about neighbors from each one of them. The individual with the two meta-features *qrt* and *cos_cent* is capable of producing statistically significant gains over *qrt*, which indicates that *qrt* and *cos_cent* provide complementary information to each other. In fact, previous works (Gopal and Yang, 2010; Yang and Gopal, 2012) only exploit these two kinds of information in order to achieve effective results. The inclusion of other kinds of information is not able to provide statistically better results on 20NG. In fact, they can even unnecessarily increase the complexity of the meta-feature space, leading the classification model to overfit due to the small number of training examples per category. In fact, SPEA2SVM is not able to find any combination of meta-features containing more than three meta-feature groups.

Unlike the results on 20NG, the removal of meta-features does not produce improve-

MF	Number of Meta-Feature Groups		
	1	2	3
cos_knn			✓
cos_cent		✓	✓
l1_knn			
l2_knn			
l2_cent			
cnt			
ncnt			
qrt	✓	✓	
fisher			✓
ig			
sum_cent			
cqrt			
MacF1	88.5(0.7)	89.5(0.5)	**89.8(0.6)**
MicF1	88.7(0.7)	89.7(0.5)	**90.1(0.7)**

Table 3.12: The three individuals of the 20NG Pareto Frontier.

ments in effectiveness on 4UNI. Table 3.13 shows that the inclusion of new meta-feature groups, beyond *qrt* and *cos_cent*, is capable of improving the effectiveness in various situations. We argue that besides the robust model learned from informative data to ignore redundant and noisy information, the inclusion of new discriminative information provided by the other meta-feature groups might in some cases overcome the potential harm of including noise associated with this new information.

Specific characteristics of categorizing web pages from 4UNI might induce the need for complementary information provided by different meta-feature groups. Specifically, the distribution of categories is very unbalanced and the largest category corresponds to general academic web pages, which is a collection of pages that were not assigned as the "main page". Instances from each category might find very close pages to the category of general academic web pages, which impairs the discriminative information provided by the meta-feature group *cos_knn*. The inclusion of the meta-feature groups *sum_cent* and *ig* provide additional evidence to evaluate the discriminative power of the neighbors, which can be useful in cases where the similarity between neighbors provided by *cos_knn* is not enough to provide discriminative evidence. The inclusion of other meta-feature groups, such as *cnt*, *l*1*_knn* and *cqrt* also provide new discriminative information from the neighborhood.

Using only five meta-feature groups (*cos_knn*, *cos_cent*, *ig*, *sum_cent* and *cnt*) it is possible to obtain equivalent results (without statistically significant differences) to the set of all meta-features. Moreover, the use of only one, two, three and four meta-feature groups is statistically inferior to the combination of five groups. Particularly, the use of the *cos_knn* alone provides most of the discriminative information by exploiting the local distribution of classes despite its previously discussed limitations. Considering two meta-features, the most effective combination contains *cos_knn* and *cos_cent*. As we shall see in all datasets, when using only two meta-feature groups, there is always one of them that exploits local and other

MF	Number of Meta-Feature Groups						
	1	2	3	4	5	6	7
cos_knn	✓	✓	✓	✓	✓	✓	✓
cos_cent		✓	✓	✓	✓	✓	✓
l1_knn						✓	✓
l2_knn							
l2_cent							
cnt					✓	✓	✓
ncnt							
qrt							
fisher							
ig				✓	✓	✓	✓
sum_cent			✓	✓	✓	✓	✓
cqrt							✓
MacF1	58.4(2.5)	61.5(1.9)	63.4(2.7)	65.1(1.8)	66.3(1.5)	**66.7(1.3)**	**67.0(1.4)**
MicF1	72.1(1.5)	75.9(1.2)	78.2(1.5)	79.1(1.4)	79.9(1.2)	80.3(1.0)	**80.6(1.2)**

Table 3.13: Individuals in the 4UNI Pareto Frontier.

that provides global information.

REUT is the only dataset which a single meta-feature group is more effective than the combination of all meta-features (see Table 3.14). In fact, classification models built from this dataset are very sensitive to overfitting due to complex and noisy meta-feature spaces, as this dataset has the smallest number of training examples per category (on average). The straightforward raw distances provided by *cos_knn* are capable of providing a simple and informative meta-feature space for building effective classification models for REUT. The complementary information provided by *cos_cent* makes the combination of *cos_knn* and *cos_cent* slightly superior to *cos_knn* alone, with statistically significant gains on MicroF1 and MacroF1. The inclusion of any meta-feature group to the combination of *cos_knn* and *cos_cent* does not provide any significant gains.

MF	Number of Meta-Feature Groups		
	1	2	3
cos_knn	✓	✓	✓
cos_cent		✓	✓
l1_knn			
l2_knn			✓
l2_cent			
cnt			
ncnt			
qrt			
fisher			
ig			
sum_cent			
cqrt			
MacF1	40.5(2.1)	**41.5(2.9)**	**41.5(3.1)**
MicF1	75.7(0.5)	**77.3(1.7)**	**77.4(1.5)**

Table 3.14: Individuals in the REUT Pareto Frontier.

The task of classifying academic publications from ACM is not trivial, since specific papers of one area might be very close to papers from other areas. In fact, Table 3.15 shows

that local information provided by *cos_knn* does not appear to be enough for classifying these hard cases. The set with meta-features *cos_cent* and *cos_knn* is capable of improving the results of *cos_knn* due to the inclusion of the global information, but is statistically inferior to the combination containing all meta-features. In order to obtain close results (indeed without statistically significant differences) with the combination of all meta-features, we need an additional kind of information provided by *sum_cent*. Specifically, *sum_cent* is capable of providing comparisons between local and global information from all pairs of categories. This information provides specific evidence about how a neighborhood is distant from the distribution of all documents from each category. Considering the specificities of the ACM, *sum_cent* provides specific evidence to identify that, if the closest publications of a paper p are not from the same area as p, they might be outliers of their respective areas.

MF	Number of Meta-Feature Groups			
	1	2	3	4
cos_knn	✓	✓	✓	✓
cos_cent		✓	✓	✓
l1_knn				
l2_knn				
l2_cent				
cnt				✓
ncnt				
qrt				
fisher				
ig				
sum_cent			✓	✓
cqrt				
MacF1	62.1(1.3)	62.6(1.7)	**64.5(1.8)**	**64.8(1.6)**
MicF1	74.0(1.2)	75.4(0.3)	**76.1(0.6)**	**76.1(0.3)**

Table 3.15: Individuals in the ACM Pareto Frontier.

The previously discussed problems in classifying academic publications might also occur in the classification of publications from the MEDLINE digital library. In fact, the specific evidence exploited by *sum_cent* also provides complementary discriminative information for MEDLINE, since our best results in Table 3.16 always include *sum_cent*.

The combination of the four meta-feature groups *sum_cent*, *l*2*_knn*, *cos_knn* and *cos_cent* achieve the best results in terms of MicroF1, which indicates that *sum_cent* complements the local information provided by *l*2*_knn* and *cos_knn* and global information of *cos_cent*. The inclusion of more meta-feature groups to this individual provides only marginal (but statistically significant) improvements on MacroF1 due to specificities of different categories.

It is important to notice that the use of different similarity measures can provide significant improvements in MEDLINE. In fact, the individual with three meta-features in Table 3.16 contains *l*2*_knn* combined with *cos_knn* and *cos_cent*, indicating the benefits of using *l*2 distance along with cosine similarity. We argue that the high number of documents

in MEDLINE helps to provide enough information to exploit the high variance of the *l*2 distance measure, as *l*2 is not normalized according to the compared document lengths.

MF	Number of Meta-Feature Groups						
	1	2	3	4	5	6	7
cos_knn		✓	✓	✓	✓	✓	✓
cos_cent		✓	✓	✓	✓	✓	✓
l1_knn							✓
l2_knn			✓	✓	✓	✓	✓
l2_cent						✓	✓
cnt					✓	✓	✓
ncnt	✓						
qrt							
fisher							
ig							
sum_cent				✓	✓	✓	✓
cqrt							
MacF1	70.9(0.7)	73.7(0.3)	74.9(0.4)	75.6(0.5)	75.7(0.3)	75.7(0.5)	**75.9(0.6)**
MicF1	81.4(0.3)	83.4(0.1)	84.2(0.2)	**84.7(0.5)**	**84.5(0.2)**	**84.5(0.5)**	**84.6(0.5)**

Table 3.16: Individuals in the MEDLINE Pareto Frontier.

3.3.5.3 Core Meta-Features Considering All Datasets

In this experiment, we aim at finding core combinations of meta-features capable of achieving good results across all datasets. In order to do so, we use the maximization of the average effectiveness in all the six datasets and the minimization of the number of meta-feature groups as objectives in our multi-objective optimization strategy SPEA2SVM. After obtaining the Pareto frontier for these objectives, we select individuals with the best average effectiveness on all datasets.

Table 3.17 presents the selected individuals considering the number of meta-feature groups in each individual. We start by noticing that *cos_knn* is the best isolated meta-feature group and the best combination of only two meta-feature groups contains *cos_knn* and *cos_cent*. As previously discussed, this combination of two groups is capable of achieving the best results on 20NG and REUT and provide most part of the discriminative information for other datasets as they are complementary to each other, providing high quality information from the raw cosine similarity scores. In fact, *cos_knn* and *cos_cent* are part of most combinations containing more than two meta-feature groups. The only exception is when using three groups, where *qrt* replaces *cos_cent*. In this case, both *qrt* and *cos_cent* provide the raw cosine similarities as features, but *qrt* is a more compact version of *cos_cent*.

The combination of three groups includes the l2 distance to complement the information of the cosine similarity. This change only benefits the MED dataset and does not provide statistically significant gains on the remaining datasets. The combination of four groups includes discriminative information provided by *sum_cent*, which provides statisti-

		Number of Meta-Feature Groups						
		1	2	3	4	5	6	7
MF	cos_knn	✓	✓		✓	✓	✓	✓
	cos_cent		✓	✓	✓	✓	✓	✓
	l1_knn							✓
	l2_knn			✓	✓	✓	✓	✓
	l2_cent						✓	✓
	cnt					✓	✓	✓
	ncnt							
	qrt			✓				
	fisher							
	ig							
	sum_cent				✓	✓	✓	✓
	cqrt							
20NG	MacF1	88.5(0.8)	**89.6(0.5)**	**89.5(0.5)**	89.3(0.4)	88.9(0.3)	89.0(0.4)	89.0(0.4)
	MicF1	88.7(0.9)	**89.9(0.6)**	**89.8(0.5)**	89.6(0.4)	89.1(0.3)	89.2(0.5)	89.2(0.5)
4UNI	MacF1	58.4(2.5)	61.0(1.6)	60.4(2.4)	64.7(1.5)	**66.0(2.2)**	**65.4(1.6)**	**65.8(1.5)**
	MicF1	72.1(1.5)	75.6(1.1)	76.3(1.5)	78.9(1.0)	**79.4(0.9)**	**79.2(0.9)**	**79.6(1.2)**
REUT	MacF1	40.5(2.1)	**41.6(2.6)**	40.1(3.1)	32.5(3.2)	32.7(3.0)	32.5(2.9)	33.3(2.8)
	MicF1	75.7(0.5)	**77.6(0.5)**	**77.1(1.5)**	70.7(1.5)	70.9(1.2)	71.1(1.3)	71.4(1.5)
ACM	MacF1	62.0(1.4)	63.0(2.2)	63.4(1.3)	64.5(1.9)	**65.0(1.5)**	**65.0(1.6)**	**65.1(1.6)**
	MicF1	74.0(1.2)	75.6(0.5)	75.7(0.4)	**76.1(0.4)**	**76.1(0.6)**	**76.1(0.5)**	**76.2(0.5)**
MED	MacF1	69.9(0.4)	73.7(0.3)	74.8(0.3)	75.6(0.3)	75.7(0.5)	75.7(0.5)	**75.9(0.4)**
	MicF1	82.2(0.2)	83.4(0.1)	84.1(0.2)	**84.7(0.2)**	**84.5(0.2)**	**84.5(0.2)**	**84.6(0.2)**

Table 3.17: Individuals chosen from the Pareto frontier according to the higher average effectiveness on different datasets.

cally significant improvements on MED, ACM, and 4UNI, but substantial losses on REUT. We argue that despite the additional information provided by *sum_cent* for most datasets, it is also potentially noisy for small datasets containing various categories. In fact, *sum_cent* is the only meta-feature group that grows quadratically with the number of categories of the dataset. Since REUT is the dataset with the largest number of categories, *sum_cent* produces various potentially noisy meta-features for this dataset, incurring in overfitting. The only dataset that can benefit from the use of more than the four aforementioned meta-feature groups is 4UNI. As previously discussed, this dataset can use various kinds of discriminative information about the neighborhood to treat its particular characteristics.

Table 3.17 shows that there is no combination of meta-feature groups capable of achieving the best results in all datasets. Despite the differences among datasets, the combination using only the two meta-feature groups *cos_knn* and *cos_cent* is the best combination on REUT and 20NG, also achieving good results in general. In fact, the best combination found for both ACM and MED datasets are slightly superior (about 1% in MicroF1 and 3% in MacroF1) to this combination of two groups. We argue that the combination of *cos_knn* and *cos_cent* provides core discriminative information for classification in the evaluated datasets by directly exploiting local and global distribution of raw cosine similarities, as they also appear among the individuals in the Pareto frontier of previous experiments on each dataset. The inclusion of a few other meta-feature groups enables the exploitation of additional information, especially on 4UNI. In fact, the combination of the five meta-feature

groups *cos_knn*, *cos_cent*, *sum_cent*, *l2_knn*, and *cnt* is capable of providing additional information from the relationship between the category centroids and neighborhood (through *sum_cent* meta-features), *l*2 distances and the normalized cosine similarity scores. This core combination of meta-features is capable of providing almost all discriminative information from the evaluated meta-features, since they are capable of achieving the best results on most datasets, except in REUT and 20NG. These results provide evidence towards the use of *cos_knn* and *cos_cent* when dealing with shortage of information per category, and the use of *l*2*_knn*, *sum_cent*, *cos_knn*, *cos_cent* and *cnt* otherwise.

Chapter 4

Distance-based Meta-features Enriched with Label Information

Despite the success of the previously discussed distance-based meta-features, the considered underlying distance relationships rely on traditional distance measures among documents. These distances aim at summarizing discriminative evidence based on simple manipulations of term weights (such as TF-IDF), which might thwart the importance of relevant discriminative terms in the similarity computation, emphasizing irrelevant terms. Also, distance measures such as Cosine, Euclidean and Manhattan are not designed to capture whether two documents belong to the same class and thus do not directly associate similarity with class information.

In this chapter, we present new strategies to exploit distance relationships for meta-feature generation. In fact, instead of relying on the selection of meta-features that directly exploit traditional distances between documents, we propose new meta-features that adapt the exploitation of distance relationships for each dataset using machine learning techniques. Particularly, we tackle the limitations of previous strategies by proposing two types of potentially complementary meta-features that correlate a set of pairwise document similarity evidence with the likelihood of these documents belonging to the same class. Using supervised strategies to enrich distance relationships with labeled data, we propose the following meta-features:

Distance-based meta-features from Synthetic Document Representations (SDRs). The first type of meta-features we propose uses SDRs built from similarity evidence (e.g., common words among documents and similarity measures) found on nearby documents to correlate pair of documents with classes. As illustrated in Figure 4.1, the common words between two documents and the similarity scores between them provide features for the resulting SDR.

SDRs and the class labels of the pairs of documents originating them are used to generate a training collection. A SDR in this collection is labeled as positive if the pair belongs to the same class and as negative, otherwise.

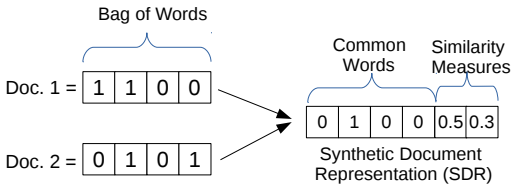

Figure 4.1: SDR built from similarity evidence.

A predictor (in our case, an SVM classifier) is learned from this "synthetic collection", producing a hyperplane able to separate positive from negative SDRs (as described in details in Section 4.1.2). In other words, an effective predictor produces high scores (i.e., high distances from the hyperplane) when there is compelling similarity evidence to assert that two original documents in the pair belong to the same class.

Once the predictor is learned, we can generate our first type of meta-features for a given document as follows. Given a target document t on the original collection, we first obtain the neighbors of t belonging to the original training set (i.e., neighbor documents whose classes are known). For each neighbor n, we obtain a SDR formed by n and t. Next, we compute the distance between each resulting SDR and the hyperplane previously obtained. These distances correspond to our first type of meta-features for the target document.

The left part of Figure 4.2 illustrates the generation of this type of meta-feature for a target document represented in the figure by a circle with a "?" inside. The circles connected to the target by a line represent the neighbors of ? in the original training set. There are four neighbors in this example. Each pair $(?, neighbor)$ produces a single SDR using similarity evidence shared by the two documents in the pair. The SDR production is emphasized by the circled pair pointing (with a dashed arrow) to a SDR on the right side of the figure. The four distances between the hyperplane and the SDRs (represented as double-arrow lines in the right part of the figure) correspond to the proposed meta-features for document "?". These hyperplane distances directly correlate with the discriminative power of the similarity evidence contained in each SDR. The higher the distances, the stronger the evidence. In sum, instead of relying on the cosine similarity score of pairs $(?, neighbor)$ to produce meta-features (i.e., using only the left part of Figure 4.2), we propose to use the hyperplane distances corresponding to such pairs, which can better assess the discriminative power of the similarity evidence.

Error rate based meta-features. An important aspect of the process to construct the proposed meta-features is that it allows us to identify hard-to-classify examples. Given a target document, it is possible to identify if it is hard-to-classify by analyzing both the proportion of errors in the predictions for SDRs formed from the target document and the differences in the distances among these SDRs and the hyperplane. For example, consider the four SDRs formed from the target "?" and its neighbors in Figure 4.2. The one represented by a green-border square positioned above the hyperplane was wrongly classified by the predictor. Since this SDR was formed by similarity evidence between the target document and a neighbor with label "1", the predictor was "fooled" by the similarity evidence in the SDR. Thus, the proportion of SDRs wrongly classified by the predictor provides evidence regarding hard-to-classify documents. Accordingly, such proportion (of incorrectly classified SDRs) corresponds to the second type of proposed meta-features.

