Joana Madeira Martins Costa

Information Extraction from Crowds in Dynamic Environments

PhD Thesis in Doctoral Program in Information Science and Technology, supervised by Professor Bernardete Ribeiro and Professor Catarina Silva and submitted to the Faculty of Sciences and Technology of the University of Coimbra

February 2018

· U C ·

UNIVERSIDADE DE COIMBRA

Information Extraction from Crowds in Dynamic Environments

Joana Costa

Department of Informatics Engineering

University of Coimbra

Supervisors:

Bernardete Ribeiro

and

Catarina Silva

February 2018

'Extração de informação colaborativa em ambientes dinâmicos'

Orientadores:

Bernardete Ribeiro
Professora Associada com Agregação
Departamento de Engenharia Informática
Universidade de Coimbra

Catarina Silva
Professora Adjunta
Departamento de Engenharia Informática
Escola Superior de Tecnologia e Gestão
Instituto Politécnico de Leiria

Abstract

Today's society is based on information sharing. The Internet, once exclusively a computer network, evolved to a network of personal devices. Mobile phones, tablets, watches, and even common household devices are acquiring, sending, and receiving data much faster than we would imagine possible a few years ago. However, as we are producing more data, we are becoming less aware of reliable information.

Social networks are paradigmatic to this scenario. Highly accepted by most Internet users, they became a potential source of information. The extraordinary and relevant amount of data made available in social networks may be used towards the resolution of challenges faced by individuals and companies. Users can also contribute with their skills to problem solving, which is the idea in which crowdsourcing lays its foundation, i.e., a multitude of non-experts that contribute to real-world problem solving. Humans' innate aptitude to deal with intrinsically subjective tasks, and to perceive related concepts, turns them into a valuable resource.

In this thesis we present novel and efficient techniques to deal with information extraction from crowds in dynamic environments. We describe the paths we have explored and propose a framework that integrates the acquired knowledge to deal with some of the major challenges of learning in such environments.

One of the techniques that is able to introduce important information into the learning process is active learning. When combined with crowdsourcing's potential for problem solving, active learning can be of major interest. We investigate an active learning strategy that uses an assertive supervisor versus crowdsourcing. A recommendation system is used as case study, and

active learning is also proposed for customization purposes. Active learning allows the integration of user feedback into the learning process, thus defining a customized crowd used for model customization.

To tackle the problem of effectively learning from crowds in dynamic environments, we use the Twitter social network as case study. We use hashtags as Twitter message classification targets, and propose the definition of semantic meta-hashtags, which cluster similar messages, to improve classification performance.

The impact of longstanding messages in Twitter is also studied to understand how informative can past events be to current learning models. A technique to define the best set of training examples using dynamic ensembles in Twitter is proposed, along with a study regarding the effect of using different metrics for combining ensembles' models, specifically performance-based metrics.

Three different models to learn in dynamic environments are proposed: a time-window model, an ensemble-based model, and an incremental model. The time-window model is characterized by taking into account recent information in a given time-window. The ensemble model is based on the idea that the use of a committee of classifiers can provide better results than the best of the single classifiers, if correctly combined. Finally, the incremental model is characterized by retaining in a single classifier all the information gathered over time.

A benchmark dataset with drift in Twitter, where real tweets are artificially timestamped to represent different drift patterns, is also proposed. To define such dataset the Drift Oriented Tool System (DOTS), a framework that allows the definition and generation of text-based datasets with different drift patterns, is used. DOTS is made publicly and freely available and it can be used by researchers to evaluate and validate learning strategies in dynamic environments.

Another important contribution is Drift Adaptive Retain Knowledge (DARK), a framework to effectively learn in dynamic environments in text classification scenarios. It uses an ensemble of classifiers with dynamic

weighting schemes and variable training window sizes for model adaptation in incremental learning. A comparative study of the performance of DARK with benchmark solutions in the field, namely the Learn++.NSE algorithm, is presented and demonstrates its potential as a learning strategy in dynamic environments.

The techniques proposed in this thesis deal with the some of the most important challenges regarding learning in dynamic environments. Different approaches are used, such as active learning, crowdsourcing, and ensembles. Experimental results show a classification improvement when compared to benchmark solutions in a real-world dataset. We finish this thesis by summarizing the contributions we have made and proposing further research paths.

Keywords: *information extraction*, *crowdsourcing*, *dynamic environments*, *social networks*, *adaptive learning*, *ensembles*

Resumo

A sociedade dos dias de hoje é baseada na partilha de informação. A Internet, tendo já sido exclusivamente uma rede de computadores, evoluiu para uma rede de dispositivos pessoais. Telemóveis, tablets, relógios, e eletrodomésticos adquirem, enviam e recebem dados a uma velocidade que antigamente não imaginávamos possível. Produzimos mais dados, mas estamos a tornar-nos incapazes de obter informação relevante.