Notice that this second type of meta-feature works as an evaluation of the quality of the first type of meta-feature. Whenever the former meta-features indicate that a document is difficult to classify, supplementary information about the neighborhood of a target document is needed. In this case, we propose to combine our proposed meta-features based on SDRs with an extended version of the two meta-feature groups (cos_knn and cos_cent) that presented core information for text classification considering all evaluated datasets in Chapter 3.

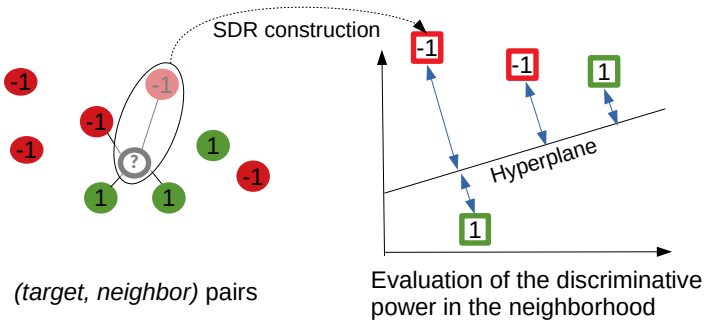

Figure 4.2: Evaluating the discriminative power of similarity evidence among the neighbors of a target document with a hyperplane as a predictor. Hyperplane distances provide evidence for new meta-features to represent the target document. Circles and squares indicate original and SDRs, respectively. Labels -1 and 1 indicate their associated category.

Different sets of the proposed meta-features can be obtained if we use different training sets or different algorithms to find the separating hyperplane for the SDRs. Particularly, we construct two different kinds of hyperplanes to evaluate SDRs by using different training sets. The first kind uses SDRs produced with all training documents as training examples to produce a hyperplane. In this scenario, the global similarity information related to all train-

ing documents is explored when learning the hyperplane, and the distances between a SDR and the hyperplane reflect the use of all training information. The second kind of hyperplane is inspired on the SVM-kNN method (Zhang et al., 2006). We build local hyperplanes using only the nearest SDRs of a target document as training examples. In this scenario, the hyperplane construction ignores potentially unrelated documents beyond the neighborhood of a document by locally adjusting the capacity of the SVM hyperplane to the properties of the training set in each localized area of the input space of SDRs.

Thus, we propose a meta-feature space that exploits not only distances from different hyperplanes, but also the identification of hard-to-classify examples and other statistics to replace (or extend) the original (bag-of-words) features of documents. Our experimental results in a large and heterogeneous set of datasets show significant improvements (up to 12%) of our proposal over a strong baseline constituted of the best literature meta-feature groups specially selected for each of the considered datasets. Such improvements helped our proposal to achieve the best results in *all tested* scenarios.

In sum, we defend that training documents can provide additional information to meta-features with a supervised evaluation of similarity evidence. In order to defend such claim, we propose: (i) the design and evaluation of new meta-features based on SDRs and distances to hyperplanes especially designed to learn and evaluate discriminative similarity evidence; (ii) a new set of meta-features for estimating the level of error introduced by the newly proposed and the existing meta-features, specially for hard-to-classify regions of the feature space and (iii) an analysis of the effects of different groups of meta-features on classification effectiveness.

4.1 Newly Proposed Meta-Features

In here, we provide the necessary details for building the meta-features newly proposed in this chapter. We first describe how SDRs are built followed by the process of building the first type of meta-features derived from these SDRs. Next, we show how to extend previously proposed meta-features to exploit potentially complementary information from traditional distance measures. Finally, we detail a set of meta-features designed to evaluate the discriminative information produced by the two previous types of meta-features. Such meta-features are useful to identify hard-to-classify documents, being of great importance when learning robust high quality classification models.

4.1.1 SDRs

Classification effectiveness based on meta-features significantly depends on the similarity evidence used to evaluate different pairs of documents. In order to further advance their potential, we propose to take advantage of the discriminative similarity evidences explicitly captured by SDRs.

Similarity evidences are formally defined as follows. Let $\mathscr{X}$ and $\mathscr{S}$ denote the bag-of-words feature space and the SDR feature space, respectively. The similarity evidence corresponding to the pair of documents $\vec{x}_a, \vec{x}_b \in \mathscr{X}$ is denoted by the SDR $\vec{s}_{ab} \in \mathscr{S}$ expressed as the concatenation of the sub-vectors below:

- $\vec{v}^{common} = [min(\vec{x}_a, \vec{x}_b)]$: A $|\mathscr{X}|$-dimensional vector s.t. each element w in $\vec{v}^{common}$ corresponds to $min(\vec{x}_{a_w}, \vec{x}_{b_w})$, where $\vec{x}_{a_w} \geq 0$ and $\vec{x}_{b_w} \geq 0$ correspond to the TFIDF weights of word w in documents $\vec{x}_a$ and $\vec{x}_b$, respectively[1]. This vector provides a new sparse representation that corresponds to the common information among two documents. This high-dimensional, fine-grained information might identify important individual common similarity evidence that appear in both documents.

- $\vec{v}^{cos} = [\cos(\vec{x}_a, \vec{x}_b)]$: A 1-dimensional vector produced by the cosine similarity between $\vec{x}_a$ and $\vec{x}_b$.

- $\vec{v}^{cent} = [min(\cos(\vec{x}_a, \vec{x}_j), \cos(\vec{x}_b, \vec{x}_j))]$: A $|\mathscr{C}|$-dimensional vector formed by the minimum cosine similarities between $\vec{x}_a$ or $\vec{x}_b$ and each one of the x_j category centroids in the training dataset. It captures explicit similarity evidence that relates both $\vec{x}_a$ and $\vec{x}_b$ documents to categories.

Since most documents in a given set of training documents $\mathbb{D}_{train}$ usually do not present meaningful similarity evidence with a target document t, SDRs are built using only the k nearest neighbors of t in $\mathbb{D}_{train}$. Algorithm 2 describes the construction of SDRs for t. It receives t, k and $\mathbb{D}_{train}$ as input and returns a set of SDRs S that represents the similarity evidence found on neighbors of t in a given set of documents $\mathbb{D}_{train}$. In Line 2, the algorithm finds the neighbors of t using the cosine similarity, which presented the best results for meta-feature generation in textual data (Canuto et al., 2018). Then, for each neighbor, the algorithm uses function $SyntheticDocumentRepresentation(t, n)$ in Line 4 to build a SDR for the pair (t, n) with the previously described similarity features. Therefore, the similarity evidence found on each neighbor is explicitly represented as features from its corresponding SDR.

4.1.2 Meta-Features based on SDRs (SYN)

After providing explicit similarity evidence in the form of SDRs, it is possible to learn a predictor that correlates the similarity evidence found in the pair of documents corresponding

[1] Whenever word w does not occur in a document $\vec{x}_a$, $\vec{x}_{a_w} = 0$.

Algorithm 2 *BuildSyntheticNeighbors*$(t, k, \mathbb{D}_{train})$

Input: Target document t, number of neighbors k and set of documents $\mathbb{D}_{train}$
Output: SDRs S for t

```
1 S ← ∅
2 N ← k nearest neighbors of x_t in D_train
3 foreach n ∈ N do
4     s_tn = SyntheticDocumentRepresentation(x_t, n)
5     S ← S ∪ {s_tn}
6 end
```

to a SDR s with the likelihood of these documents belonging to the same class. Thus, the predictor is able to estimate the relevance of similarity evidence to build more informed meta-features for effective text classification.

Algorithm 3 details how the training samples are processed to learn a predictor for SDRs. In Lines 1-2, the algorithm prepares the training data for the generation of an SVM hyperplane hw_c for each category c considering each training document $d \in \mathbb{D}_{train}$. Lines 5-7 generate SDRs that are positive training examples related to d using the subset of training examples $\mathbb{D}_{pos}$ of the same category as d. Note that there is no "else" after line 7, since we want to include negative training examples for documents of category c. In other words, these lines produce a set of SDRs $\mathbb{S}_{pos}$ with Algorithm 2 only for pairs of training documents that belong to the same class. On the other hand, Lines 8-9 produce a set of SDRs $\mathbb{S}_{neg}$ only for pairs of training documents that belong to different classes. Finally, in Line 11 an SVM classifier is trained using those positive and negative training samples ultimately defining a separating hyperplane. Such hyperplane is used as a predictor to estimate the likelihood of the similarity evidence in a SDR being related to category c.

Algorithm 3 Global hyperplanes for SDRs.

Input: Training set $\mathbb{D}_{train}$
Output: Hyperplanes hw_c for each category c

```
1  foreach category c do
2      D_pos ← ∅; D_neg ← ∅; S_pos ← ∅; S_neg ← ∅;
3      foreach d ∈ D_train do
4          if category of d = c then
5              D_pos ← documents of category c in D_train
6              S_pos ← S_pos ∪ BuildSyntheticNeighbors(d, k, D_pos)
7          end
8          D_neg ← docs. that are not of category c in D_train
9          S_neg ← S_neg ∪ BuildSyntheticNeighbors(d, k, D_neg)
10     end
11     hw_c ← TrainSVM(S_pos, S_neg)
12 end
```

After learning the hyperplanes hw_c, we use the distances among SDRs and the hyperplanes as meta-features that measure the similarity evidence between a target document and its neighbors. Algorithm 4 describes how meta-features $\vec{m}_t$ are generated for a target document t with the previously built hyperplanes hw_c. Lines 3-4 build the SDRs from the neighbors of t that belong to the training documents of category c. Using the generated

SDRs, the method *ComputeHyperplaneDistance* in Line 7 computes the normalized distance (with sigmoid function (Platt, 1999)) between each SDR $s \in S_c$ and the hyperplane hw_c. Such normalized distances correspond to the likelihood of the similarity evidence in each SDR in S_c being related to category c. It is worth noting that each SDR s corresponds to the similarity evidence between the target document t and one of its neighbors. Therefore, a high hyperplane (Euclidean) distance between s and hw_c corresponds to a high likelihood of t being related to c.

In Line 10, all the computed distances stored in the set H are sorted in ascending order, generating meta-features in $\vec{m}_{tc}$. This allows a learning method to compare the i-th greatest meta-feature value related to hperplane c of a document to the i-th greatest meta-feature value regarding the same hyperplane of another document during the learning process. Finally, in Line 11, the computed meta-features for class c are concatenated with the output vector $\vec{m}_t$ that contains meta-features for all categories.

It is important to notice that Algorithm 4 is used directly only for the corresponding SDRs of a test example. If applied to SDRs generated for documents in the training set $\mathbb{D}_{train}$, the meta-features generated from these SDRs would be biased to the training data, which consequently overfits the classifier. In order to generate meta-features for training documents, it is necessary to apply Algorithm 4 with cross-validation in the training set, where the hyperplanes hw_c are built from a subset of the training data, and the meta-features are generated for documents in the remaining subset.

Algorithm 4 Building Meta-features from SDRs using Global Hyperplanes (Synglob).

Input: Hyperplanes hw_c, target document t, training set $\mathbb{D}_{train}$
Output: Meta-features $\vec{m}_t$ for the document t

```
1  m_t ← []
2  foreach category c do
3      D_pos ← documents of category c in D_train
4      S ← BuildSyntheticNeighbors(t, k, D_pos)
5      H ← ∅
6      foreach s ∈ S do
7          h_dist ← ComputeHyperplaneDistance(s, hw_c)
8          H ← H ∪ {h_dist}
9      end
10     m_tc ← sort(H)
11     m_t ← concatenate(m_t, m_tc)
12 end
```

As previously mentioned, we also propose a version of our meta-features inspired on the SVM-kNN method (Zhang et al., 2006), which builds local hyperplanes using only the nearest SDRs of a target document as training examples. In this scenario, the hyperplane construction ignores potentially unrelated documents beyond the neighborhood of a document by locally adjusting the capacity of the SVM hyperplane to the properties of the training set in each area of the input space of SDRs. The construction of meta-features using such hy-

perplanes is illustrated in Algorithm 5, which differs from the Algorithm 4 by the fact that it builds each hyperplane using only the neighborhood of each target document, as illustrated in Lines 3-7. The construction of one hyperplane for each category of each target document is feasible because of the reduced number of training elements (only the neighbors). This naive implementation can be further improved with the combined use of kernel trick and DAGSVM (Zhang et al., 2006). Then the algorithm generates one meta-feature for each distance between the target document and the locally built hyperplane in Lines 8-9. The fact that SDRs are generated from training documents of all categories assures the evaluation of similarity evidence found on the relationship between t and neighbors from each category even on unbalanced training datasets.

Algorithm 5 Building Meta-features from SDRs using Local Hyperplanes (Synloc).

Input: Target document t, training set $\mathbb{D}_{train}$
Output: Meta-features $\vec{m}_t$ for the document t

```
1  m_t ← []
2  foreach category c do
3      D_pos ← documents of category c in D_train
4      S_pos ← BuildSyntheticNeighbors(t, k, D_pos)
5      D_neg ← docs that are not of category c in D_train
6      S_neg ← BuildSyntheticNeighbors(t, k, D_neg)
7      hw_c ← TrainSVM(S_pos, S_neg)
8      h_dist ← ComputeHyperplaneDistance(t, hw_c)
9      m_t ← concatenate(m_t, h_dist)
10 end
```

4.1.3 Extended version of literature Meta-features (EXT)

We extend the literature meta-features in two different ways. The first strategy exploits the space of original features and some literature meta-features using an SVM classifier. The correlation between both feature spaces are enriched with label information using the SVM classifier. The second strategy extends the centroid distances previously defined in Section 3.1 as cos_cent by evaluating the neighborhood in a projected meta-feature space. In our extensions, we focus on the two groups of literature meta-features which correspond to the vectors $\vec{v}^{cos}_{\vec{x}_f}$ and $\vec{v}^{cent}_{\vec{x}_f}$ (named as cos_knn and cos_cent in Chapter 3.1) built from the cosine similarity. These two meta-features were the only groups consistently among the individuals in the Pareto frontier of different datasets, as pointed out in Chapter 3.

Original features and meta-features (*Orig_ext*). Inspired by the success of previous works (Canuto et al., 2014) in combining the high-dimensional original input space $\mathscr{X}$ with distance-based meta-features, we here propose a compact set of meta-features capable of embodying the main benefits of such combination. This combination, named $\mathscr{X}_{MFextend}$, is an extended feature space that represents documents with the concatenation of their original document representation with core literature meta-features *cos_knn* and *cos_cent*. The

extended space $\mathscr{X}_{MFextend}$ enables the classifier that operates in such space to find interesting connections/relationships between meta-features and individual words. In this sense, $\mathscr{X}_{MFextend}$ combines the best of two worlds: summarized discriminative evidence about documents in the form of meta-features and very specific information about the individual words of the documents. In order to build a compact set of meta-features that exploits the relationship between the original and meta-features, we propose to exploit the distances between documents $\vec{x} \in \mathscr{X}_{MFextend}$ and SVM hyperplanes trained to categorize such documents. The resulting hyperplanes discriminate documents $\vec{x}$ according to the information from features and meta-features, which allows us to automatically evaluate the relationships between the two types of features in $\mathscr{X}_{MFextend}$. Particularly, we use a training set to generate one hyperplane per category. The meta-features for a test document are the normalized distances between the document and each hyperplane[2].

In sum, let $hwdist_c(\vec{x})$ be the hyperplane distance between a document $\vec{x} \in \mathscr{X}_{MFextend}$ and the hyperplane trained to categorize documents for category c. We define $Orig\_ext = [hwdist_c(\vec{x})]$ as a $|C|$-dimensional vector that contains the distance between $\vec{x}$ and the hyperplanes generated for each category $c \in C$. In order to avoid overfitting when generating these meta-features for training documents, we use cross-validation in the training set, where the hyperplanes are built from a subset of the training data, and the meta-features are generated for documents in the remaining subset.

Centroid-based meta-features (*cent_ext*). Our extension of centroid meta-features evaluates the neighborhood of documents in a low-dimensional space spanned by class centroids. Due to the already strong discriminative power of class centroids, our extension aims at enhancing them to better handle issues related to class imbalance and noisy terms in the original textual data representation. The strategy to extend centroid meta-features relies on two steps. In the first step, we replace the original input space $\mathscr{X}$ with a new space $\mathscr{M}_{cent}$ corresponding to the *cos_cent* meta-features. By doing so, documents that were represented as a bag-of-words are then represented as the compact set of centroid distances between the original document and each category centroid.

In the second step, we evaluate the neighborhood of each projected document $\vec{m} \in \mathscr{M}_{cent}$ using the same strategy described in Chapter 3.1 to generate the distance vectors $\vec{v}^{cos}_{\vec{x}_f}$. Therefore, we generate meta-features that correspond to the Euclidean distance between a projected document document $\vec{m} \in \mathscr{M}_{cent}$ and each neighbor in the projected meta-feature space of centroids. In other words, we generate a vector $cent\_ext = [dist(\vec{m}_i, \vec{m})]$, which is a $|\mathscr{C}| * k$-dimensional vector whose elements $dist(\vec{m}_i, \vec{m})$ denote the euclidean distance

[2]In order to generate meta-features for training documents, it is necessary to apply cross-validation in the training set, where the hyperplanes are built from a sub-set of the training data, and the meta-features are generated for documents in the remaining subset.

between $\vec{m}$ and the i^{th} nearest class c_j neighbor of $\vec{m}$ in $\mathscr{M}_{cent}$ space. The evaluation of the distribution of distances among neighbors in the projected space enables a deeper exploitation of centroids distances, since it takes into account the relationships between close centroids distances from different documents.

4.1.4 Error rate based Meta-features (ERR)

The main goal of error rate based meta-features is to estimate whether a target document needs additional information that complements meta-features built from SDRs or from our proposed extension of literature meta-features. We propose three strategies to evaluate prediction errors on the prediction of SDRs and documents in the previously described extended space $\mathscr{X}_{Orig\_ext}$, which are both used to generate meta-features relying on SVM predictions.

Error rate for SDRs (Err_syn). Let $\mathscr{S}_t$ and $\mathscr{S}_{tcorrect}$ denote the SDRs produced for a target document t using Algorithm 2 and let $\mathscr{S}_{tcorrect}$ be the number of correctly classified SDRs evaluated with the previously trained SVM models in Algorithm 3. We define $Err\_syn = [\frac{|S_{tcorrect}|}{|S_t|}]$ as a 1-dimensional vector produced by the proportion of correctly classified synthetic neighbors generated for t. A high proportion of correctly classified SDRs generated for t indicates that there is reliable similarity evidence provided by SDRs to classify it.

Error rate for extended meta-features (Err_ext). Similarly to the error rate of SDRs, we compute the proportion of correctly classified documents in the previously described extended space $\mathscr{X}_{Orig\_ext}$. In this scenario, we define $Err\_ext = [\frac{|N_{tcorrect}|}{|N_t|}]$ as a 1-dimensional vector produced by the proportion of correctly classified neighbors $N_{tcorrect}$ from all neighbors of a target document $\vec{t} \in \mathscr{X}_{Orig\_ext}$.

Discrepancy on literature-extended meta-features (Discr). We also evaluate discrepancies on scores of $\mathscr{X}_{Orig\_ext}$. The main idea is to evaluate how the hyperplane distance of a target document differs from the hyperplane distances of its neighbors Discrepancy meta-features were not generated for SDRs, as they are already focus on the discrepancies between a target document and its neighbors. Accordingly, the fact that a target document is as distant from a hyperplane as its neighbors correlates with the reliability of the evidence in $\mathscr{X}_{Orig\_ext}$ for such target document. Considering the vector of hyperplane distances $\vec{v}_{\vec{x}}^{hwdist}$ defined for a document x in Section 4.1.3, we define the discrepancy meta-features as $\vec{v}^{discrepancy} = [\vec{v}_{\vec{x}}^{hwdist} - \vec{v}_{\vec{x_{ij}}}^{hwdist}]$, which is a k-dimensional vector whose elements denote the difference between the hyperplane distance of $\vec{x}$ to a hyperplane and each hyperplane distance of its i^{th} nearest class c_j neighbor of $\vec{x}$.

4.2 Experimental Results

In this section we describe the experiments using the proposed meta-features. The experiments follow the same methodology of the experimental setup described in Chapter 3, which relies on 5-fold cross-validation to evaluate parameterization and average effectiveness (in $MacF_1$ and $MicF_1$), with with the best results and values that are not statistically inferior to the best (with 95% confidence) marked in **bold**. We start by evaluating the overall effectiveness results of different meta-feature spaces followed by experiments that evaluate the proposed groups meta-features in this chapter.

4.2.1 Effectiveness of the Proposed Meta-features (Q5)

We here present the effectiveness results of classifiers trained with the meta-features from different literature works and our proposed approach. Our goal is to show that our proposed strategies to enrich distance relationships with label information can improve the classification effectiveness, answering our fifth research question (Q5). Particularly, in the experiments of this section we compare the strategies proposed in this chapter with the following meta-feature baselines:

- Canuto et al. (2014), Gopal and Yang (2010) and Pang et al. (2015): Meta-level features proposed to exploit traditional distance measures, described in Chapter 3.

- SPEA2SVM (Canuto et al., 2018): The best combination of meta-features proposed to exploit traditional distance measures obtained for each dataset using the SPEA2SVM genetic algorithm described in Chapter 3.

- Bag-of-words + SPEA2SVM: Concatenation of SPEA2SVM meta-features with the original Bag-of-words representation. This baseline is capable of representing both individual words in the original feature space and meta-features, which enables the classification method to evaluate the relationship between both feature spaces and the exploitation of potentially complementary information among them.

- BOW-CNN (Johnson and Zhang, 2015): a recently proposed Convolutional Neural Network implementation specially designed for sparse and high dimensional text data. As our proposal, it uses label information to predict the importance of meta-features. However, the meta-features themselves are automatically generated in a convolutional layer of the neural network based on the co-occurrence of terms in documents. In terms of potential neural network baselines, we believe BOW-CNN is an good choice as it

takes as input the same representation as our meta-features, i.e.., BOW, and automatically generates internal "latent" representations that may be thought as automatically generated meta-features.

For this method, we kept the default parameterization of the implementation, except for the parameters learning rate and number of neighbors, which were chosen respectively, among five values from 0.1 to 0.5 and among six values from 500 to 3000 by using 5-fold cross-validation on each training dataset.

- The EXT+SYN+ERR (*Proposed*) meta-features corresponds to the union of the proposed SYN, EXT, and ERR meta-features described respectively, in Sections 4.1.2, 4.1.3 and 4.1.4.

		20NG	4UNI	REUT	ACM	MED
EXT+SYN+ERR *(Proposed)*	$macF_1$	**91.4(0.5)**	**74.4(1.8)**	**41.8(1.9)**	**67.3(1.2)**	**78.9 (0.6)**
	$micF_1$	**91.6(0.5)**	**83.0(0.6)**	**79.7(1.0)**	**77.9(0.3)**	**87.8 (0.4)**
Canuto et al. (2014)	$macF_1$	88.3(0.6)	66.1(2.6)	32.4(2.6)	64.1(1.1)	72.7(0.5)
	$micF_1$	88.5(0.6)	78.9(1.6)	71.5(0.9)	75.5(0.8)	82.5(0.2)
Gopal and Yang (2010)	$macF_1$	89.5(0.5)	60.6(2.7)	**41.7(2.8)**	62.7(1.4)	74.9(0.2)
	$micF_1$	89.8(0.6)	75.6(0.7)	77.9(1.2)	75.6(0.4)	84.2(0.1)
Pang et al. (2015)	$macF_1$	77.4(0.6)	56.4(1.8)	37.2(1.6)	52.1(1.6)	46.3(1.0)
	$micF_1$	78.3(0.7)	67.6(1.1)	71.8(0.8)	65.0(0.9)	66.3(1.0)
SPEA2SVM (Canuto et al., 2018)	$macF_1$	89.7(0.6)	66.5(1.4)	**41.5(3.1)**	64.9(1.4)	75.7(0.6)
	$micF_1$	90.0(0.7)	79.9(1.4)	77.4(1.5)	76.3(0.7)	84.4(0.5)
BOW-CNN (Johnson and Zhang, 2015)	$macF_1$	89.1(0.6)	68.0(1.0)	25.2(1.3)	59.6(0.3)	60.7(0.2)
	$micF_1$	89.3(0.7)	81.3(0.8)	70.9(0.6)	74.9(0.4)	82.5(0.4)
Bag-of-words	$macF_1$	87.8(0.2)	60.4(1.0)	29.5(2.1)	61.6(0.4)	76.0(0.2)
	$micF_1$	87.6(0.2)	70.7(0.8)	65.7(0.7)	72.1(0.5)	85.6(0.5)
Bag-of-words+SPEA2SVM	$macF_1$	89.5(0.4)	66.7(1.5)	**39.8(4.2)**	64.6(1.4)	76.8(0.2)
	$micF_1$	89.8(0.5)	77.7(1.5)	76.7(0.9)	76.5 (0.5))	86.1(0.3)

Table 4.1: Average effectiveness on different meta-features.

Table 4.1 shows the obtained values of $MacroF_1$ and $MicroF_1$ for the meta-features proposed in this chapter and the previously described baselines. As it can be seen, our proposed meta-features consistently achieve the best results in **all** evaluated datasets, a remarkable result. This provides evidence that the combination of meta-features described in Section 4.1 does produce more discriminative information than other distance-based meta-features in the literature, which rely on distance measures not designed to relate pairwise similarity evidence with categories.

The main difference between our proposal and the remaining methods is the identification of strong clues indicating that one particular neighbor contains important similarity evidence. Such clues may be associated to high prediction scores, which are most likely not "false positives" (Platt, 1999). We further exploit the confidence of predictions about similarity information thru ERR meta-features that provide evidence about the discriminative

information in our proposals. In fact, the proposed meta-features improved the results of previous works by Gopal, Canuto, Pang, and the combination of the best of them (SPEA2SVM) by 13%, 22%, 28%, and 12%, respectively. Our new meta-features achieved better results than the ones obtained by the traditional Bag-of-words and the combination Bag-of-words+SPEA2SVM in all datasets, including MED. This is specially important, since none of the previous meta-feature works can take advantage of the massive amount of data in MED to provide better results than Bag-of-words for this dataset.

BOW-CNN is a different strategy to exploit the bag-of-words representation. It has the potential advantage of the automated design of meta-features adapted to the dataset by the learning framework. Despite this potential advantage, our manually designed meta-features achieve significantly superior results than BOW-CNN. We conjecture that this is due mainly to (i) overfitting, as a joint effect of a complex model operating in a high dimensionality, which can be observed on the low error-rate for training data on early iterations of BOW-CNN, and (ii) skewness combined with large number of classes (between 7 and 90) and insufficient number of training documents per class in the smaller ones. This is corroborated by the poor results in MacroF1 of BOW-CNN in REU and ACM.

4.2.1.1 Group Evaluation

After evaluating the behavior of the combined use of all the proposed meta-features, we further analyze each component of our proposal. Table 4.2 shows the effectiveness of each meta-feature group in isolation in terms of MicroF1[3]. We first turn our attention to the SYN meta-features described in Section 4.1.2, which are based on the supervised evaluation of SDRs described by two algorithms: Algorithm 3 that generates *Synglob* meta-features, and Algorithm 4 that generates *Synloc* meta-features. *Synglob* meta-features always perform significantly better than *Synloc* as the latter only uses the limited information provided by the nearest neighbors of training examples, while *Synglob* takes advantage of the whole labeled data. *Synglob*, for instance, was the sole winner on 4UNI, achieving the best results among all groups in this dataset. The task of categorizing academic webpages in 4UNI is difficult as the general pages category is commonly mistaken by others. *Synglob* also appears among the top performers in 4 out of 5 datasets (the exception beinng REUT), a very strong and consistent performance. Particularly, in the case of REUT, it contains several classes with just a few training documents (less than 5 for 26 categories), which prevents the exploitation of SDRs in an effective way.

EXT meta-features described in Section 4.1.3 also obtained high effectiveness using different strategies to exploit information from similarity evidence. Particularly, *Orig_ext*

[3]Results with MacroF1 were qualitatively the same.

	SYN		EXT				ERR		
Dataset	*Synglob*	*Synloc*	*cent_ext*	*Orig_ext*	*cos_knn*	*cos_cent*	*Err_syn*	*Err_ext*	*discrepancy*
4UNI	**79.0(1.5)**	64.5(1.0)	71.6(0.9)	75.5(1.0)	71.4(0.6)	70.4(0.4)	45.1(1.2)	45.4(1.1)	69.0(1.3)
20NG	**88.8(0.9)**	72.7(0.7)	78.6(0.6)	87.7(0.2)	**88.5(0.6)**	81.1(0.4)	5.9(0.3)	5.3(0.1)	63.9(1.2)
ACM	**73.9(0.5)**	62.3(0.8)	68.9(0.6)	**74.3(0.4)**	**74.3(0.5)**	70.0(0.3)	24.3(0.5)	26.3(0.3)	60.7(0.4)
REUT	64.7(3.2)	56.8(1.3)	**76.3(0.6)**	69.9(1.5)	**76.1(0.7)**	74.7(0.7)	30.1(0.6)	29.7(0.5)	61.6(0.9)
MED	**84.9(0.6)**	72.9(0.4)	80.3(0.3)	**84.5(0.7)**	82.9(0.9)	79.9(0.3)	51.1(0.5)	52.9(0.4)	75.2(1.0)

Table 4.2: Average Micro-F1 effectiveness on each group of proposed meta-features.

produced high effectiveness results, in general, achieving the best results among EXT on 4UNI and MED with the combined exploration of original features and meta-features. Despite its benefits, *Orig_ext* suffers from potential generalization errors because of imbalanced classes and small training, as seen in REUT. Other EXT meta-features, namely *cent_ext* and *cos_cent* obtained significantly lower results in most datasets, as they are designed to complement other meta-features by exploiting only the global information related to class centroids.

Considering the ERR meta-features, we can see that they obtained the lowest results in general. In fact, they were designed to identify hard-to-classify documents based on meta-features, being not explicitly designed to provide direct evidence for classification. As such, their largest benefit should be observed when used in conjunction with other groups.

Since the groups SYN, EXT, and ERR were designed to explore different aspects and idiosyncrasies of the text classification task, we expect the presence of complementary information among them. In fact, there is clear empirical evidence to support such complementarity hypothesis. This is best seen with the combination SYN+EXT+ERR, which provides statistically significant superior results when compared to all other possible combinations of SYN, EXT, and ERR in all datasets, as shown in Table 4.3.

Particularly, the comparison between SYN+EXT+ERR and SYN+EXT highlights the importance of ERR meta-features to identify potentially hard-to-classify examples. Such identification provides means for a better optimization process during learning, improving the ability of SYN or EXT for categorizing hard-to-classify examples. This can also help mitigate potential noise or sampling errors from such examples during the model construction.

Disregarding ERR, the combination EXT+SYN is always superior to EXT or SYN in isolation. In fact, SYN and EXT exploit similarity evidence using different methods. Particularly, SYN meta-features take advantage of SDRs to directly express the relationship between categories and neighborhood-based similarity evidence. On the other hand, EXT meta-features summarize all the similarity evidence with cosine scores without exploiting the other components of the SDRs, especially the relationships with categories. Such significant (and complementary) differences on strategies to exploit similarity information justify the

		20NG	4UNI	REUT	ACM	MED
EXT+SYN+ERR	$macF_1$	**91.4(0.5)**	**74.4(1.8)**	**41.8(1.9)**	**67.3(1.2)**	**78.9 (0.6)**
	$micF_1$	**91.6(0.5)**	**83.0(0.6)**	**79.7(1.0)**	**77.9(0.3)**	**87.8 (0.4)**
SYN+ERR	$macF_1$	89.1(0.5)	70.4(3.3)	34.8 (2.1)	63.5(1.1)	76.5 (0.4)
	$micF_1$	89.3(0.4)	80.0(1.3)	71.8 (2.3)	75.6(0.4)	86.0 (0.3)
EXT+ERR	$macF_1$	88.3(0.4)	65.5(1.0)	36.8 (1.2)	63.1(1.5)	75.9 (0.4)
	$micF_1$	88.5(0.3)	78.7(0.7)	76.9 (0.8)	75.6(0.5)	85.9 (0.2)
EXT+SYN	$macF_1$	90.6(0.4)	70.5(2.0)	**39.4(0.6)**	64.4(0.9)	75.5 (0.7)
	$micF_1$	90.8(0.4)	80.2(0.8)	77.9(0.5)	77.0(0.5)	86.8 (0.5)
SYN	$macF_1$	88.3(0.7)	67.1(3.6)	25.8 (2.5)	59.4(0.6)	75.3 (0.7)
	$micF_1$	88.5(0.7)	79.2(1.5)	65.9 (2.9)	74.5(0.6)	85.1(0.5)
EXT	$macF_1$	88.2(0.2)	65.0(0.9)	37.3 (0.9)	63.7(0.7)	74.3 (0.7)
	$micF_1$	88.3(0.3)	78.6(0.8)	77.0 (0.7)	75.5(0.4)	85.7 (0.3)
ERR	$macF_1$	64.2(0.9)	49.1(2.3)	22.5(1.7)	41.2(1.1)	47.4 (1.3)
	$micF_1$	64.6(0.9)	68.3(1.3)	62.7(1.1)	60.6(0.6)	75.1 (1.2)

Table 4.3: Average effectiveness on each meta-feature group.

consistent and statistically significant gains ranging from 1% to 5% of EXT+SYN over the best results found either in EXT or SYN.

4.2.1.2 Importance of Groups with using $2^k r$ Factorial Design

We further analyze the importance of the three groups of meta-features, as well as their interactions, to explain the current results, using all 2^k possible combinations of groups for each dataset. We consider the case without any group using a "random" classifier, which returns an arbitrary category for each document. We also consider the replication of the experiments with each possible combination (using 5-fold cross validation) to evaluate the effects of uncontrollable external factors. We follow the standard quantitative approach called $2^k r$ factorial design (Jain, 1991) to analyze the effects of the individual groups of meta-features, as well as the effectiveness improvements produced by their interactions.

The first step to perform a $2^k r$ factorial design is to define the binary factors that may affect a response variable (e.g., Micro-F_1 score). In our case, each factor corresponds to the presence or absence of one group of meta-features. We show the presence of meta-features with their combined names (e.g., SYN:ERR indicates the presence of SYN and ERR).

Since we use five replications for each combination of meta-features, we estimate the residuals from the sum of the squared errors obtained from the replications. Then, we compute the 95% confidence intervals for residuals and effects from the variance on the results of our replications.

Table 4.4 presents the percentage of variation in the results that can be explained by each individual group of meta-features and by the interaction between groups of meta-features considering each possible combination. As we can see, the variations observed on all combinations can mostly be explained by the groups SYN and EXT in isolation and the iteration SYN:EXT. The effects of SYN and EXT each always account for more than 23%

	20NG	4UNI	REUT	ACM	MED
SYN	25.63	29.03	11.34	25.28	23.58
EXT	24.89	26.13	48.49	27.29	36.89
ERR	7.34	7.68	7.49	7.53	8.16
SYN:EXT	21.15	18.77	9.73	20.29	16.94
SYN:ERR	6.65	5.06	7.14	6.04	3.52
EXT:ERR	7.37	6.46	7.22	7.15	6.79
SYN:EXT:ERR	6.94	5.94	7.59	6.33	4.12
Residuals	0.03	0.94	0.98	0.09	0.01

Table 4.4: Explained percentage of result variation by individual meta-feature groups and interactions between them. The 95% confidence intervals are always inferior to 0.5%.

of all the effectiveness variation, as the presence of each one of these groups in isolation provides discriminative information for the text classification task. The iteration SYN:EXT also explains up to 21% of all variations in the results, highlighting the complementarity between SYN and EXT. Altogether, SYN, EXT and SYN:EXT explain more than 70% of the results in all analyzed datasets.

Other measured effects that present the interaction of ERR with other groups consistently explain statistically significant portions of the variation in the results, ranging from 4% to 8%. Such consistent variations, though relatively small in isolation, account altogether for about 17% of the total variation in the results. This provides strong evidence of the importance of ERR to improve the results of other meta-feature groups when interacting with them. But even in isolation, ERR can explain a rather interesting portion of the results (between 7%-8%). Finally, the residuals (the inexplicable fraction of the variation) are quite low, meaning that we can safely ignore external factors beyond EXT, SYN and ERR.