As redes sociais são paradigmáticas neste contexto. Altamente aceites pela maioria dos utilizadores da Internet, tornaram-se numa potencial fonte de informação. A quantidade extraordinária e a relevância dos dados hoje disponibilizados pelas redes sociais podem ser usadas na resolução de imensos desafios enfrentados hoje em dia por indivíduos e empresas. Os utilizadores das redes socias podem mesmo contribuir com conhecimento para a resolução de problemas, uma ideia de base do *crowdsourcing*, onde uma multidão indiferenciada pode contribuir para a resolução de problemas do mundo real. A aptidão inata dos seres humanos para lidarem com tarefas altamente subjectivas, e para perceber a relação entre conceitos, torna-os num valioso recurso.

Nesta tese apresentamos novas e eficientes técnicas que nos permitem lidar com a extracção de informação colaborativa em ambientes dinâmicos. Descrevemos as abordagens desenvolvidas e propomos uma *framework* que integra o conhecimento adquirido, de forma a lidar com alguns dos principais desafios de aprendizagem em ambientes desta natureza.

Uma das técnicas que permite a introdução de informação importante ao processo de aprendizagem é a aprendizagem activa. Quando combinada com a potencialidade da utilização de *crowdsourcing* na resolução de problemas,

a aprendizagem activa pode relevar-se imensamente interessante. Investigámos a utilização de uma estratégia de aprendizagem activa que compara o uso de um supervisor assertivo em detrimento de *crowdsourcing*. Foi utilizado um sistema de recomendação como caso de estudo, tendo a aprendizagem activa sido usada para fins de personalização. A aprendizagem activa permite a integração de *feedback* no processo de aprendizagem, possibilitando assim a definição de grupos de utilizadores para a personalização de modelos.

Para contribuir para o problema da aprendizagem efectiva em ambientes dinâmicos com recurso a grupos, utilizámos a rede social Twitter como caso de estudo. Recorremos às *hashtags* das mensagens de Twitter como alvo de classificação e propusemos a definição de *meta-hashtags* semânticas, que agrupam mensagens similares para melhorar o desempenho da classificação.

O impacto da eternidade de mensagem no Twitter é também estudado para perceber quão informativos podem ser os eventos passados em modelos de aprendizagem atuais. Uma técnica que define qual o melhor conjunto de treino a utilizar em *ensembles* dinâmicos no Twitter é também proposta, assim como um estudo referente ao efeito da utilização de diferentes métricas para combinar modelos de ensemble, nomeadamente métricas baseadas em *performance*.

São propostos três modelos diferentes para aprendizagem em ambientes dinâmicos: um baseado em janelas temporais, outro baseado em *ensembles* e outro incremental. O modelo baseado em janelas temporais é caracterizado por ter em conta a informação mais recente de acordo com uma determinada janela temporal. O modelo baseado em *ensembles* usa um grupo de classificadores que, quando corretamente combinados, permitem encontrar uma solução melhor que o a utilização do melhor classificador do grupo. Finalmente, o modelo incremental caracteriza-se por reter num único classificador toda a informação recolhida ao longo do tempo.

É apresentado um *dataset* de referência com *drift* em Twitter, onde *tweets* reais são temporalmente marcados artificialmente, por forma a representar diferentes padrões de *drift*. Para definir o referido *dataset* foi utilizada

a *framework* Drift Oriented Tool System (DOTS), que permite não só a definição como a geração de *datasets* de texto com diferentes padrões de *drift*. A DOTS foi tornada pública e está disponível gratuitamente para possibilitar a sua utilização por investigadores, permitindo assim que possam avaliar e validar as suas estratégias de aprendizagem em ambientes dinâmicos.

Outra importante contribuição é a *framework* Drift Adaptive Retain Knowledge (DARK), que permite uma aprendizagem efectiva em ambientes dinâmicos em cenários de classificação de texto. Utiliza *ensembles* de classificadores com pesos dinâmicos e janelas de aprendizagem de tamanho variável para adaptação de modelos em aprendizagem incremental. É também apresentado um estudo comparativo da *performance* da DARK com soluções de topo na área, nomeadamente o algoritmo Learn++.NSE, e demonstrando assim o seu potencial como estratégia de aprendizagem em ambientes dinâmicos.

As técnicas propostas nesta tese fazem face a alguns dos mais importantes desafios no que toca a aprendizagem em ambientes diâmicos. São utilizadas diferentes abordagens, tais como aprendizagem activa, *crowdsourcing* e *ensembles*. Os resultados experimentais demonstram melhorias na classificação quando comparados com soluções de topo em *dataset* reais. Terminamos esta tese sumariando as contribuições e propondo novos caminhos de investigação.

Palavras-chave: *extração de informação*, *crowdsourcing*, *ambientes dinâmicos*, *redes sociais*, *aprendizagem adaptativa*, *ensembles*

Acknowledgements

I would like to express my deepest gratitude to both my supervisors, Professor Bernardete Ribeiro and Professor Catarina Silva. Their continuous guidance, encouragement and support helped me to complete this research work.