Chapter 5

Exploiting Meta-features for Sentiment Analysis

The popularity of online forums, reviews and social networks has led numerous people to share their opinions on a wide range of subjects, including products, events, news, and even daily experiences. Dealing with this massive amount of data, generated everyday on online platforms, can bring a number of new opportunities to businesses and markets. In particular, the sentiment analysis of such unstructured data can reveal how people feel about a particular product or service.

We dedicate this chapter to address the problem of how to automatically learning to classify the sentiment of short messages/reviews by exploiting information derived from meta-features, our seventh research question. Particularly, we evaluate the use of previously proposed meta-features along with new proposals that exploit sentiment and distances distributions. All this considering a (potential) noisy neighborhood with scarcity of information (short messages).

In more details, we make use of BM25 (Manning et al., 2008) as similarity score to find neighbors, since it is an useful measure to rank documents with short messages as queries [1]. We also exploit the neighborhood of a test example in both the training set and in a dataset containing 1.6 million tweets automatically labeled by its users with emoticons (Go et al., 2009). This methodology allows us to exploit discriminative information from different domains, even noisy ones, like the large twitter dataset with emoticons. The last additional evidence we exploit is taken from the weighted sentiment polarity of the nearest neighbors by using lexicon-based methods to infer the message's polarity towards a sentiment (Baccianella et al., 2010; Hutto and Gilbert, 2014; Thelwall et al., 2010).

[1] On topic classification, the use of BM25 might not be adequate because of the use of long documents as queries.

In sum, we argue that the exploitation of sentiment analysis idiosyncrasies to design meta-features can improve the classification effectiveness in short text messages. The following sections provide a detailed description of our meta-features, followed by a detailed evaluation of our proposals and previously described meta-features.

5.1 Proposed Meta-Level Features for Sentiment Analysis

Similarly to the meta-level features described in Chapter 3.1, the proposed meta-features for sentiment analysis are based on nearest neighbor search. However, we focus on dealing with specific aspects of sentiment analysis of short messages. We use BM25 (Manning et al., 2008) for dealing with short texts as queries and exploit additional information from lexical-based methods as well as the inexpensive (though noisy) tweet messages labeled by Tweeter users with emoticons.

Given the examples in the original input space $\mathscr{X}$, the proposed vector of meta-level features $m_f \in \mathscr{M}$ is expressed as the concatenation of the following sub-vectors, which are defined for each example $x_f \in \mathscr{X}$ and category $c_j \in \mathscr{C}$ for $j = 1, 2, \ldots, |\mathscr{C}|$.

- $\vec{v}_{\vec{x}_f}^{rawsim} = [sim(\vec{x}_{ij}, \vec{x}_f)]$ A k-dimensional vector produced by considering the k nearest neighbors of class c_j to the target vector x_f. More specifically, $\vec{x}_{ij}$ is the i^{th} ($i \leq k$) nearest neighbor to $\vec{x}_f$, and $sim(\vec{x}_{ij}, \vec{x}_f)$ is the a similarity score (BM25 or cosine) between them. Thus, k meta-level features are generated to represent x_f.

- $\vec{v}_{\vec{x}_f}^{cagglex} = \left[\sum sim(\vec{x}_{ij}, \vec{x}_f) * p_{ij}\right]$ A 1-dimensional vector produced by considering the weighted polarity sum of the k nearest neighbors of x_f that belong to the class c_j. The similarity $sim(\vec{x}_{ij}, \vec{x}_f)$ between $\vec{x}_f$ and its i^{th} ($i \leq k$) nearest neighbor $\vec{x}_{ij}$ is used to weight the polarity p_{ij} of the document x_{ij}. The polarity score p_{ij} (that represents the valence of a positive sentiment, for example) is given by a lexical-based method.

- $\vec{v}_{\vec{x}_f}^{maxminlex} = [max(sim(\vec{x}_{ij}, \vec{x}_f) * p_{ij}), min(sim(\vec{x}_{ij}, \vec{x}_f) * p_{ij})]$ A 2-dimensional vector produced by considering the maximum and minimum document weighted polarity of the k nearest neighbors of the target vector x_f that belong to the class c_j. The similarity $sim(\vec{x}_{ij}, \vec{x}_f)$ between $\vec{x}_f$ and its i^{th} ($i \leq k$) nearest neighbor $\vec{x}_{ij}$ is used to weight the polarity p_{ij} of the document x_{ij}.

- $\vec{v}_{\vec{x}_f}^{agglex} = \left[\sum sim(\vec{t}_i, \vec{x}_f) * p_i)\right]$ A 1-dimensional vector with the weighted polarity sum of the k-nearest neighbor's polarities. The similarity $sim(\vec{t}_i, \vec{x}_f)$ between $\vec{x}_f$ and its i^{th} ($i \leq k$) nearest neighbor $\vec{t}_i$ is used to weight the polarity p_i of the document t_i

- $\vec{v}_{\vec{x}_f}^{rawlex} = \left[p_f\right]$ A 1-dimensional vector produced by one of the (possibly many) outputs of a lexical-based method. The output p_f corresponds to the polarity score of the document represented by x_f.

With exception of $\vec{v}_{\vec{x}_f}^{rawlex}$, all remaining described vectors depend on a similarity measure in order to find the nearest neighbors. We generate our meta-features with two similarity scores: Cosine (i.e., $sim(\vec{x},\vec{q}) = Cosine(\vec{x},\vec{q})$) and BM25 (i.e., $sim(\vec{x},\vec{q}) = BM25(\vec{x},\vec{q})$). We use the cosine similarity because of it produced effective results when applied together with bag-of-words weighted with TF-IDF in previous meta-features studies (Canuto et al., 2018). We also compute the vectors using the BM25 similarity score due to its effectiveness on dealing with short queries in information retrieval, which is the case when dealing with short messages as queries for kNN. These two similarity scores are fundamentally different, since BM25 computes the similarity based on terms in a query document appearing in other documents, regardless the additional terms that are not in the query document. On the contrary, cosine similarity considers all terms from both compared documents, since it is symmetrical.

The vectors were grouped into 4 categories, considering whether they use only the polarity information (RAWLEX), only the neighborhood distance information (RAWSIM), the combination of distances with polarities (KNNLEX) and information generated from the external data containing tweet messages labeled with emoticons, instead of the original training data (TWEMOT). Table 5.1 summarizes the names we give to different groups of meta-features depending on the variations of these factors.

Group	Description
RAWSIM	Vector $\vec{v}_{\vec{x}_f}^{rawsim}$
RAWLEX	Vector $\vec{v}_{\vec{x}_f}^{rawlex}$
KNNLEX	Vectors $\vec{v}_{\vec{x}_f}^{cagglex}$, $\vec{v}_{\vec{x}_f}^{maxminlex}$ and $\vec{v}_{\vec{x}_f}^{agglex}$
TWEMOT	Vectors $\vec{v}_{\vec{x}_f}^{cagglex}$, $\vec{v}_{\vec{x}_f}^{maxminlex}$, $\vec{v}_{\vec{x}_f}^{agglex}$ and $\vec{v}_{\vec{x}_f}^{rawsim}$ computed using only the neighborhood from the tweet dataset with emoticons.

Table 5.1: Groups of proposed meta-features.

As described in Table 5.1, the proposed meta-features are able to capture discriminative information from the labeled set using different sources of information. We now further analyze the proposed groups of meta-features, describing which characteristics of the data each of them aim to exploit.

For the RAWSIM features, each test example $\vec{x}_f$ is directly compared to a set of nearest labeled examples. The intuition behind these meta-features consists in the assumption that if the distances between an example to the nearest neighbors belonging to a category c are small, then the example is likely to belong to c.

The RAWLEX features are the raw scores produced by outputs of the used lexicon-based methods. All used methods produce one score for positive polarity and another for negative polarity. Each one of these raw scores correspond to a one-dimensional vector $\vec{v}^{rawlex}_{\vec{x}_f}$.

The KNNLEX features also use the neighborhood distances, but they are only employed for finding and weighting the polarity of the nearest neighbors. More specifically, the meta-features $\vec{v}^{cagglex}_{\vec{x}_f}$ supply the agglomerated polarity of the category c's neighbors. This meta-feature obtains, for each neighbor, the polarity valence p towards a sentiment (e.g., negativity score) using a lexicon-based method (e.g., SentiStrength). The weighted sum of these polarities can be seen as the summarization of the distribution of the neighbor's polarities. If the neighbors of a test document $\vec{x}_f$ have very high values for the positive polarity p and they are also very close to $\vec{x}_f$, the sentiment of $\vec{x}_f$ is likely to be positive. This summarization can be useful, since the lexical-based approaches usually have a small coverage of words which often are not in a particular message. The exploitation of the neighborhood of this message aims at easing this coverage problem by looking at the words of the neighbors.

Another way to analyze the neighborhood polarity distribution is by looking at the maximum and minimum values. The meta-features $\vec{v}^{maxminlex}_{\vec{x}_f}$ extract this additional information to complement the agglomerated polarity previously described. The final meta-feature belonging to the KNNLEX group is the agglomeration $\vec{v}^{agglex}_{\vec{x}_f}$. It differs from the other meta-features because it does not use the labeled information, but just summarizes the polarity of the k nearest neighbors without looking at their categories. By ignoring the category of the neighbors, it is possible to exploit the existence of a mode in the polarity distribution of the neighborhood.

Most meta-features depend on a training set in which we find the k-nearest neighbors. We generate them using the traditional training set (that follows the same distribution of the test documents). Besides this traditional training set, we exploit an additional very large dataset with 1.6 million tweet messages with emoticons inserted by the users (which can be considered as noisy labels) as a training set. The group of meta-features TWEMOT comprises all the proposed kNN-based meta-features generated using this twitter training set. There are two factors that make possible the exploration of this highly noisy tweet dataset. First, we exclude all messages from the tweet dataset that are not in the neighborhood of a query document $\vec{x}_f$. By doing so, we ignore a huge volume of messages that are not close to the domain of interest in which $\vec{x}_f$ is inserted. The second factor is related to the fact that the neighborhood's information is summarized with similarity scores. This summarization with fewer features can be easily evaluated regarding its discriminative power.

Considering k_t neighbors from the tweet dataset, k_o neighbors from the original training data, the $|\mathscr{C}|$ sentiment categories, 2 similarity measures (cosine and BM25) and p polar-

ity scores, the number of features generated will be $2|\mathscr{C}|(k_t+k_o)+p+4(1+3|\mathscr{C}|)$. Specifically, k_t meta-features per category are generated by the vector $\vec{v}_{\vec{x}_f}^{rawsim}$ derived from the k_t nearest neighbors from the tweet dataset. Since we use two similarity measures, we will generate one vector for each similarity measure which gives us $2k_t$ meta-features per category, or $|\mathscr{C}|2k_t$. Similarly, we generate $2k_o$ meta-features driven by the original labeled data, which gives us $2|\mathscr{C}|(k_t+k_o)$ meta-features generated by $\vec{v}_{\vec{x}_f}^{rawsim}$. The equation also accounts for p polarity scores given by each single lexical-based method output, as described in $\vec{v}_{\vec{x}_f}^{rawlex}$.

The vectors $\vec{v}_{\vec{x}_f}^{cagglex}$ and $\vec{v}_{\vec{x}_f}^{maxminlex}$ provide three meta-features per category and $\vec{v}_{\vec{x}_f}^{agglex}$ provides only one meta-feature, generating the total of 1+3$|\mathscr{C}|$ meta-features. Since they are generated using two similarity measures and two different datasets (original training dataset and tweets data-set), the final number of features driven from these vectors are 4(1+3$|\mathscr{C}|$).

In any case, the size of this meta-level feature set is much smaller than that typically found in bag-of-words representation, while explicitly capturing class discriminative information from the labeled set, polarity scores, and external data.

5.2 Experimental Evaluation

In this section we describe the experiments using the proposed meta-features. We start by presenting the experimental setup (Section 5.2.1) followed by the experimental results using distinct sets of meta-features (Section 5.2.2).

5.2.1 Experimental Setup

The experiments follow the same methodology described in Chapter 3, which includes the standard preprocessing task for all datasets (removal of stopwords and use of the TF-IDF weighting scheme) and the use of 5-fold cross-validation to evaluate parametrization (for the neighborhood size k and the regularization parameter of the SVM). The meta-level features were compared using the Micro-F_1 score, which is a standard measure for binary-class categorization tasks (Sokolova and Lapalme, 2009), with the best results and values that are not statistically inferior to the best (with 95% confidence) marked in **bold**.

The meta-features RAWLEX, KNNLEX and TWTEMOT rely on lexicon-based assessments to estimate the sentiment value of each message. We use three recent and freely available lexicon-based classifiers Vader (Hutto and Gilbert, 2014), SentiStrength (Thelwall et al., 2010) and SentiWordnet (Baccianella et al., 2010) to estimate the sentiments of message. Specifically, we extract the positive and negative sentiment scores of each message using these methods, along with combined and neutral scores given by Vader.

In order to evaluate the meta-feature strategies, we use publicly available benchmark (Ribeiro et al., 2016) with nineteen real-world textual datasets gathered from different works. They are named aisopos_tw (Fotis Aisopos, 2014), debate (Diakopoulos and Shamma, 2010), narr_tw (Narr et al., 2012), pappas_ted (Pappas and Popescu-Belis, 2013), pang_movie (Pang and Lee, 2004), sanders_tw[2], ss_bbc (Thelwall, 2013), ss_digg (Thelwall, 2013), ss_myspace (Thelwall, 2013), ss_rw (Thelwall, 2013), ss_twitter (Thelwall, 2013), ss- youtube (Thelwall, 2013), stanford_tw (Go et al., 2009), semeval_tw[3], vader_amzn (Hutto and Gilbert, 2014), vader_movie (Hutto and Gilbert, 2014), vader_nyt (Hutto and Gilbert, 2014), vader_tw (Hutto and Gilbert, 2014) and yelp_re-view[4]. In addition to the previously described datasets, we also exploit the information of 1.6 millions of tweets automatically labeled with emoticons (Go et al., 2009). All these datasets contain short texts as documents, which are distinct from documents usually found in text classification (e.g. news and web pages).

We consider only messages with a clear polarity (positive or negative), since we focus on the supervised (binary) task of discriminating between positive and negative polarities of the messages. The reasons for this are threefold: (i) in several domains (e.g., reviews and micro-reviews), the basic motivation for people to write such messages is to provide positive or negative feedback on products, experiences and services that can be helpful to others; (ii) even in other domains in which "neutral" opinions can occur more frequently, many applications are interested in knowing only the most "polarized" opinions about certain topics (e..g., politicians, events, etc); and finally (iii), even if identifying neutral positions is important, some works (e.g., Barbosa and Feng (2010); Pang and Lee (2004); Wang et al. (2015) have advocated doing this in a prior step (aka, subjectivity extraction) before determining the polarity of the message, which is our focus here.

Table 5.2 shows some of the characteristics of the datasets used to evaluate our meta-features. The first column indicates the name of the dataset used in our experiments, the second column is the number of messages in the dataset, the third shows the number of features (words) represented in the dataset, the fourth corresponds to the average number of words (density) of a message, and the last two columns show the number of positive and negative messages, respectively.

[2] http://www.sananalytics.com/lab/twitter-sentiment
[3] https://www.cs.york.ac.uk/semeval-2013/task2
[4] http://www.yelp.com/dataset_challenge

dataset	#msgs	#feat	density	#pos	#neg
aisopos_tw	278	1493	13.0	159	119
debate	1979	3360	11.5	730	1249
narr_tw	1227	3508	11.3	739	488
pappas_ted	727	1635	11.7	318	409
pang_movie	10662	12432	13.9	5331	5331
sanders_tw	1091	3102	13.5	519	572
ss_bbc	752	5655	40.3	99	653
ss_digg	782	4015	22.2	210	572
ss_myspace	834	2639	14.7	702	132
ss_rw	705	4595	43.3	484	221
ss_twitter	2289	7777	13.8	1340	949
ss_youtube	2432	6275	12.2	1665	767
stanford_tw	359	1620	12.0	182	177
semeval_tw	3060	9087	16.4	2223	837
vader_amzn	3610	3678	11.9	2128	1482
vader_movie	10568	11980	14.0	5242	5326
vader_nyt	4946	8756	13.0	2204	2742
vader_tw	4196	7346	11.2	2897	1299
yelp_review	5000	19398	71.5	2500	2500

Table 5.2: Dataset caracteristics.

5.2.2 Experimental Results

In this section we present results of a series of experiments to evaluate the effectiveness of proposed meta-features for sentiment analysis. Initially, we present results comparing the effectiveness of our solution with the traditional Bag of Words and previously proposed meta-features (not designed for sentiment analysis). Then, we compare the results of the best lexicon-based methods and a supervised ensemble of lexicon-based methods with our meta-features. Finally, we present a study regarding the role of each individual group of meta-features.

5.2.2.1 Meta-features in the Sentiment Analysis Context (Q6)

We start by evaluating different meta-features in sentiment analysis datasets. Our main goal is to answer our sixth research question (Q6) by providing evidence that meta-features can improve the effectiveness of sentiment analysis, specially (Section 5.1). We compare our proposal with the the set of original (Bag of Words) features, the combination (SYN+EXT+ERR) of meta-features based on enriching distance relationships with labeled information (Chapter 4), and three sets of meta-level features proposed by Canuto et al. (2014), Gopal and Yang (2010) and Pang et al. (2015). We also include the combination of these three literature meta-features using the SPEA2SVM (Canuto et al., 2018) genetic algorithm described in Chapter 3 to select the best meta-feature combination for each dataset.

As shown in Table 5.3, the set of proposed meta-features for sentiment analysis achieved the highest effectiveness among the evaluated meta-features in most datasets (sixteen of the nineteen), being (statistically) superior than all baselines in nine datasets. This is

a strong evidence towards the robustness of the proposed meta-features for the specific task of sentiment analysis of short messages. Notice particularly, the significant improvements obtained by the proposed meta-features on vader_tw, ss_twitter, semeva_tw, vader_nyt, ss_-youtube, vader_amz, ss_digg, pang_movie and narr_tw with gains ranging from 5% to 10% over the best baseline.

The reasons for the low competitiveness of the baseline meta-features are twofold: (i) most sentiment analysis datasets have only a few documents with very short messages and (ii) they do not exploit any specific characteristic of sentiment analysis. The proposed meta-features address these issues by: (i) using an external tweet dataset with 1.6 millions of documents; (ii) utilizing a specific similarity score for short queries (BM25), and (iii) by exploiting lexicon-based methods which deal with sentiment analysis specificities.