I would also like to thank Professor Mário Antunes for the fruitful collaboration and the discussions regarding dynamic environments that greatly influenced this work.

This thesis would not be possible without the efforts of many people that have provided help and insights at different times. I would like to thank my colleagues in the Informatics Department of the School of Technology and Management (ESTG) of the Polytechnic Institute of Leiria (IPLeiria) for lightening my burden whenever possible during the last few years. I also wish to thank the Center of Informatics and Systems (CISUC) of the Informatics Department of the Faculty of Sciences and Technology of the University of Coimbra, for the means provided during my research. Finally, to the Portuguese Science and Technology Foundation (FCT), for financially supporting the attending of a conference during the early years.

To my friends, for the joyful and de-stressful moments.

Last, but not least, I would like to thank my family, my parents Carlos and Isabel and my sister Inês, for their unconditional love and support, and Pedro, for his endless patience during these uneasy times.

Contents

List of Figures v

List of Tables vii

Acronyms ix

Notation xi

I Fundamentals 1

1 Introduction 3

1.1 Motivation . . . 3

1.2 Contributions . . . 4

1.3 Thesis structure . . . 8

2 Dynamic Environments 11

2.1 Introduction . . . 11

2.2 Problem Definition . . . 12

2.3 Information Extraction . . . 15

2.3.1 Document Representation . . . 17

2.3.2 Preprocessing Methods . . . 19

2.3.3 Learning Methods . . . 23

2.3.4 Evaluation . . . 28

2.4 Learning in Dynamic Environments . . . 30

2.4.1 Active Approaches . . . 31

2.4.2 Passive Approaches . . . 34

2.5 Conclusion 36

II Learning from Crowds **39**

3 Crowdsourcing **41**

3.1 Introduction 41
3.2 Definition 42
3.3 Crowdsourcing Perspectives 43
3.4 Crowd Characteristics 45
3.5 Applications 47
3.6 Crowd Platforms 48
3.7 Challenges 51
3.8 Conclusion 51

4 Active Learning and Crowdsourcing **53**

4.1 Introduction 53
4.2 Crowdsourcing: Case Study 54
4.2.1 Problem Setting 56
4.2.2 Jester Dataset 56
4.3 Enhancing Classification with Active Learning 57
4.3.1 Baseline Approach 58
4.3.2 Research Design with Active Learning 60
4.3.3 Experimental Results and Analysis 61
4.4 Crowd Customization 63
4.4.1 Personal Active Approach 64
4.4.2 Custom Active Approach 65
4.4.3 Closeness Metrics 66
4.4.4 Experimental Results and Analysis 66
4.5 Conclusion 67

III Learning in Dynamic Environments **69**

5 Social Networks **71**

5.1 Introduction 71

5.2 Background . . . 72
5.2.1 Relevant Social Networks . . . 73
5.2.2 Twitter . . . 76
5.3 Paramount Applications . . . 79
5.4 Current Research . . . 80
5.4.1 Research Paths . . . 81
5.4.2 Challenges . . . 82
5.5 Twitter Classification . . . 86
5.5.1 Definition . . . 86
5.5.2 Dataset . . . 87
5.5.3 Experimental Results and Analysis . . . 88
5.6 The Use of Meta-hashtags in Twitter Classification . . . 90
5.6.1 Dataset . . . 91
5.6.2 Experimental Results and Analysis . . . 91
5.7 Conclusion . . . 94

6 Learning in Dynamic Social Environments 97
6.1 Introduction . . . 97
6.2 Drift Oriented Tool System . . . 98
6.2.1 Specific Features of DOTS . . . 101
6.2.2 Using DOTS in Twitter Classification . . . 102
6.3 Dynamic Learning Models . . . 104
6.3.1 Time-window Model . . . 105
6.3.2 Incremental Model . . . 105
6.3.3 Ensemble Model . . . 106
6.3.4 Dataset . . . 106
6.3.5 Experimental Results and Analysis . . . 109
6.4 Dataset for Drift in Twitter . . . 111
6.5 Time in Dynamic Environments . . . 114
6.5.1 Batch Learning Model . . . 115
6.5.2 Experimental Results and Analysis . . . 115
6.6 Conclusion . . . 121

7 Adaptive Learning in Dynamic Environments **123**
7.1 Introduction . . . 123
7.2 DARK Framework . . . 125
7.3 Combining Ensembles' Models . . . 128
7.4 Ensemble Model Enrichment . . . 131
7.5 Comparative Study with Learn++.NSE . . . 135
7.5.1 Learn++.NSE . . . 136
7.5.2 Experimental Results and Analysis . . . 136
7.5.3 Statistical Evaluation of the Comparative Study . . . 140
7.6 Conclusion . . . 143

IV Conclusions and Future Work **147**

8 Conclusion **149**
8.1 Introduction . . . 149
8.2 Discussion and Conclusions . . . 150
8.3 Future Work . . . 153

V Appendix **157**

A Published Materials **159**
A.1 Journal Papers . . . 159
A.2 Conference Papers . . . 159
A.3 Software . . . 161

References **163**

List of Figures

2.1 Graphical representation of concept drift types 14
2.2 Document collection representation matrix 18

4.1 Example of a joke from the Jester dataset 57
4.2 Enhancing classification with active learning: Baseline approach 58
4.3 SVM Optimal Separating Hyperplane (OSH) representation 59
4.4 Enhancing classification with active learning: Active approach 60
4.5 Crowd customization: Personal Active and Custom Active approaches . 64

5.1 An example of a tweet 77
5.2 Tweet and hashtag representation 78
5.3 Most frequent hashtags present in the Twitter dataset 88
5.4 The use of meta-hashtags in Twitter classification: proposed approach . 90

6.1 The DOTS interface 99
6.2 The DOTS framework 99
6.3 DOTS: add task feature 100
6.4 Time in dynamic environments: graphical results by training window . . 116

7.1 The DARK framework 126
7.2 Ensemble model enrichment: the use of misclassified examples 133

8.1 Context and framing of the thesis 150

List of Tables

2.1 Contingency table for binary classification 29
2.2 Learning in dynamic environments: approaches' summary 32

4.1 Enhancing classification with active learning: comparative results 61
4.2 Crowd customization: performance results for active approaches 67

5.1 Twitter classification: comparative results 88
5.2 The use of meta-hashtags in Twitter classification: positive documents . 92
5.3 The use of meta-hashtags in Twitter classification: comparative results . 93

6.1 DOTS: representation of the CSV table 103
6.2 Using DOTS in Twitter classification: performance results 104
6.3 Twitter dataset: correspondence between type of drift and hashtag . . . 107
6.4 Twitter dataset: frequency of hashtags over time 108
6.5 Dynamic learning models: comparative results 110
6.6 Dataset for drift in Twitter: mapping between type of drift and hashtag 114
6.7 Time in dynamic environments: comparative results 117
6.8 Time in dynamic environments: effect of increasing the training window 118

7.1 Combining ensembles' models: performance results (F_1) 129
7.2 Combining ensembles' models: performance results (recall) 130
7.3 Ensemble model enrichment: comparative results 134
7.4 Comparative study with Learn++: performance results (window size = 1) 139
7.5 Comparative study with Learn++: results (training window size = 4) . . 139
7.6 McNemar's test of Learn++.NSE.SVM against DARK (window size = 1) 144
7.7 McNemar's test of Learn++.NSE.SVM against DARK (window size = 4) 144

7.8 McNemar's test of Learn++.NSE.CART against DARK (window size = 1) 145
7.9 McNemar's test of Learn++.NSE.CART against DARK (window size = 4) 145

Acronyms

AL	Active Learning
API	Application Programming Interface
ARFF	Attribute-Relation File Format
CART	ClAssification and Regression Tree
CDS	Concept Drift and SMOTE
COMPOSE	COMPacted Object Sample Extraction
CSV	Comma-Separated Values
DARK	Drift Adaptive Retain Knowledge
DNF	Disjunctive Normal Form
DOTS	Drift Oriented Tool System
DT	Decision Trees
DWM-WIN	Dynamic Weighted Majority-WINnow
ECOC	Error-Correcting Output Codes
ELM	Extreme Learning Machines
FN	False Negative
FP	False Positive
G+	Google+
HITS	Human Intelligence TaskS
IDF	Inverse Document Frequency
JIT	Just-In-Time
LSI	Latent Semantic Indexing
MLP	MultiLayer Perceptron
MOA	Massive Online Analysis
NER	Name Entity Recognition
NIE	Nonstationary and Imbalanced Environments

ACRONYMS

NLP	Natural Language Processing
NSE	Non-Stationary Environments
OSH	Optimal Separating Hyperplane
PCA	Principal Components Analysis
POS	Part-Of-Speech
SEA	Streaming Ensemble Algorithm
SMOTE	Synthetic Minority Over-sampling TechniquE
SMS	Short Message Service
SRL	Semantic Role Labeling
SVM	Support Vector Machines
TF	Term Frequency
TN	True Negative
TP	True Positive
VFDT	Very Fast Decision Trees
WEKA	Waikato Environment for Knowledge Analysis
YA	Yahoo! Answers