The SYN+EXT+ERR (Chapter 4) meta-feature group constitutes the most competitive baseline. It achieves the single best result in one dataset (yelp_review) and ties in first place in ten other datasets. In fact, the messages of yelp_review contain longer texts (highest density in Table 5.2), which enables the exploitation of more common words among documents, and consequently, more informative synthetic document representations. The relatively high results on such datasets is evidence that the meta-features described in Chapter 4 can perform relatively well in sentiment analysis, even without domain-specific adaptations.

The other strategies do not use additional information for sentiment analysis or label-enriched representations. In fact, both SPEA2SVM and Bag of Words presented close results in most datasets, being never able to produce statistically superior results than both, Proposed and SYN+EXT+ERR. This is mostly probably due to the lack of specific strategies to extract and expose the sentiment of short texts. In any case, SPEA2SVM was never inferior to Canuto et al. (2014), Gopal and Yang (2010) and Pang et al. (2015), since it is a combination, with meta-feature selection, of these groups. Such combination provided statistically superior gains on top of Bag of Words on vader_tw, vader_movie, stanford_tw, ss_digg and narr_tw.

5.2.2.2 Proposed versus Lexical Approaches

We now compare the relative effectiveness of our proposed meta-features with a simplified ensemble using lexical-based methods (Baccianella et al., 2010; Hutto and Gilbert, 2014; Thelwall et al., 2010) as first level classifiers, in which their outputs are combined with an SVM classifier. We also use the individual classification of the best unsupervised lexical method as baseline. As show in Table 5.4, our approach outperformed by large margins both, the lexical ensemble and the best unsupervised method, in various situations, with gains over the best baseline ranging from 9% to 16% in six datasets, namely vader_movie,

Datasets	*Proposed*	*SYN+EXT+ERR (Chapter 4)*	*SPEA2SVM (Chapter 3)*	*Gopal et al. (2010)*	*Canuto et al. (2014)*	*Pang et al. (2015)*	*Bag of Words*
aisopos_tw	**89.2(5.4)**	**88.5(3.5)**	83.5(3.3)	71.2(6.3)	82.4(4.6)	83.5(3.3)	**89.6(2.8)**
debate	**80.0(2.9)**	**78.5(1.3)**	**78.5(1.9)**	**78.2(1.7)**	76.6(1.6)	76.2(0.9)	77.4(1.3)
narr_tw	**88.8(2.0)**	85.0(1.3)	83.3(1.3)	82.2(1.0)	81.0(2.9)	83.3(1.3)	81.4(2.7)
pappas_ted	72.8(3.0)	**78.3(1.4)**	**77.5(3.2)**	67.0(10.0)	**76.6(3.8)**	**77.5(3.2)**	**76.1(3.8)**
pang_movie	**78.6(1.0)**	77.5(0.3)	77.3(0.7)	77.3(0.9)	77.5(1.4)	74.3(1.3)	76.9(1.3)
sanders_tw	**86.5(2.4)**	**84.2(1.8)**	83.2(2.8)	83.2(3.2)	83.4(1.7)	80.8(2.6)	**84.5(3.1)**
ss_bbc	**88.6(3.8)**	**86.9(4.3)**	**87.0(4.3)**	**86.1(5.3)**	**86.3(6.3)**	**86.9(4.7)**	**87.4(5.6)**
ss_digg	**82.1(2.6)**	78.6(2.3)	78.6(1.1)	76.3(4.1)	78.5(3.8)	77.1(1.5)	77.6(3.7)
ss_myspace	**88.4(1.2)**	**86.1(2.2)**	**86.3(2.4)**	**86.2(4.0)**	85.7(3.2)	85.3(1.8)	**86.2(2.8)**
ss_rw	**79.8(5.0)**	**76.6(2.8)**	**77.8(3.5)**	75.2(2.3)	70.9(1.0)	**77.8(3.5)**	**76.0(4.1)**
ss_twitter	**82.6(1.1)**	75.4(1.5)	73.3(2.2)	73.4(2.2)	73.1(1.4)	73.3(2.2)	72.8(2.8)
ss_youtube	**86.1(1.6)**	82.2(0.8)	82.6(1.8)	79.8(1.7)	80.2(1.9)	80.8(1.5)	81.2(2.5)
stanford_tw	**86.9(3.5)**	**84.7(3.9)**	**86.6(2.7)**	80.5(4.5)	**84.4(3.4)**	84.3(0.8)	**84.7(3.0)**
semeval_tw	**85.8(1.9)**	81.9(1.5)	79.4(1.0)	79.1(1.2)	71.6(1.1)	67.7(12.3)	80.2(1.6)
vader_amzn	**78.0(1.0)**	74.2(1.2)	74.5(1.2)	74.2(2.2)	74.0(1.6)	73.5(1.5)	74.1(1.4)
vader_movie	**79.9(0.6)**	**79.0(0.5)**	**78.9(0.6)**	78.8(1.0)	78.8(1.0)	75.4(0.4)	78.2(0.6)
vader_nyt	**71.2(2.5)**	66.1(1.6)	67.1(0.9)	66.1(1.9)	66.4(2.8)	66.3(0.7)	67.1(1.2)
vader_tw	**97.2(0.6)**	88.2(0.6)	86.2(0.3)	85.6(0.8)	84.2(1.1)	86.0(0.4)	84.0(1.1)
yelp_review	93.4(1.1)	**94.2(0.1)**	90.4(1.2)	87.3(1.5)	90.0(1.3)	87.2(1.7)	93.3(0.4)

Table 5.3: Average Micro-F_1 with different groups of meta-features and the Bag of Words representation.

pang_movie, debate, sanders_tw, stanford_tw and vader_amzn. The most significant gains were on our two biggest datasets vader_movie and pang_movie, with 16% and 15% respectively, providing evidence that the proposed meta-features can extract more discriminative information from large training datasets.

The results of the best lexical unsupervised method (last column of Table 5.4) are close to the supervised methods in only two datasets: pappas_ted and ss_rw, some of the smallest ones. However, the best unsupervised approach could achieve results superior to 75% of effectiveness in twelve of the nineteen datasets, a significant fact, as it does not use any domain-specific labeled information to classify messages. This demonstrates why and how this type of source can produce useful and complementary information to be exploited in other approaches. Finally, exploiting the lexical approaches in a supervised manner (the Lexical ensemble) helped to improve results substantially in most datasets, achieving gains of about 7% on pang_movie, vader_movie, semeval_tw and yelp_review. This further justifies the use of lexical information in our meta-features.

dataset	Proposed	Lex Ensem	Best Lex
aisopos_tw	**89.2 ± 5.4**	85.2 ± 5.9	83.4 ± 6.8
debate	**80.0 ± 2.9**	70.1 ± 1.2	67.4 ± 3.3
narr_tw	**88.8 ± 2.0**	85.8 ± 2.4	81.2 ± 1.9
pappas_ted	**72.8 ± 3.0**	**72.6 ± 7.0**	**72.8 ± 3.2**
pang_movie	**78.6 ± 1.0**	68.4 ± 1.3	63.7 ± 1.0
sanders_tw	**86.5 ± 2.4**	75.5 ± 2.2	73.1 ± 3.6
ss_bbc	**88.6 ± 3.8**	**88.6 ± 4.6**	84.7 ± 1.8
ss_digg	**82.1 ± 2.6**	80.5 ± 5.4	77.2 ± 3.4
ss_myspace	**88.4 ± 1.2**	87.3 ± 1.5	82.6 ± 1.8
ss_rw	**79.8 ± 5.0**	**78.2 ± 5.6**	**78.0 ± 3.8**
ss_twitter	**82.6 ± 1.1**	79.5 ± 2.6	75.8 ± 2.9
ss_youtube	**86.1 ± 1.6**	83.0 ± 3.4	78.6 ± 1.0
stanford_tw	**86.9 ± 3.5**	79.4 ± 6.9	78.5 ± 3.2
semeval_tw	**85.8 ± 1.9**	82.9 ± 1.5	77.9 ± 1.1
vader_amzn	**78.0 ± 1.0**	71.4 ± 2.2	68.5 ± 1.6
vader_movie	**79.9 ± 0.6**	68.8 ± 1.5	64.2 ± 1.0
vader_nyt	**71.2 ± 2.5**	67.5 ± 1.6	65.4 ± 2.1
vader_tw	**97.2 ± 0.6**	**97.3 ± 0.6**	95.0 ± 1.6
yelp_review	**93.4 ± 1.1**	88.8 ± 2.0	82.7 ± 0.9

Table 5.4: Average Micro-F_1 with the proposed meta-features, a simplified ensemble of the lexical-based outputs and best result of an individual lexical method.

5.2.2.3 Similarity Measure for the Proposed Meta-level Features

In this experiment, we investigate the role and the impact of the two used similarity functions – cosine and BM25 – in our results. Table 5.5 shows that using BM25 to deal with the similarity of short messages is in fact better than using the cosine similarity, since it is consistently better or not statistically inferior to the cosine results. In all datasets, BM25 never performs worse than cosine, with small but statistically significant gains up to 2% in six datasets. It is important to point out that the these statistically significant gains are present only in datasets with relatively small density.

5.2.2.4 Analysis of the Proposed Groups of Meta Features

In this section we investigate the quality of each proposed group of meta-level features in isolation as well as the complementarity with regard to other groups. Table 5.7 shows the effectiveness of the SVM classifier trained with each group of the proposed meta-features in isolation. As can be seen, all groups achieved relatively good results in isolation. In fact, there are seven datasets for which there are no statistically significant differences among the groups' results. Moreover, there is no isolated group that is consistently among the best in all datasets, and there are only two datasets in which one specific group is better than all remaining ones.

TWEMOT achieves the worst performance among the four groups in eight datasets.

dataset	COSINE	BM25
aisopos_tw	**88.2 ± 6.5**	**90.0 ± 6.9**
debate	79.0 ± 2.8	**80.6 ± 2.6**
narr_tw	87.5 ± 2.7	**89.4 ± 2.1**
pappas_ted	**73.6 ± 5.7**	**74.8 ± 5.3**
pang_movie	**78.2 ± 1.1**	**78.1 ± 0.6**
sanders_tw	**86.4 ± 1.7**	**86.1 ± 3.3**
ss_bbc	**88.3 ± 4.7**	**87.4 ± 5.9**
ss_digg	**82.0 ± 2.5**	**81.0 ± 5.1**
ss_myspace	**88.7 ± 1.2**	**86.3 ± 4.3**
ss_rw	**80.4 ± 5.4**	**79.0 ± 3.5**
ss_twitter	82.1 ± 1.3	**83.0 ± 0.8**
ss_youtube	85.1 ± 1.6	**86.7 ± 1.3**
stanford_tw	**88.3 ± 6.5**	**87.5 ± 3.7**
semeval_tw	**85.8 ± 1.0**	**85.9 ± 2.4**
vader_amzn	77.3 ± 1.3	**78.2 ± 1.1**
vader_movie	**79.4 ± 1.1**	**79.5 ± 0.6**
vader_nyt	**71.6 ± 1.8**	**71.8 ± 1.7**
vader_tw	96.9 ± 0.7	**97.4 ± 0.6**
yelp_review	**92.6 ± 0.9**	**92.9 ± 0.9**

Table 5.5: Average Micro-F_1 of the proposed meta-features generated using only cosine or BM25 as similarity scores.

This is expected, since TWEMOT exploits primarily the evidence from an external dataset of tweets, labeled with emoticons. It is interesting to notice how TWEMOT and RAWLEX have similar results in most datasets. Similarly, KNNLEX and RAWSIM also share some close results. This may be due the fact that both TWEMOT and RAWLEX try to exploit external sources of information to classify messages, but KNNLEX and RAWSIM are focused primarily on exploiting the neighborhood inside the training data.

In order to further analyze the effects of possibly noisy meta-features and their complementary information, we performed an ablation analysis in which we removed one group of meta-features at a time from the full set ALL, before training the SVM classifier. Table 5.8 shows the effects of such procedure.

The first observation is that the removal of the meta-feature group RAWLEX produced significant losses with regard to the set with ALL meta-features in most datasets. This is an evidence supporting the complementary discriminative evidence provided by the output scores of lexical methods. This was somewhat expected, since these methods provide specific additional information about lexical clues on the message.

The second most impacting group is RAWSIM, which provides the similarity information between a message and their neighborhood. The removal of this group causes statistically significant losses in six datasets, demonstrating the complementary nature of pure similarity scores. In three datasets, KNNLEX is capable of providing complementary information by combining similarity scores with lexical evidence in a message's neighborhood.

dataset	RAWLEX	RAWSIM	KNNLEX	TWEMOT
aisopos_tw	**85.2 ± 5.9**	**86.4 ± 2.4**	**83.9 ± 8.3**	**83.8 ± 5.7**
debate	70.1 ± 1.2	**79.0 ± 3.0**	**78.7 ± 2.0**	69.0 ± 1.4
narr_tw	**85.8 ± 2.4**	82.2 ± 3.4	81.2 ± 2.8	**85.3 ± 2.1**
pappas_ted	**72.6 ± 7.0**	**76.5 ± 2.5**	65.5 ± 4.3	**74.8 ± 6.0**
pang_movie	68.4 ± 1.3	**77.5 ± 0.7**	**77.1 ± 0.4**	65.5 ± 1.2
sanders_tw	75.5 ± 2.2	**82.7 ± 3.1**	**83.5 ± 3.5**	74.6 ± 2.8
ss_bbc	**88.6 ± 4.6**	**86.5 ± 6.1**	**86.7 ± 5.8**	**87.7 ± 5.9**
ss_digg	**80.5 ± 5.4**	**78.3 ± 2.4**	74.9 ± 2.7	**77.1 ± 1.8**
ss_myspace	**87.3 ± 1.5**	**85.6 ± 2.1**	**85.4 ± 3.5**	**87.2 ± 3.2**
ss_rw	**78.2 ± 5.6**	72.6 ± 2.8	72.8 ± 2.4	71.4 ± 4.1
ss_twitter	**79.5 ± 2.6**	73.7 ± 2.7	72.6 ± 2.0	**78.2 ± 2.6**
ss_youtube	**83.0 ± 3.4**	**78.8 ± 1.3**	**78.1 ± 2.0**	**81.2 ± 1.3**
stanford_tw	**79.4 ± 6.9**	**81.1 ± 3.9**	**83.3 ± 6.4**	**80.5 ± 8.0**
semeval_tw	**82.9 ± 1.5**	78.0 ± 1.3	77.3 ± 2.0	**80.8 ± 2.5**
vader_amzn	**71.4 ± 2.2**	**73.4 ± 1.3**	**75.0 ± 2.4**	66.3 ± 1.1
vader_movie	68.8 ± 1.5	**78.3 ± 0.6**	**78.6 ± 0.9**	66.7 ± 1.3
vader_nyt	**67.5 ± 1.6**	**66.9 ± 2.4**	**66.4 ± 2.8**	**66.0 ± 1.4**
vader_tw	**97.3 ± 0.6**	84.7 ± 1.5	84.1 ± 1.2	89.1 ± 0.7
yelp_review	88.8 ± 2.0	**91.1 ± 0.6**	87.9 ± 0.9	79.8 ± 1.3

Table 5.6: Average Micro-F_1 of each proposed group of meta-features in isolation.

Although it does not produce high improvements, since RAWLEX, RAWSIM and TWEMOT already provide some of this information, the statistical significant losses demonstrate that there is additional information that can be extracted from this group in some datasets.

The removal of the TWEMOT group also produces significant losses in four datasets. This is evidence towards the importance of the additional information exploited by the large (though noisy) tweet dataset. As we can see, the removal of this group does not improve the effectiveness in any dataset. This is surprising – TWEMOT is capable of capturing useful classification evidence from a highly noisy dataset for some datasets, without harming the remaining ones.

As we can see, the removal of any single group does have impacts on the effectiveness in several datasets. This is evidence that the proposed meta-features are not inserting significant complexity in the meta-feature space and can be included in the pool of available evidence for the (hard) task of analyzing the sentiment of short messages.

5.2.2.5 Importance of Groups with using $2^k r$ Factorial Design

The best way to evaluate the interactions among our four groups of meta-features is by doing an analysis of all 2^k possible combinations of groups for each dataset[5]. By replicating the experiments with each possible combination (using 5-fold cross validation), we can also

[5]We consider the case without any group using a "random" classifier, which returns positive with a 50% chance.

dataset	RAWLEX	RAWSIM	KNNLEX	TWEMOT
aisopos_tw	**85.2 ± 5.9**	**86.4 ± 2.4**	**83.9 ± 8.3**	**83.8 ± 5.7**
debate	70.1 ± 1.2	**79.0 ± 3.0**	**78.7 ± 2.0**	69.0 ± 1.4
narr_tw	**85.8 ± 2.4**	**82.2 ± 3.4**	**81.2 ± 2.8**	**85.3 ± 2.1**
pappas_ted	**72.6 ± 7.0**	**76.5 ± 2.5**	**65.5 ± 4.3**	**74.8 ± 6.0**
pang_movie	68.4 ± 1.3	**77.5 ± 0.7**	**77.1 ± 0.4**	65.5 ± 1.2
sanders_tw	75.5 ± 2.2	**82.7 ± 3.1**	**83.5 ± 3.5**	74.6 ± 2.8
ss_bbc	**88.6 ± 4.6**	**86.5 ± 6.1**	**86.7 ± 5.8**	**87.7 ± 5.9**
ss_digg	**80.5 ± 5.4**	**78.3 ± 2.4**	**74.9 ± 2.7**	**77.1 ± 1.8**
ss_myspace	**87.3 ± 1.5**	**85.6 ± 2.1**	**85.4 ± 3.5**	**87.2 ± 3.2**
ss_rw	**78.2 ± 5.6**	**72.6 ± 2.8**	**72.8 ± 2.4**	**71.4 ± 4.1**
ss_twitter	**79.5 ± 2.6**	73.7 ± 2.7	72.6 ± 2.0	**78.2 ± 2.6**
ss_youtube	**83.0 ± 3.4**	**78.8 ± 1.3**	**78.1 ± 2.0**	**81.2 ± 1.3**
stanford_tw	**79.4 ± 6.9**	**81.1 ± 3.9**	**83.3 ± 6.4**	**80.5 ± 8.0**
semeval_tw	**82.9 ± 1.5**	78.0 ± 1.3	77.3 ± 2.0	**80.8 ± 2.5**
vader_amzn	**71.4 ± 2.2**	**73.4 ± 1.3**	**75.0 ± 2.4**	66.3 ± 1.1
vader_movie	68.8 ± 1.5	**78.3 ± 0.6**	**78.6 ± 0.9**	66.7 ± 1.3
vader_nyt	**67.5 ± 1.6**	**66.9 ± 2.4**	**66.4 ± 2.8**	**66.0 ± 1.4**
vader_tw	**97.3 ± 0.6**	84.7 ± 1.5	84.1 ± 1.2	89.1 ± 0.7
yelp_review	88.8 ± 2.0	**91.1 ± 0.6**	87.9 ± 0.9	79.8 ± 1.3

Table 5.7: Average Micro-F_1 of each proposed group of meta-features in isolation.

dataset	ALL	no RAWLEX	no RAWSIM	no KNNLEX	no TWEMOT
aisopos_tw	89.2 ± 5.4	86.4 ± 6.3	86.0 ± 6.4 ↓	88.9 ± 6.7	90.7 ± 3.6
debate	80.0 ± 2.9	78.8 ± 2.4 ↓	79.9 ± 2.9	79.5 ± 2.7	80.2 ± 2.9
narr_tw	88.8 ± 2.0	86.1 ± 1.3 ↓	88.8 ± 2.4	89.1 ± 2.2	88.0 ± 1.5
pappas_ted	72.8 ± 3.0	72.5 ± 1.6	76.6 ± 5.5	69.6 ± 6.8	72.9 ± 3.3
pang_movie	78.6 ± 1.0	78.0 ± 1.0 ↓	78.2 ± 0.8	78.3 ± 0.9	78.5 ± 1.0
sanders_tw	86.5 ± 2.4	84.7 ± 2.4 ↓	85.8 ± 2.3 ↓	85.6 ± 2.6	86.2 ± 3.1
ss_bbc	88.6 ± 3.8	88.3 ± 5.3	87.9 ± 4.1	88.6 ± 4.7	86.7 ± 4.7 ↓
ss_digg	82.1 ± 2.6	80.1 ± 1.3 ↓	81.5 ± 2.8	81.8 ± 3.0	81.6 ± 3.3
ss_myspace	88.4 ± 1.2	86.2 ± 1.4 ↓	88.7 ± 1.5	89.0 ± 1.5	87.9 ± 2.7
ss_rw	79.8 ± 5.0	74.8 ± 5.1 ↓	79.7 ± 5.6	80.2 ± 4.8	80.4 ± 4.4
ss_twitter	82.6 ± 1.1	77.8 ± 3.1 ↓	83.0 ± 1.5	82.7 ± 1.0	80.9 ± 2.2 ↓
ss_youtube	86.1 ± 1.6	83.0 ± 0.9 ↓	85.8 ± 2.0 ↓	86.4 ± 1.8	84.7 ± 1.9 ↓
stanford_tw	86.9 ± 3.5	83.9 ± 5.6	86.3 ± 4.0	86.1 ± 3.0 ↓	84.7 ± 5.1 ↓
semeval_tw	85.8 ± 1.9	82.0 ± 1.2 ↓	85.8 ± 1.7	85.6 ± 1.7	85.3 ± 0.5
vader_amzn	78.0 ± 1.0	75.3 ± 2.0 ↓	76.9 ± 1.4 ↓	77.3 ± 1.3	77.5 ± 2.8
vader_movie	79.9 ± 0.6	79.3 ± 0.9 ↓	79.1 ± 1.1 ↓	79.1 ± 1.0 ↓	79.8 ± 0.6
vader_nyt	71.2 ± 2.5	69.5 ± 1.3 ↓	71.4 ± 2.0	71.2 ± 2.1	70.9 ± 2.6
vader_tw	97.2 ± 0.6	89.6 ± 1.3 ↓	97.2 ± 0.7	97.2 ± 0.7	97.3 ± 0.5
yelp_review	93.4 ± 1.1	91.1 ± 0.7 ↓	93.0 ± 1.1 ↓	92.3 ± 1.0 ↓	93.0 ± 1.1

Table 5.8: Average Micro-F_1 removing one group of meta-features from the full set. ↓ indicates statistically significant losses due to the removal of a meta-feature group from the full set ALL.

evaluate the effects of uncontrollable external factors. We follow the standard quantitative approach called $2^k r$ factorial design (Jain, 1991) to analyze the effects of the individual groups of meta-features, as well as the effectiveness improvements produced by their interactions.

The first step to perform a $2^k r$ factorial design is to define the binary factors that may affect a response variable (e.g., Micro-F_1 score). In our case, each factor corresponds to the presence or absence of one group of meta-features. We name the presence of each group of features as follows:

- A – Presence of KNNLEX
- B – Presence of RAWSIM
- C – Presence of TWEMOT
- D – Presence of RAWLEX

In order to summarize the analysis of the nineteen datasets with the sixteen possible combinations of our four factors, we clustered our datasets into two groups, according to the importance and interaction of meta-features in each dataset. We hope to keep similar datasets in the same group in order to summarize the results of the group. We use a simple k-means algorithm to group close datasets according to their individual factorial design results. The found groups are:

- **Group 1 datasets:** debate, pang_movie, sanders_tw, vader_amzn, vader_movie and yelp_review.
- **Group 2 datasets:** aisopos_tw, narr_tw, pappas_ted, ss_bbc, ss_digg, ss_myspace, ss_rw, ss_twitter, ss_you-tube, stanford_tw, semeval_tw, vader_nyt and vader_tw.

For each dataset, we compute the percentage of variation in the results that can be explained by each individual group of meta-features and by the interaction between groups of meta-features considering each possible combination. We summarize the effects of each combination[6] on different datasets by showing its average. Table 5.9 shows this summarization for the two groups of datasets. In addition to the mean value, we also show the lowest and the highest value among the datasets in each group.

As we can see, the variation on results in the datasets of group 1 can mostly be explained by the presence of A (KNNLEX) and B (RAWSIM) in isolation. The inclusion of each one of these groups account for about 20% of all the variation in the observed results. The interaction between A:B is the third most important observed factor. In fact, the effects of A, B and A:B account for 60% of all the Micro-F_1 variation. The fourth most important factor is the inclusion of D, which accounts for about 10% of the variation, showing the importance of the RAWLEX meta-features in isolation. The remaining effects, though small, account together for about 30% of the total variation in the results, which highlights the complementarity among the proposed groups of meta-features.

[6]The 95% confidence interval on the effects of all combinations of the tested datasets are always inferior to 1%.

Factor/ Interaction	Group 1 datasets		Group 2 datasets	
	Mean	Range	Mean	Range
A	19.5	[14.4, 24.3]	6.1	[3.8, 10.1]
B	19.8	[10.9, 26.3]	5.7	[1.1, 9.3]
C	3.3	[2.2, 5.5]	9.5	[3.2, 16.2]
D	9.6	[4.6, 17.2]	20.7	[7.3, 39.5]
A:B	17.3	[10.3, 22.4]	5.6	[3.6, 9.1]
A:C	2.6	[1.6, 3.8]	4.8	[2.8, 7.4]
B:C	2.6	[1.8, 4.2]	4.0	[2.2, 6.2]
A:D	3.9	[2.4, 8.1]	4.5	[3.2, 8.0]
B:D	3.8	[2.8, 6.3]	3.6	[1.6, 5.3]
C:D	2.2	[1.4, 3.2]	6.8	[3.8, 9.4]
A:B:C	2.9	[1.9, 4.2]	4.6	[2.6, 10.0]
A:B:D	3.9	[2.6, 6.3]	3.8	[2.4, 5.6]
A:C:D	1.9	[1.4, 3.6]	4.0	[1.4, 6.0]
B:C:D	1.8	[1.2, 2.8]	3.3	[0.4, 6.2]
A:B:C:D	1.9	[1.3, 2.9]	3.0	[0.6, 5.6]
Residuals	2.8	[0.7, 5.7]	9.9	[0.6, 30.4]

Table 5.9: Explained percentage of result variation by individual meta-feature groups and interactions between them.

Regarding the datasets in group 2, the presence of D (RAW-LEX) in isolation accounts for 20% of all the Micro-F_1 variations in the experiments. The presence of C (TWEMOT) in isolation and the interaction C:D are the most important effects to explain the results. This means that the use of additional information from the lexical-based approaches and from the large tweet corpus are the most important factors for these datasets. It is important to notice that A, B and most of the interactions have a considerable participation to explain the results. The residuals (inexplicable fraction of the variation in the results) are quite high (about 10%) on the second group. This may be due the fact that most of the datasets on the second group are small, which leads to higher variability when classifying their samples. Since they are more likely to suffer from overfitting due to shortage of training information, their dependence on external data (provided by RAWLEX and TWEMOT) is expected.

Chapter 6

GPU-based kNN Implementation

Since the proposed meta-features rely on the computation of distances between documents, unless this procedure is efficiently implemented, their use may have limited applicability. Particularly, most distance-based meta-features rely on finding the k nearest neighbors of a document using some distance measure, which can easily become prohibitive for large datasets in highly dimensional spaces. In this Chapter, we describe our proposal to make use of low cost commodity GPUs, providing a scalable solution to the kNN problem for text data. We start providing a brief introduction to parallelism in GPU in Section 6.1, followed by the description of our GPU-based implementation, specially designed for highly dimensional, sparse data in Section 6.2.

6.1 Parallelism and the GPU

In the last few years, the focus on processor architectures has moved from increasing clock rate to increasing parallelism. Rather than increasing the speed of its individual processor cores, traditional CPUs are now virtually all multicore processors. In a similar fashion, manycore architectures like GPUs have concentrated on using simpler and slower cores, but in much larger counts, in the order of thousands of cores. The general perception is that processors are not getting faster, but instead are getting wider, with an ever increasing number of cores. This has forced a renewed interest in parallelism as the only way to increase performance.

The high computational power and affordability of GPUs has led to a growing number of researchers making use of GPUs to handle massive amounts of data. While multicore CPUs are optimized for single-threaded performance, GPUs are optimized for throughput and a massive multi-threaded parallelism. As a result, the GPUs deliver much better energy efficiency and achieves higher peak performance for throughput workloads. However, GPUs

have a different architecture and memory organization and to fully exploit its capabilities it is necessary considerable parallelism (tens of thousands of threads) and an adequate use of its hardware resources. This imposes some constraints in terms of designing appropriate algorithms, requiring the design of novel solutions and new implementation approaches.

A GPU consists of a M-SIMD machine, that is, a Multiple SIMD (Single Instruction Multiple Data) processor. Each SIMD unit is known as a streaming multiprocessor (SM) and contains streaming processor (SP) cores. At any given clock cycle, each SP executes the same instruction, but operates on different data. The GPU supports thousands of light-weight concurrent threads and, unlike the CPU threads, the overhead of creation and switching is negligible. The threads on each SM are organized into thread groups that share computation resources such as registers. A thread group is divided into multiple schedule units, called warps, that are dynamically scheduled on the SM. Because of the SIMD nature of the SP's execution units, if threads in a schedule unit have to perform different operations, such as going through branches, these operations will be executed serially as opposed to in parallel. Additionally, if a thread stalls on a memory operation, the entire warp will be stalled until the memory access is done. In this case the SM selects another ready warp and switches to that one. The GPU global memory is typically measured in gigabytes of capacity. It is an off-chip memory and has both a high bandwidth and a high access latency. To hide the high latency of this memory, it is important to have more threads than the number of SPs and to have threads in a warp accessing consecutive memory addresses that can be easily coalesced. The GPU also provides a fast on-chip shared memory which is accessible by all SPs of a SM. The size of this memory is small but it has a low latency and it can be used as a software-controlled cache. Moving data from the CPU to the GPU and vice versa is done through a PCIExpress connection.

The GPU programming model requires that part of the application runs on the CPU while the computationally-intensive part is accelerated by the GPU. The programmer has to modify his application to take the compute-intensive kernels and map them to the GPU. A GPU program exposes parallelism through a data-parallel SPMD (Single Program Multiple Data) kernel function. During implementation, the programmer can configure the number of threads to be used. Threads execute data parallel computations of the kernel and are organized in groups called thread blocks, which in turn are organized into a grid structure. When a kernel is launched, the blocks within a grid are distributed on idle SMs. Threads of a block are divided into warps, the schedule unit used by the SMs, leaving for the GPU to decide in which order and when to execute each warp. Threads that belong to different blocks cannot communicate explicitly and have to rely on the global memory to share their results. Threads within a thread block are executed by the SPs of a single SM and can communicate through the SM shared memory. Furthermore, each thread inside a block has

its own registers and private local memory and uses a global thread block index, and a local thread index within a thread block, to uniquely identify its data.

6.2 Proposed Implementation

The proposed parallel implementation, called GPU-based Textual kNN (GT-kNN), greatly improves the k nearest neighbors search in textual datasets. The solution efficiently implements an inverted index in the GPU, by using a parallel counting operation followed by a parallel prefix-sum calculation, taking advantage of Zipf's law, which states that in a textual corpus, few terms are common, while many of them are rare. This makes the inverted index a good choice for saving space and avoiding unnecessary calculations. At query time, this inverted index is used to quickly find the documents sharing terms with the query document. This is made by constructing a query index which is used for a load balancing strategy to evenly distribute the distance calculations among the GPU's threads. Finally, the k nearest neighbors are determined through the use of a truncated bitonic sort to avoid sorting all computed distances. Next we present a detailed description of these steps.

6.2.1 Creating the Inverted Index

The inverted index is created in the GPU memory, assuming the training dataset fits in memory and is static[1]. Let $\mathscr{V}$ be the vocabulary of the training dataset, that is the set of distinct terms of the training set. The input data is the set $\mathscr{E}$ of distinct term-documents (t,d), pairs occurring in the original training dataset, with $t \in \mathscr{V}$ and $d \in \mathbb{D}_{train}$. Each pair $(t,d) \in \mathscr{E}$ is initially associated with a term frequency tf, which is the number of times the term t occurs in the document d. An array of size $|\mathscr{E}|$ is used to store the inverted index. Once the set $\mathscr{E}$ has been moved to the GPU memory, each pair in it is examined in parallel, so that each time a term is visited the number of documents where it appears (document frequency - df) is incremented and stored in the array df of size $|\mathscr{V}|$. A parallel prefix-sum is executed, using the CUDPP library (Sengupta et al., 2011), on the df array by mapping each element to the sum of all terms before it and storing the results in the *index* array. Thus, each element of the *index* array points to the position of the corresponding first element in the *invertedIndex*, where all (t,d) pairs will be stored ordered by term. Finally, the pairs (t,d) are processed in parallel and the *frequency-inverse document frequency* tf-$idf(t,d)$ for each pair is computed and included together with the documents identification in the *invertedIndex* array, using the pointers provided by the *index* array. Also during this parallel processing, the value of the

[1] If the training data is larger than the GPU memory, we can split the data into parts, execute our strategy on each subset and then perform a merge-sort on the kNN results obtained for each part.

norm for each training document, which is used in the calculus of the cosine or Euclidean distance, is computed and stored in the *norms* array. Algorithm 1 depicts the inverted index creation process.

Algorithm 1: *CreateInvetedIndex*(E)

input : term-document pairs in $E[0..|\mathscr{E}|-1]$.
output: df, *index*, *norms*, *invertedIndex*.

1 array of integers $df[0..|\mathscr{V}|-1]$ `// document-frequency array, initialized with zeros.`
2 array of integers $index[0..|\mathscr{V}|-1]$.
3 array of floats $norms[0..|\mathbb{D}_{train}-1|]$.
4 $invertedIndex[0..|\mathscr{E}|-1]$ `// the inverted index`
5 Count the occurrences of each term in parallel on the input and accumulates in df.
6 Perform an exclusive parallel prefix sum on df and stores the result in *index*.
7 Access in parallel the pairs in E, with each processor performing the following tasks:
8 **begin**
9 Compute the tf-idf value of each pair.
10 Accumulate the square of the tf-idf value of a pair (t,d) in $norms[d]$.
11 Store in *invertedIndex* the entries corresponding to pairs in E, according to *index*.
12 **end**
13 Compute in parallel the square root of the values in array *norms*.
14 **Return** the arrays: *count*, *index*, *norms* and *invertedIndex*.

Figure 6.1 illustrates each step of the inverted index creation for a five documents collection where only five terms are used. If we take t_2 as an example, the index array indicates that its inverted document list (d_2, d_4) starts at position 3 of the *invertedindex* array and finishes at position 4 (5 minus 1).

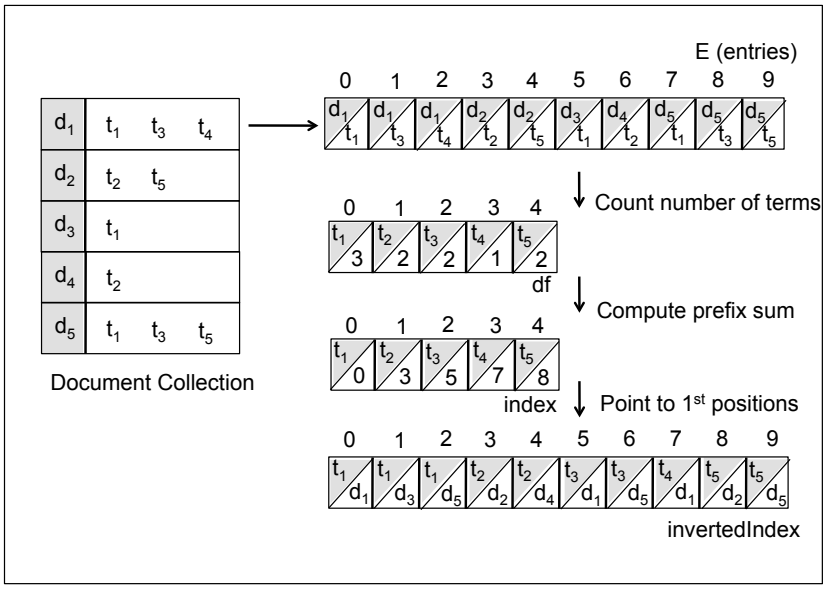

Figure 6.1: Creating the inverted index

6.2.2 Calculating the distances

Once the inverted index has been created, it is now possible to calculate the distances between a given query document q and the documents in $\mathbb{D}_{train}$. The distances computation can take advantage of the inverted index model, because only the distances between query q and those documents in $\mathbb{D}_{train}$ that have terms in common with q have to be computed. These documents correspond to the elements of the *invertedIndex* pointed to by the entries of the *index* array corresponding to the terms occurring in the query q.