Notation

$\|.\|$	Number of elements in a set
$\|\|.\|\|$	$L_2 - norm$
A	Collection of attributes
α	Lagrange multipler in SVMs optimization
c_j	Class
C	Set of classes
C_i^a	Classification of item i given by user a
$\mathcal{C}$	Classifier
$\mathcal{C}^t$	Classifier created at time-window t
$d(.,.)$	Euclidean distance between two points
$\mathbf{d}, \mathbf{d}_i$	Document
$\mathcal{D}, \mathcal{T}$	Collection of documents
e^t	Ensemble prediction function at time-window t
$\mathcal{E}$	Ensemble
$\mathcal{E}^t$	Ensemble created at time-window t
f	Target function
F_β, F_1	van Rijsbergen's measure
h	Prediction function
h^t	Prediction function at time-window t
$\mathcal{J}$	Collection of jokes
λ, β, γ	Parameters for Rocchio's method
Neg	Negative examples, not belonging to a category
$P(.)$	Probability
$p(.)$	Probability function
$p(.\|.)$	Conditional probability

Pos	Positive examples, belonging to a category
q	Query
t	Time-window
t_i	Time point
$\mathcal{W}$	Set of features (words or terms), dictionary
w_k	Word or term
w_{ik}	Value representing word w_k in a document $\mathbf{d}_i$
ω_k	Weight of term w_k for a given model
ω_{ik}	Weight of term w_k in document $\mathbf{d}_i$ for a given model
ω_{qk}	Weight of term w_k in query q
X	Input variables
y	Target variable

Part I

Fundamentals

Chapter 1

Introduction

1.1 Motivation

The World Wide Web unquestionably created a new information sharing paradigm, turning everyone into an author, able to create and disseminate information in numerous formats, and thus being responsible for a deluge of data which is now *status quo*.

We can undoubtedly benefit from all these data, as everyone can input valuable information, e.g. by spreading news, communicating recently occurred phenomena, or simply choosing the most convenient answer to a given question. This scenario is commonly described as *crowdsourcing*, a low-cost human work force that is able to solve problems, specially when benefiting from humans' innate capability of dealing with intrinsically subjective tasks, a capacity that machines struggle to achieve.

On the other hand, the main drawback of this deluge of data is the inability to easily perceive important, significant, and accurate information, not only because the amount of data can be overwhelming to process, but also because time plays an important role by fastly out-dating information.

Although it may be easy to understand the importance of extracting this information, new and emergent challenges have arisen. Machine learning approaches usually build models that can perform well in newly unseen data depending on algorithms and data for each problem. However, in dynamic environments, as data evolves each time faster, models' performance can be hampered when there are significant changes

between data distributions used to define models and those responsible for generating new data.

This thesis describes the research done in information extraction from crowds in dynamic environments and proposes a framework to deal with the major challenges of learning in those environments.

1.2 Contributions

Huge amounts of data are generated on a daily basis in social networks, news feeds, research projects, or companies, but we have created a traditional needle in a haystack problem, as we have more data than we are truly able to manually process. Processing the available data we have today could help in different ways, for instance perceiving in real-time the eminence of natural disasters and increasing readiness on how to handle them, finding molecular patterns which could lead to new treatments, or better distributing goods and resources by identifying more suitable distribution patterns.

In the context of Internet online users, time also plays a significant role. A typical example is the prediction of an email's relevance, as the importance we give to an email changes over time, i.e., today we can be involved in a project and emails referring that project can be important but in a couple of months, when we possibly have embraced a different project, those emails may no longer be as relevant. It is very challenging for a learning algorithm to cope with these changes, because if a learning algorithm is designed to quickly react to the first signs of drift, it can easily be misled into overreacting to noise and erroneously interpreting it as drift. On the other hand, if it is designed to be highly robust in the face of noise, it will probably take too long to react to real changes, making it less sensitive than it would be required to learn in drift scenarios (Widmer & Kubat, 1996; Tsymbal, 2004).

Another current challenge is dealing with subjective tasks, like those influenced by contextual, emotional, and social constraints, which are extremely demanding tasks for a machine to cope with. One of the emergent solutions is the use of crowdsourcing. However, the contribution obtained for a given example may differ according to the heterogeneous background (e.g social, cultural, emotional, and scientific) expressed by the reliability of the annotators (refers to a person labelling a set of examples). Hence, the overall contribution provided by the crowd can be the result of different and even

opposite contributions provided by each annotator individually. Understanding the importance of each contribution to the final decision is also a challenging issue (Rodrigues *et al.*, 2013).

In this thesis we aim to contribute to the above mentioned challenges. The main contributions of this thesis research work can be summarized as follows:

- **Crowdsourcing as data source**.

  We are nowadays fully connected. Due to the continuous growth of Internet and mobile devices, everyone is connected everywhere. As a consequence, every user is a potential source of information anytime, anywhere. By providing their opinion about a particular emotion expressed on a post or picture, users are often required for professional advertisement, promoting services, or market sensing. We have explored this subject using a humour classification case study in Costa *et al.* (2011a) and Costa *et al.* (2011c). The task at hand is intrinsically subjective and complex, not only because humour and the perception of humorous content is quite dependent on cultural and educational backgrounds, but also because it can be particularly altered by contextual constraints. It is thus more appropriate to be tackled by humans, and therefore we proposed a classification and recommendation strategy based on crowdsourcing, showing that crowds can be used in machine learning as data source.

- **Active learning for customization purposes**.

  Along with crowdsourcing, we have used active learning to improve classification in highly subjective tasks, like customization. The rationale of this idea is that a machine learning algorithm can achieve greater accuracy with fewer training labels if it is allowed to choose the data from which it learns.

  In Costa *et al.* (2011c) we have used crowdsourcing and active learning to demonstrate the possible enhancements in classification by adding active learning examples classified by a crowd to an existing model. We have also proposed a similarity measure in Costa *et al.* (2013a) to determine the closeness of a specific user to each crowd contributor, and hence define the more appropriate crowd to be tuned

to replace the user profile definition. Different levels of customization were also compared using crowd-based information.

In Costa *et al.* (2013b) we proposed an active learning framework to allow non-experts classification to be performed by crowds and to be used to define the user profile, mitigating the labelling effort normally requested to the user. The framework is designed to be generic and suitable to different scenarios, whilst customizable for each specific user.

- **Preprocessing in social networks**.

  Given the widespread use of social networks, research efforts to extract information from such media have gained relevance. One of the major challenges is the multitude of users with different social and educational contexts, which often results in different ways of communicating the same concept. For example, the definition of a hashtag for a given concept, acting as a topic for a message and freely introduced by users, is extremely dependent on age, gender, location, and even social upbringing, thus becoming easily biased.

  Specific preprocessing strategies can tackle these disparities, as we have proposed in Costa *et al.* (2013c). By defining semantic meta-hashtags that cluster similar messages we are able to deal with the aforementioned bias and improve classification performance. The main idea of defining such semantic meta-hashtags is to avoid classification problems raised by having different classes that represent the same concept.

- **Learning in dynamic environments**.

  Modern challenges in machine learning include non-stationary environments. Due to their intrinsically dynamic nature, learning in these environments is not an easy task, as models have to deal both with continuous learning processes and also with the acquisition of new concepts. Current challenges include temporal data streams, drift, and non-stationary scenarios, often with text data, whether in social networks or in business systems. In Costa *et al.* (2014) we have studied different strategies to learn in dynamic environments. Three different models were

proposed: a time-window model, an ensemble-based model, and an incremental model. The time-window model is characterized by taking into account recent information in a given time-window. The ensemble model is based on the idea that the use of a committee of classifiers can provide better results than the best of the single classifiers, if correctly combined. Finally, the incremental model is characterized by retaining in a single classifier all the information gathered over time.

In dynamic environments it is also crucial to understand how informative can past events be to current learning models and for how long is it relevant to store previously seen examples, thus avoiding unnecessary computational burden. In this regard, in Costa *et al.* (2015b), we have studied the impact of longstanding messages in micro-blogging classification using different training window sizes in the learning process, devising a strategy to determine the best cost benefit ratio for the training window size.

Machine learning approaches often focus on optimizing the algorithm rather than assuring that the source data is as rich as possible. However, when it is possible to enhance the input examples to construct models, one should consider it thoroughly. In Costa *et al.* (2016), we propose a technique to define the best set of training examples using dynamic ensembles in text classification scenarios. Besides defining the best set for training purposes, we have investigated in Costa *et al.* (2017b) the effect of using different metrics for combining ensembles' models, specifically performance-based metrics.

- **Drift Oriented Tool System (DOTS)**.

  It is difficult to find adequate benchmarks in dynamic environments, as there are multiple drift pattern types: concepts that appear and disappear suddenly, recurrently, or even gradually or incrementally. Thus, researchers strive to propose and test algorithms and techniques to deal with drift. In Costa *et al.* (2015a) we have presented DOTS, the Drift Oriented Tool System, a framework developed to allow for the definition and generation of text-based datasets where drift characteristics can be thoroughly defined, implemented, and tested. The framework is freely available in `http://dotspt.sourceforge.net` and can be used in different

text classification problems, exporting datasets in multiple widely known formats.

- **Drift Adaptive Retain Knowledge (DARK)**.

  Various efforts have been pursued in machine learning settings to learn in dynamic environments, specially because of their non-trivial nature, since changes occur between the distribution data used to define the model and the current environment. In Costa *et al.* (2017a) we have presented DARK, the Drift Adaptive Retain Knowledge framework. The rationale of DARK is to use ensembles of classifiers to integrate multiple experts with different characteristics and benefit from their multitude, specially as they are created in different moments. Due to its modular structure, which enables temporal adaptation to new incoming examples on the basis of the data sampling real distribution over time, we have a built-in memory mechanism that is inherited from the (recent) past, and thus we can achieve an improvement in the classification performance that otherwise would be dependent on more examples and computational burden. We have compared DARK with benchmark solutions in the field, namely the Learn++.NSE algorithm, and experimental results revealed that DARK outperforms Learn++.NSE. The results were validated with the use of McNemar's test.