The obvious solution to compute the distances is to distribute the terms of query q evenly among the processors and let each processor p access the inverted lists corresponding to terms allocated to it. However, the distribution of terms in documents of text collections is known to follow approximately the Zipf Law. This means that few terms occur in large amount of documents and most of terms occur in only few documents. Consequently, the sizes of the inverted list also vary according to te Zipf Law, thus distributing the work load according to the terms of q could cause a great imbalance of the work among the processors.

In this dissertation besides using an inverted index to boost the computation of the distances, we also propose a load balance method to distribute the documents evenly among the processors so that each processor computes approximately the same number of distances. In order to facilitate the explanation of this method, suppose that we concatenate all the inverted lists corresponding to terms in q in a logical vector $E_q = [0..|E_q|-1]$, where $|E_q|$ is the sum of the sizes of all inverted lists of terms in q. Considering the example in Fig. 6.1 and supposing that q is composed by the terms t_1, t_3 and t_4, the logical vector E_q would be formed by the following pairs of the inverted index: $E_q = [(t_1,d_1), (t_1,d_3), (t_1,d_5), (t_3,d_1), (t_3,d_5), (t_4,d_1)]$ and $|E_q|$ equals to six.

Given a set of processors $\mathscr{P} = \{p_0, \cdots p_{|\mathscr{P}|-1}\}$, the load balance method should allocate elements of E_q in intervals of approximately the same size, that is, each processor $p_i \in \mathscr{P}$ should process elements of E_q in the interval $[i\lceil\frac{|E_q|}{|\mathscr{P}|}\rceil, min((i+1)\lceil\frac{|E_q|}{|\mathscr{P}|}\rceil - 1, |E_q|-1)]$. Consider the example stated above, and suppose that the set of processors is $\mathscr{P} = \{p_0, p_1, p_2\}$. Thus elements of E_q with indices in the interval $[0,1]$ would be assigned to p_0, indices in $[2,3]$ would be processed by p_1 and indices in $[4,5]$ would be processed by p_2.

Since each processor knows the interval of the indices of the logical vector E_q it has to process, all that is necessary to execute the load balancing is a mapping of the logical indices of E_q to the appropriate indices in the inverted index (array *invertedIndex*). In the case of the example associated to Fig. 6.1, the following mappings between logical indices and indices of the invertedIndex array must be performed: $0 \rightarrow 0$, $1 \rightarrow 1$, $2 \rightarrow 2$, $3 \rightarrow 5$, $4 \rightarrow 6$ and $5 \rightarrow 7$. Each processor executes the mapping for the indices in the interval corresponding to it and

finds the corresponding elements in the invertedIndex array for which it has to compute the distances to the query.

Let $\mathscr{V}_q \subset \mathscr{V}$ be the vocabulary of the query document d. The mapping proposed in this work uses three auxiliary arrays: $df_q[0..|\mathscr{V}_q|-1]$, $start_q[0..|\mathscr{V}_q|-1]]$ and $index_q[0..|\mathscr{V}_q|-1]$. The arrays df_q and $start_q$ are obtained together by copying in parallel $df[t_i]$ to $df_q[t_i]$ and $index[t_i]$ to $start_q[t_i]$, respectively, for each term t_i in the query q. Once the df_q is obtained, an inclusive parallel prefix sum on df_q is performed and the results are stored in $index_q$.

Algorithm 2: *DistanceCalculation*$(invertedIndex, q)$

input : *invertedIndex*, df, *index*, *query* $q[0..|\mathscr{V}_q|-1]$.
output: distance array $dist[0..|\mathbb{D}_{train}|-1]$ initialized according to the distance function used.

```
1  array of integers df_q[0..|V_q|-1] initialized with zeros
2  array of integers index_q[0..|V_q|-1]
3  array of integers start_q[0..|V_q|-1]
4  for each term t_i ∈ q, in parallel do
5  |   df_q[i] = df[t_i];
6  |   start_q[i] = index[t_i];
7  end
8  Perform an inclusive parallel prefix sum on df_q and stores the results in index_q
9  foreach processor p_i ∈ P do
10 |   for x ∈ [i⌈|E_q|/|P|⌉, min((i+1)⌈|E_q|/|P|⌉ − 1, |E_q| − 1)] do
   |   |   // Map position x to the correct position indInvPos of the invertedIndex
11 |   |   pos = min(i : index_q[i] > x);
12 |   |   if pos = 0 then
13 |   |   |   p = 0; offset = x;
14 |   |   else
15 |   |   |   p = index_q[pos − 1]; offset = x − p;
16 |   |   end
17 |   |   indInvPos = start_q[pos] + offset
18 |   |   use q[pos] and invertedIndex[indInvPos] in the partial computation of the distance between q and
   |   |     the document associated to invertedIndex[indInvPos]
19 |   end
20 end
```

Algorithm 2 shows the pseudo-code for the parallel computation of the distances between documents in the training set and the query document. In lines 4-7 the arrays df_q and $start_q$ are obtained. In line 8 the array $index_q$ is obtained by applying a parallel prefix sum on array df_q. Next, each processor executes a mapping of each position x in the interval of indices of E_q associated to it to the appropriate position of the invertedIndex. This mapping is described in lines 10-17 of the algorithm. Then, the mapped entries of the inverted index are used to compute the distances between each document associated with these entries and the query.

Figure 6.2 illustrates each step of Algorithm 2 for a query containing three terms, t_1, t_3 and t_4, using the same collection presented in the example of Figure 6.1. Initially, the arrays df_q and $start_q$ are obtained by copying in parallel entries respectively from arrays df and *index*, corresponding to the three query terms. Next a parallel prefix sum is applied to array

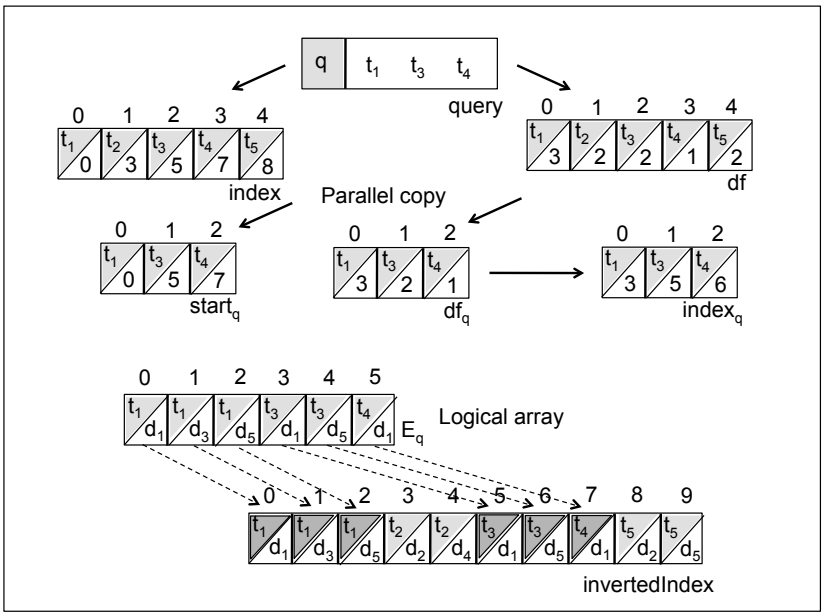

Figure 6.2: Example of the execution of Algorithm 2 for a query with three terms.

df_q and the $index_q$ array is obtained. Finally the Figure shows the mapping of each position of the logical array E_q into the corresponding positions of the invertedIndex array.

6.2.3 Finding the k Nearest Neighbors

With the distances computed, it is necessary to obtain the k closest documents. This can be accomplished by making use of a partial sorting algorithm on the array containing the distances, which is of size $|\mathbb{D}_{train}|$. For this, we implemented a parallel version of the Truncated Bitonic Sort (TBiS), which was shown to be superior to other partial sorting algorithms in this context (Sismanis et al., 2012). One advantage of the parallel TBiS is data independence. At each step, the algorithm distributes elements equally among the GPU's threads avoiding synchronizations as well as memory access conflicts. Although the partial bitonic sort is $O(|\mathbb{D}_{train}| \log^2 k)$, worse than the best known algorithm which is $O(|\mathbb{D}_{train}| \log k)$, for a small k the ratio of $\log k$ becomes almost negligible. In the case of ADC using kNN, the value of k is usually not greater than 50. Our parallel TBiS implementation also uses a reduction strategy, allowing each GPU block to act independently from each other on a partition of array containing the computed distances. Results are then merged in the CPU using a priority queue.

6.3 Experimental Results

In this section, we compare the experimental results of our implementation with literature baselines. Particularly, we compare the computation time to generate meta-features using three different algorithms: (1) **GT*k*NN**, our GPU-based implementation of kNN; (2) **BF-CUDA**, a brute force [2] kNN implementation using CUDA proposed by Garcia et al. (2008); and (3) **ANN**, a C++ library that supports exact and approximate nearest neighbor searching[3]. We use the ANN exact version, since it was used in the previous meta-feature works (Gopal and Yang, 2010; Yang and Gopal, 2012). We chose BF-CUDA because it is the main representative of the GPU-based brute force approach. However, the other implementations (Kuang and Zhao, 2009; Sismanis et al., 2012) (some not available for download) also work with a complete distance matrix and would produce similar results.

All experiments were run on a Quad-Core Intel® Xeon® E5620, running at 2.4GHz, with 32Gb RAM. The GPU experiments were run on a NVIDIA Tesla K40, with 12Gb RAM. In order to consider the costs of all data transfers in our efficiency experiments, we report the wall times on a dedicated machine so as to rule out external factors, like high load caused by other processes.

6.3.1 Computational Time to Find Neighbors (Q7)

Here, we evaluate the potential efficiency benefits of using the GT*k*NN GPU implementation to find the neighborhood of documents, which is pre-requisite to enable the use of distance-based meta-features.

Table 6.1 shows the average time to find the neighborhood for a batch of test examples using ours and the baseline's kNN implementations. Since we use a 5-fold cross validation, we measure the time to find the neighborhood for all the examples in the batch of examples in each test fold.

As can be seen in Table 6.1, the search for neighbors using GT*k*NN shows significant speedups in relation to the other kNN implementations. In particular, the speedups for the small datasets range from 4.8 to 141.3 in relation to the ANN implementation, used in previous works to generate meta-features. This high speedup was somewhat expected, since the ANN do not explore parallelism to compute the distances. However, even when compared to the parallel BF-CUDA implementation (Garcia et al., 2008), GT*k*NN was able to achieve speedups ranging from 3.6 to 15.7. This was possible mainly because BF-CUDA does not

[2] A brute force kNN method computes the distance function between a query and each one of the training documents.

[3] `http://www.cs.umd.edu/~mount/ANN/`

optimize the distance calculations to deal with the low density of terms present in textual documents nor tries to balance the load among threads.

GT*k*NN produced the best speedups in 4UNI, 20NG and ACM, but it obtained a lower speedup in REUT90. This may be due to the fact that REUT90 has a large number of classes and only a few documents in each class. Since the meta-features are generated one class at a time, we could not explore the parallelism in its full potential. For example, if there are only 10 training documents in a class, we can only perform at most 10 simultaneous distance calculations, leaving most CUDA cores idle.

Dataset	*Execution Time (s)*			*Speedup*	
	GT*k*NN	BF-CUDA	ANN	BF-CUDA	ANN
4UNI	$\mathbf{40 \pm 1}$	259 ± 46	1590 ± 29	6.4	39.6
20NG	$\mathbf{187 \pm 4}$	2004 ± 17	10947 ± 1323	10.7	68.7
ACM	$\mathbf{112 \pm 3}$	1760 ± 91	13589 ± 1539	15.7	141.3
REUT90	$\mathbf{625 \pm 12}$	2242 ± 5	3024 ± 303	3.6	4.8
MED	4637 ± 43	*	*	*	*

Table 6.1: Average time in seconds (and 95% confidence interval) to find the neighborhood of documents using different kNN strategies. GT*k*NN is significantly better than others and makes the generation of meta-features possible to MED.

The effectiveness of the proposed load balancing strategy was evaluated by comparing it to the more straightforward unbalanced solution that assigns each term of a query to a thread running on the GPU. The problem with this simple strategy is that the threads will do uneven work since the lists associated with the terms vary considerably in size. In contrast, our strategy associates each thread with the same number of entries (for all terms of the given query) in the inverted index. Since the load balancing is applied to the distance calculation operation, we measure the time to perform this operation with and without the load balancing. Table 6.2 summarizes the times to calculate the distances when working with one query considering our solution (with load balancing and the simple unbalanced solution). As expected, the speedups obtained with load balancing demonstrates the effectiveness of our approach, since the performance gains vary from 105 to 303 times the version without load balancing.

DATASET	With Load Balancing	Without Load Balancing
4UNI	0.03	4.71
20NG	0.05	12.65
ACM	0.01	1.05
REUT90	0.02	6.87
MED	0.08	24.27

Table 6.2: Average time (in miliseconds) to calculate distances for each query with and without the proposed load balancing.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This is the first dissertation that thoroughly investigated the impact of different distance-based meta-feature groups for text classification. In this dissertation, we not only proposed a comprehensive set of new meta-features, but also performed a thorough analysis of different vector spaces drawn from meta-features. Specifically, we provided empirical evidence about the potential benefits of combining groups of meta-features that contain complementary discriminative information. We also investigated potential issues due to the use of more complex and highly dimensional meta-feature spaces. Furthermore, we enriched meta-features with labeling information, adapted meta-features for the sentiment analysis context and parallelized the neighborhood search for text documents, which is crucial for the practical application of meta-features. Before delving into the planned future work, in the following sections we outline each specific contribution in more details.

7.1.1 Meta-Features: Proposition, Evaluation and Selection

Q1 – How effective is the combination of the meta-feature groups proposed in different works?

We showed that distance-based meta-features from different works present complementary information, improving the results of Gopal and Yang (2010), Pang et al. (2015) and Canuto et al. (2014) by up to 9%, 40% and 4%, respectively. This was an important result towards demonstrating their potential benefits. However, we also showed the challenges of combining meta-features, since their concomitant presence may incur on overfitting of some classification methods.

Q2 – How effective and efficient are different strategies for feature selection on the meta-feature space?

In order to mitigate the potentially harmful effects of the combination of some meta-feature groups, we evaluated diverse feature selection strategies on the meta-feature space. The proposed meta-feature selection strategy SPEA2SVM was able to considerably reduce the number of meta-features by up to 89%, while maintaining or even improving the effectiveness by removing noise. This was not possible using any other of our baselines in the evaluated datasets. In fact, SPEA2SVM effectiveness results were equivalent to the results obtained by SingleGA and the gold standard Brute-force, while being significantly faster and more feature-efficient than both. The second proposed strategy SPEA2Fast also achieved results close to SPEA2SVM, while spending only a small fraction (up to $\frac{1}{30}$) of the SPEA2SVM execution time.

Q3 – How effective are different classification strategies on distance-based meta-features considering meta-feature selection?

The SPEA2Fast approach also provided means to efficiently evaluate high-quality combinations of meta-features on different classification methods, providing substantial effectiveness gains (up to 50%) on classifications approaches that are very sensitive to complex meta-feature groups. These results add valuable evidence to our evaluation of meta-features on multiple classification approaches.

Q4 –Which combinations of meta-feature groups provide the core information to classify documents?

A comprehensive analysis of the core combinations of meta-feature groups considering the specificities of each dataset was also important to reveal details about which kind of information induced by meta-features can effectively provide discriminative patterns. We found important clues that explain why in most datasets we can use only a few meta-feature groups (up to seven [1] of twelve) to achieve the same or even better results in comparison to the ones achieved by the set of of all evaluated meta-feature groups. We also discovered that most datasets achieve reasonable results using only a specific combination of two meta-feature groups, and most datasets can achieve effectiveness very close to their best results with a specific combination of five meta-feature groups. These conclusions are important to provide a starting point for the exploitation of new meta-features in other classification tasks or related problems.

[1] cos_cnn, cos_cent, l1_knn, l2_knn, l2_cent and sum_cent.

7.1.2 Enriching, Adapting and Parallelizing Meta-Feature Generation

Q5 – Is it possible to improve the effectiveness of distance-based meta-features by enriching distance relationships with label information?

Our new SYN meta-features based on SDRs aim at explicitly capturing the relationships between document distances and label information. The EXT meta-features explore potential relationships that do exist between meta-features and the original words used to build them. Finally, the ERR meta-features aim at identifying hard-to-classify documents, complementing other meta-feature representations and allowing classifiers to adjust the learning process, as this information is now part of the document representation itself. Through a detailed and carefully designed set of experiments, we showed that our proposal can achieve significant gains of more than 12% against the best combination of meta-features found with SPEA2SVM in all considered datasets. Our group and factorial analyses demonstrated that the newly proposed meta-features groups did provide complementary information to each other, producing their best results when used altogether.

Q6 – How to exploit meta-features to provide effective results in the sentiment analysis context?

We also extended the use of meta-features to the sentiment analysis context, especially for the case of short messages. In order to exploit this context, we proposed new meta-features that exploit BM25 **?** to use short messages as queries and made use of automatically labeled external training data as well as weighted sentiment polarities of the neighborhood (obtained from lexicon-based methods) to infer the message's polarity. Our controlled experiments with these new meta features on nineteen benchmark datasets showed significant effectiveness improvements in most datasets over previous meta-features and the original bag-of-words representation. This provides evidence towards the benefits of carefully designing meta-features considering the specific characteristics of the task at hand.

Q7 – How to exploit the modern manycore GPU architectures to reduce the computational time to find the neighborhood of documents in text data?

Despite the important improvements in classification effectiveness, most meta-features are based on intensive use and direct application of the kNN algorithm to exploit local information regarding the neighborhood of training documents. However, intensive use of kNN, combined with the high dimensionality and sparsity of textual data, make this a challenging computational task for meta-feature generation. We have presented a very fast and scalable GPU-based approach for computing meta-feature document classification. In fact, different from other GPU-based kNN implementations, we avoided comparing the query document

with all training documents. Instead we built an inverted index in the GPU that is used to quickly find the documents sharing terms with the query document. Although the index does not allow a regular and predictable access to the data, we used a load balancing strategy to evenly distributed the computation among thousand threads in the GPU. After calculating the distances, we again used a massive number of threads to select the k smallest distance by implementing a truncated bitonic sort in the GPU followed by a merge operation in the CPU.