1.3 Thesis structure

This thesis has four parts, eight chapters and one appendix. The first part represents the fundamentals, the second part is about learning from crowds, in the third part the focus is on learning in dynamic environments, and finally the last one includes the conclusion and future work.

The current chapter introduces the research work done and the document itself.

In Chapter 2 the background in dynamic environments is presented. In particular we introduce the problem definition of learning in dynamic environments and the background in information extraction. In information extraction we will focus on document representation, preprocessing methods, learning methods, and performance metrics. Finally, a state-of-the-art is presented about learning in dynamic environments.

Chapter 3 presents crowdsourcing definition, perspectives, and characteristics. We also present relevant applications, existing platforms, and major challenges.

In Chapter 4 we introduce active learning and present the contributions using active learning and crowdsourcing to improve model's performance. We have developed different strategies in the field that are detailed in this chapter, like crowd customization.

Chapter 5 presents social networks' history and main features. Then, we introduce learning in social networks, detailing the current approaches used to learn in those scenarios. We also introduce social media as a crowdsourcing data source, presenting the contributions using crowdsourcing based on social networks to improve model's performance.

Chapter 6 describes the Drift Oriented Tool System (DOTS) framework we have settled. We firstly introduce the motivation and the framework setting, and then detail the three developed models, namely the time-window model, the incremental model and the ensemble model. Our main focus are memory mechanisms and for how long it is important to retain knowledge. The contributions made using the framework, along with real-world implementation scenarios, and the dataset used to evaluate and validate our strategies, are also described in this chapter.

Chapter 7 describes the Drift Adaptive Retain Knowledge (DARK) framework to tackle adaptive learning in dynamic environments based on recent and retained knowledge. DARK processes an ensemble of multiple Support Vector Machine (SVM) models that are dynamically weighted and have distinct training window sizes. A comparative study with benchmark solutions in the field, namely the Learn++.NSE algorithm, is also presented.

Finally, conclusions are presented in Chapter 8 along with future work.

Chapter 2

Dynamic Environments

In this chapter we focus on learning in dynamic environments, highlighting approaches to extract information in such drifting scenarios. We start by formally defining dynamic environments and their characteristics, namely in text-based settings that will constitute the basis of this work, laying a foundation of information and applications that can motivate further studies. Next, we briefly present preprocessing methods, state-of-the-art learning methods, and performance evaluation metrics. Finally, we discuss current approaches that try to tackle these learning challenges.

2.1 Introduction

The enormous growth of computational power in recent years, along with the popularization of social networks and mobile devices, have changed our society. New challenges have arisen as we created a deluge of data that needs to be acquired and handled as fast as it is being produced. Nonetheless, difficulties are not only related to the amount of data we need to cope with, but also with its dynamic nature, as data evolves faster and faster.

We have multiple examples in our society that reflect the effects of this information evolution, like the decrease of printed newspapers. During decades, even centuries, the printing process developed so we could wake up everyday with fresh news. However, it was unable to fulfil today's requirements as a few printing hours seem an eternity when we have mobile devices and powerful networks, that can put us live almost everywhere,

allowing us to read news in real-time from multiple sources, and even participate by sharing information worldwide through social networks.

Due to the dynamic nature of social networks, learning in these environments, known as dynamic or non-stationary, is not an easy task and requires special approaches, distinct from stationary counterparts. Considering this new reality, modern challenges in machine learning must include dynamic environments. A typical example is the pattern of customer's buying preferences that changes over time. For instance, in winter customers tend to buy warmer clothes, while in summer they prefer lighter ones. The preference pattern can change according to the temperature, which may depend on complex factors and can be infeasible to predict. Another example, related to the same problem, is detecting financial fraud based on the customers bought items, because, as customer's buying preferences evolve, it might be difficult to understand if a credit card payment is in line with previous payments, and thus normal, or part of a fraud. This type of problems, known as *concept drift* problems since Schlimmer & Granger (1986), will be detailed next.

2.2 Problem Definition

Dynamic environments can be defined as those where concept drift is present, so firstly one must understand what concept drift is. It refers to a variation of the underlying data distribution that defines the concept to be learned, for which the decision boundary is different from the previously seen examples. This means concept drift occurs when a set of examples has distinctive and legitimate class labels at two different moments (Kolter & Maloof, 2003; Muhlbaier & Polikar, 2007).

The core assumption, when dealing with a concept drift problem, is the uncertainty about the future (Žliobaitė, 2010), which means that concepts depend on some *hidden context*, not given explicitly in the form of predictive features, with the ability to induce more or less radical changes in the target concept (Widmer & Kubat, 1996).