Our results showed very significant gains in speedup when compared to our baselines. In fact, running our baselines in our largest dataset demonstrated to be unfeasible, stressing the value of our efficiency contribution. And even in scenarios where the dataset is too big for our implementation (a single GPU), our approach could easily be extended by splitting the dataset, executing the kNN in each part, and merging the partial results. Thus, meta-feature based classification can be applied in huge collections of documents, taking a reasonable time and without requiring expensive hardware.

7.2 Future Work

This dissertation paved the way for numerous directions of future research based on the exploitation of meta-features for machine learning. In general, the application of machine learning on domains not discussed in this work can be focused on the exploitation of meaningful distances between documents instead of new classification algorithms capable of coping with the idiosyncrasies of their original feature spaces.

Particularly, one potential new venue of research consists in adapting our GPU implementation to efficiently compute other distance measures. Further improvements on efficiency can be investigated with distributed and multi-gpu implementations for exact and approximate nearest neighbor search algorithms. Moreover, parallel implementations can also improve the efficiency for generating and predicting our proposed SDRs.

In addition to the exploitation of SDRs, other strategies can be used to explicitly capture the relationships between document distances and label information. Particularly, future investigations to improve the underlying distance measures with *dataset-oriented distance learning techniques* are very promising. A possible scenario is the use of genetic programming for the evolution of similarity measures specially designed for the characteristics of specific textual datasets. Although there is some efforts on automatically finding good similarity measures (Agapitos, 2014), it is not clear how to adapt them to the high dimensionality and sparsity of textual data. In the same vein, future research on meta-features can benefit from similarity learning strategies based on the vast literature on Mahalanobis metric learning (Globerson and Roweis, 2005; Weinberger and Saul, 2009) or metrics derived from

potentially non-linear relationships in the data (Kedem et al., 2012).

Despite the potential of improving meta-features, scalability and overfitting issues should be taken into account by future research on meta-features that exploit similarity learning. Particularly, the efficiency of the classification using meta-features relies on the fast computation of learned distances, which poses new challenges for new parallel implementations. Moreover, the use of use labeled training data to learn new distances can bias the distance distribution to labeled documents, which might impair the generalization power of the classifiers.

Instead of exploiting only the training data, meta-features might enable the exploitation of information from huge external corpus such as the web pages categorized into topics provided by the open directory initiative DMOZ[2]. In this scenario, it is possible to build meta-features that exploit the similarity between a document and the set of documents from each DMOZ category, filtering out distant ones. As a result, the high similarity between an arbitrary document and a (well represented) DMOZ category can provide additional evidence that relates a document to a topic. This new information might be relevant for classification, specially in the scenario of limited training data.

Besides document distances, we could also exploit similarities between features to generate new meta-features. Specifically, we can adapt our proposed meta-features, which are based on document distances, to the fine-grained task of providing meta-features that represent individual words. The pre-required similarity between words can be computed using a wide range of measures, such as co-occurrence of words in documents (Figueiredo et al., 2011), mutual information, semantic (Kusner et al., 2015), lexical and phonetic similarities. It is worth noting that this strategy can provide a novel vectorial representation of words (i.e., a new kind of word embedding) adaptable to the classification task, since the proposed n-dimensional meta-features would rely on the distribution of distances and categories related to the neighbor words. Giving our vectorial representation of words, it is possible to use recent statistical techniques that combine the proposed word embeddings in a compact document representation (Lev et al., 2015).

We could also build meta-features based on alternative documents representations, such as the those based on Word embeddings (Kusner et al., 2015), masked-based representations as used in BERT (Devlin et al., 2019) and clusters of similar words (Cluwords (Viegas et al., 2019)), which in theory capture semantic information based on local co-occurrences of words in sentences. Meta-features generated with different document representations (and adequate distances for such spaces) might complement each other, consequently providing a more informative feature space for classification methods.

[2]http://dmoz.org

Finally, it is possible to combine all above suggestions into a single framework. However, as we have seen, noise and overfitting may still be issues we have to deal with. In this scenario, our meta-feature selection strategies can evaluate the most promising combinations and provide the adequate combination of meta-features that complement each other considering the specificities of each dataset. There is room for further improvements in our feature selection methods such as the use of statistical significance tests to deal with ties in the Pareto Frontier (Sousa et al., 2019) and better selection of individuals from the frontier when considering multi-objective scenarios (e.g., maximize effectiveness while reducing the complexity of the representation).

Bibliography

Agapitos, A. (2014). Higher order functions for kernel regression. In *Genetic Programming*, volume 85, pages 1–12.

Araújo, M., Gonçalves, P., Benevenuto, F., and Cha, M. (2014). ifeel: A system that compares and combines sentiment analysis methods. In *WWW'14*.

Baccianella, S., Esuli, A., and Sebastiani, F. (2010). Senti wordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC'10*.

Banea, C., Mihalcea, R., Wiebe, J., and Hassan, S. (2008). Multilingual subjectivity analysis using machine translation. In *EMNLP'08*, pages 127--135, Stroudsburg, PA, USA.

Barbosa, L. and Feng, J. (2010). Robust sentiment detection on twitter from biased and noisy data. In *COLING '10*, pages 36--44, Stroudsburg, PA, USA.

Biyani, P., Bhatia, S., and Caragea, C. (2014). Using non-lexical features for identifying factual and opinionative threads in online forums. *Knowledge-Based Systems*, 69(0):170 – 178.

Breiman, L. (2001). Random forests. *JMLR*, 45(1):5--32.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA.

Breiman, L. and Spector, P. (1992). Submodel selection and evaluation in regression – the x-random case. *International Statistical Review*, 60:291--319.

Canuto, S., Gonçalves, M. A., and Benevenuto, F. (2016). Exploiting new sentiment-based meta-level features for effective sentiment analysis. In *WSDM*, pages 53--62. ACM.

Canuto, S., Marcos, G., Santos, W., Rosa, T., and Wellington, M. (2015). Efficient and scalable metafeature-based document classification using massively parallel computing. In *SIGIR*, pages 333--342.

Canuto, S., Salles, T., Gonçalves, M. A., Rocha, L., Ramos, G., Gonçalves, L., Rosa, T., and Martins, W. (2014). On efficient meta-level features for effective text classification. In *CIKM*, pages 1709--1718.

Canuto, S., Salles, T., Rosa, T. C., and Gonçalves, M. A. (2019). Similarity-based synthetic document representations for meta-feature generation in text classification. In *Proceedings of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, pages 355--364, New York, NY, USA. ACM.

Canuto, S., Sousa, D. X., Goncalves, M. A., and Rosa, T. C. (2018). A thorough evaluation of distance-based meta-features for automated text classification. *IEEE Transactions on Knowledge and Data Engineering*, pages 1--1.

Chen, T., Tang, K., Chen, G., and Yao, X. (2012). A large population size can be unhelpful in evolutionary algorithms. *Theoretical Computer Science*, 436:54 – 70.

Czyzak, P. and Jaszkiewicz, A. (1998). Pareto simulated annealing - a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 7(1):34--47.

Dalip, D. H., Lima, H., Gonçalves, M. A., Cristo, M., and Calado, P. (2014). Quality Assessment of Collaborative Content With Minimal Information. *JCDL*, pages 201--210.

de Siqueira, G. O., Canuto, S. D., Gonçalves, M. A., and Laender, A. H. F. (2017). Automatic hierarchical categorization of research expertise using minimum information. In *Research and Advanced Technology for Digital Libraries - 21st International Conference on Theory and Practice of Digital Libraries, TPDL 2017, Thessaloniki, Greece, September 18-21, 2017, Proceedings*, pages 103--115.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171--4186.

Diakopoulos, N. A. and Shamma, D. A. (2010). Characterizing debate performance via aggregated twitter sentiment. In *SIGCHI'10*, pages 1195--1198. ACM.

et al., E. Z. (2001). SPEA2: Improving the Strength Pareto Evolutionary Algorithm. *EMDOCAIP*.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). LIBLINEAR: A library for large linear classification. *JMLR*, 9:1871--1874.

Feldman, R. (2013). Techniques and applications for sentiment analysis. *Commun. ACM*, 56(4):82--89.

Figueiredo, F., Rocha, L., Couto, T., Salles, T., Gonçalves, M. A., and Meira Jr., W. (2011). Word co-occurrence features for text classification. *Inf. Syst.*, 36(5):843--858.

Fisher, R. A. (1925). *Statistical Methods for Research Workers*. Oliver and Boyd, Edinburgh.

Fotis Aisopos (2014). Manually annotated sentiment analysis twitter dataset ntua.

Garcia, V., Debreuve, E., and Barlaud, M. (2008). Fast k nearest neighbor search using gpu. In *CVPR Workshops*, pages 1--6.

Globerson, A. and Roweis, S. T. (2005). Metric learning by collapsing classes. In *NIPS*, pages 451–458.

Go, A., Bhayani, R., and Huang, L. (2009). Twitter sentiment classification using distant supervision. Technical report, Stanford.

Godbole, S. and Sarawagi, S. (2004). Discriminative methods for multi-labeled classification. In *Proc. PAKDD*, pages 22–30.

Gonçalves, P., Araújo, M., Benevenuto, F., and Cha, M. (2013). Comparing and combining sentiment analysis methods. In *COSN'13*.

Gonçalves, P., Dalip, D. H., Costa, H., Gonçalves, M. A., and Benevenuto, F. (2016). On the combination of "off-the-shelf" sentiment analysis methods. In *SAC'16*. ACM.

Gopal, S. and Yang, Y. (2010). Multilabel classification with meta-level features. In *Proc. SIGIR*, pages 315--322.

Hall, M. A. (2000). Correlation-based feature selection for discrete and numeric class machine learning. ICML, pages 359--366, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Hastie, T., Tibshirani, R., and Friedman, J. H. (2003). *The Elements of Statistical Learning*. Springer.

Hull, D. (1993). Using statistical testing in the evaluation of retrieval experiments. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '93, pages 329--338, New York, NY, USA. ACM.

Hutto, C. J. and Gilbert, E. (2014). VADER: A parsimonious rule-based model for sentiment analysis of social media text. In *ICWSM'14*.

Jain, R. (1991). *The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling.* Wiley professional computing. Wiley.

Johnson, R. and Zhang, T. (2015). Effective use of word order for text categorization with convolutional neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 103--112, Denver, Colorado. Association for Computational Linguistics.

Kedem, D., Tyree, S., Sha, F., Lanckriet, G. R., and Weinberger, K. Q. (2012). Non-linear metric learning. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 2573--2581. Curran Associates, Inc.

Kim, H., Howland, P., and Park, H. (2005). Dimension reduction in text classification with support vector machines. *J. Mach. Learn. Res.*, 6:37--53.

Kim, S.-B., Han, K.-S., Rim, H.-C., and Myaeng, S.-H. (2006). Some effective techniques for naive bayes text classification. *IEEE TKDE*, 18(11):1457–1466.

Kira, K. and Rendell, L. A. (1992). The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, AAAI'92, pages 129--134. AAAI Press.

Kiritchenko, S., Zhu, X., and Mohammad, S. M. (2014). Sentiment analysis of short informal texts. *J. Artif. Int. Res.*, 50(1):723--762.

Kuang, Q. and Zhao, L. (2009). L.: A practical gpu based knn algorithm. In *In Proc. ISCSCT)*, pages 151--155.

Kusner, M. J., Sun, Y., Kolkin, N. I., and Weinberger, K. Q. (2015). From word embeddings to document distances. In *ICML'15*, pages 957--966.

Kyriakopoulou, A. and Kalamboukis, T. (2006). Text classification using clustering. In *ECML-PKDD'06*.

Kyriakopoulou, A. and Kalamboukis, T. (2007). Using clustering to enhance text classification. In *SIGIR'07*, pages 805--806.

Kyriakopoulou, A. and Kalamboukis, T. (2008). Combining Clustering with Classification for Spam Detection in Social Bookmarking Systems. RSDC '08.

Lan, M., Tan, C.-L., and Low, H.-B. (2006). Proposing a new term weighting scheme for text categorization. In *Proc. AAAI*, pages 763--768.

Laumanns, M., Zitzler, E., and Thiele, L. (2001). *On The Effects of Archiving, Elitism, and Density Based Selection in Evolutionary Multi-objective Optimization*, pages 181--196. Springer Berlin Heidelberg, Berlin, Heidelberg.

Lev, G., Klein, B., and Wolf, L. (2015). In defense of word embedding for generic text representation. In *NLDB '15*.

Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *JMLR.*, 5:361--397.

Li, B., Li, J., Tang, K., and Yao, X. (2015). Many-objective evolutionary algorithms: A survey. *ACM Comput. Surv.*, 48(1):13:1--13:35.

Liang, S., Wang, C., Liu, Y., and Jian, L. (2009). Cuknn: A parallel implementation of k-nearest neighbor on cuda-enabled gpu. In *IEEE YC-ICT'09.*, pages 415--418.

Liu, H. and Setiono, R. (1995). Chi2: Feature selection and discretization of numeric attributes. In *International Conference on Tools with Artificial Intelligence*, pages 388--391.

Liu, H. and Setiono, R. (1996). A probabilistic approach to feature selection - a filter solution. In *ICML*, pages 319–327.

Luengo, J., García, S., and Herrera, F. (2009). A study on the use of statistical tests for experimentation with neural networks: Analysis of parametric test conditions and non-parametric tests. *Expert Systems with Applications*, 36:7798–7808.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge Press, NY, USA.

Narr, S., Hulfenhaus, M., and Albayrak, S. (2012). Language-independent twitter sentiment analysis. *Knowledge Discovery and Machine Learning (KDML)*, pages 12--14.

Pang, B. and Lee, L. (2004). A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL'04*, Stroudsburg, PA, USA.

Pang, G., Jin, H., and Jiang, S. (2015). Cenknn: a scalable and effective text classifier. *Data Mining and Knowledge Discovery*, 29(3):593--625.

Pappas, N. and Popescu-Belis, A. (2013). Sentiment analysis of user comments for one-class collaborative filtering over ted talks. In *SIGIR'13*, pages 773--776. ACM.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *JMLR*, 12:2825--2830.

Platt, J. C. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, pages 61--74. MIT Press.

Raskutti, B., Ferrá, H. L., and Kowalczyk, A. (2002). Using unlabelled data for text classification through addition of cluster parameters. In *ICML'02*, pages 514--521.

Ribeiro, F. N., Araújo, M., Gonçalves, P., Gonçalves, M. A., and Benevenuto, F. (2016). Sentibench-a benchmark comparison of state-of-the-practice sentiment analysis methods. *EPJ Data Science*, 5(1):1--29.

Rocha, L., Mourão, F., Pereira, A., Gonçalves, M. A., and Meira, Jr., W. (2008). Exploiting temporal contexts in text classification. In *Proc. CIKM*, pages 243--252.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1--47.

Sengupta, S., Harris, M., Garland, M., and Owens, J. D. (2011). Efficient parallel scan algorithms for many-core gpus. *Sci. Comp. with Multicore and Acc.*, pages 413--442.

Sismanis, N., Pitsianis, N., and Sun, X. (2012). Parallel search of k-nearest neighbors with synchronous operations. In *HPEC*, pages 1–6. IEEE.

Sokolova, M. and Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Inf. Process. Manage.*, 45(4):427--437.

Sousa, D. X., Canuto, S. D., Gonçalves, M. A., Rosa, T. C., and Martins, W. S. (2019). Risk-sensitive learning to rank with evolutionary multi-objective feature selection. *ACM Trans. Inf. Syst.*, 37(2):24:1--24:34.

Sousa, D. X. D., Canuto, S. D., Rosa, T. C., Martins, W. S., and Gonçalves, M. A. (2016). Incorporating risk-sensitiveness into feature selection for learning to rank. In *CIKM*, pages 257--266, New York, NY, USA. ACM.

Srinivas, M. and Patnaik, L. M. (1994). Genetic algorithms: a survey. *IEE CS*.

Su, F. and Markert, K. (2008). From words to senses: A case study of subjectivity recognition. In *COLING '08*, pages 825--832, Stroudsburg, PA, USA.

Thelwall, M. (2013). Heart and soul: Sentiment strength detection in the social web with sentistrength.

Thelwall, M., Buckley, K., Paltoglou, G., Cai, D., and Kappas, A. (2010). Sentiment in short strength detection informal text. *J. Am. Soc. Inf. Sci. Technol.*, 61(12):2544--2558.

Urbano, J., Lima, H., and Hanjalic, A. (2019). Statistical significance testing in information retrieval: An empirical analysis of type i, type ii and type iii errors. In *Proceedings of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, pages 505--514, New York, NY, USA. ACM.

Viegas, F., Canuto, S., Gomes, C., Luiz, W., Rosa, T., Ribas, S., Rocha, L., and Gonçalves, M. A. (2019). Cluwords: Exploiting semantic word clustering representation for enhanced topic modeling. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19, pages 753--761, New York, NY, USA. ACM.

Wang, X., Zhang, C., and Wu, M. (2015). Sentiment classification analysis of chinese microblog network. In Mangioni, G., Simini, F., Uzzo, S. M., and Wang, D., editors, *Complex Networks VI*, volume 597 of *Studies in Computational Intelligence*, pages 123–129. Springer.

Weinberger, K. Q. and Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *JMLR*, pages 207--244.

Wiebe, J. and Riloff, E. (2005). Creating subjective and objective sentence classifiers from unannotated texts. In *CICLing'05*, pages 486--497, Berlin, Heidelberg. Springer-Verlag.

Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Inf. Ret.*, 1:69–90.

Yang, Y. and Gopal, S. (2012). Multilabel classification with meta-level features in a learning-to-rank framework. *JMLR*, 88:47–68.

Yang, Y. and Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In *ICML*, pages 412--420, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Yu, L. and Liu, H. (2003). Feature selection for high-dimensional data: A fast correlation-based filter solution. In *ICML*, pages 856–863. AAAI Press.

Zhang, H., Berg, A. C., Maire, M., and Malik, J. (2006). Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, pages 2126--2136, Washington, DC, USA. IEEE Computer Society.

Zitzler, E., Laumanns, M., and Thiele, L. (2001). SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In *Proceedings of the EUROGEN'2001 Conference*, pages 95--100.