Gama *et al.* (2014) proposes the following formal definition of concept drift between time point t_0 and time point t_1:

$$\exists X : p_{t_o}(X, y) \neq p_{t_1}(X, y) \tag{2.1}$$

where p_{t_o} denotes the joint distribution at time t_0 between the set of input variables X and the target variable y.

As stated above, in the presence of concept drift the learning task is harder and requires a specific approach, as incoming instances can not be treated as equally important contributors to the final concept (Tsymbal *et al.*, 2008).

Two kinds of concept drift may occur, though they are not always clearly separable: *real concept drift* and *virtual concept drift*. When hidden changes occur in context, causing a change of the target concept, we are in the presence of real concept drift. Widmer & Kubat (1993) also define real concept drift as hidden changes that reflect real changes in the world. Virtual concept drift is, on the contrary, defined by the same author as not occurring in reality but rather in a computer model that reflects the reality. This kind of effect can emerge when the representation language is poor and fails to identify all relevant features, or when the order of training examples for learning is skewed so that different types of instances are not evenly distributed over the training sequence (Widmer & Kubat, 1993; Gama *et al.*, 2014). Virtual concept drift can then be defined, in a nutshell, as the drift that occurs by a change in the underlying data distribution rather than in the concept.

From a practical point of view, it is not important which one of the two types occurs, since they can often occur together. Even if the target concept remains the same, and if there is only a slight variation of the underlying data distribution (virtual concept drift), this may lead to the necessity to change the current model, as the model's error may no longer be acceptable given the new data distribution. This means that the previously built models become useless and a new model must be built in order to cope with the new environment (Tsymbal, 2004). Real concept drift is also referred in literature as concept shift and virtual concept drift as sampling shift, like in Stavropoulos *et al.* (2016).

Concept drift is not always present in the whole instance space. In some problems, changes in the concept, or in the data distribution, occur just during a certain period of time, or only in some regions of the instance space. In the latter, when changes occur at instance level, rather than dataset level, we are in the presence of a *local concept drift*. Identifying local concept drift and being able to deal with it is an effort consuming challenge.

Moreover, concept drift occurs differently concerning the pattern of a change. Although there is not a consensual nomenclature in literature regarding the organization of drift types, we prefer the plain grouping approach proposed by Žliobaitė (2010): four types of concept drift were identified, namely *sudden* or *abrupt drift*, *gradual drift*, *incremental* or *stepwise drift* and *reoccurring drift*. Figure 2.1 graphically represents the four different concept drift types.

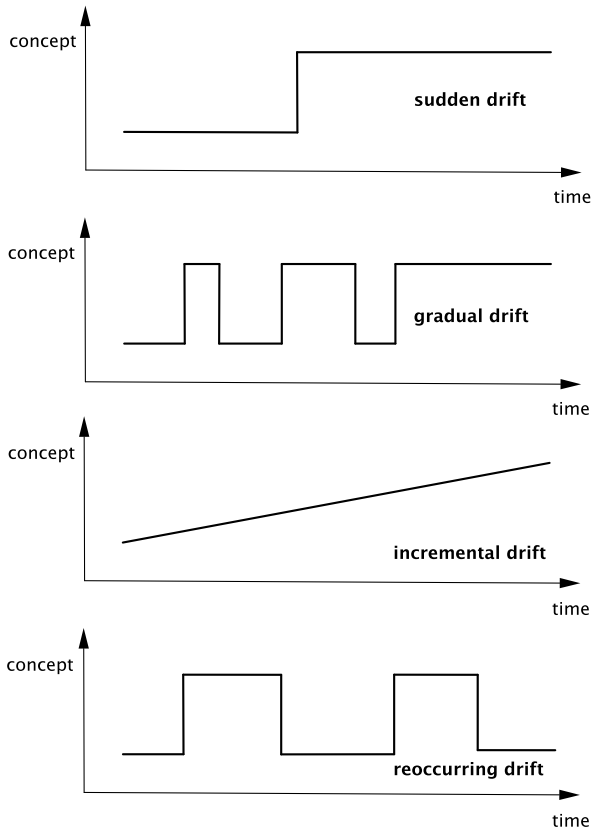

Figure 2.1: Graphical representation of concept drift types

Sudden drift occurs when the speed of the drift is high and a concept is abruptly substituted by another. Although in literature this kind of drift is usually stated as sudden or abrupt drift, it is also sometimes referred as *concept change*.

Gradual drift is another type of drift characterized by a slow speed drift. In this drift, the probability of a given context being associated with a concept decreases during a certain period of time, and proportionally, increases the probability of being

associated to another. In the beginning of a gradual drift, before more instances are seen, this drift may easily be confused with noise.

Incremental drift, also considered a subgroup of gradual drift or even just categorized as gradual drift without particular characteristics, can be seen as different because the change between the two concepts is very slow, being just perceived when looking to what is occurring during a larger period of time.

Reoccurring drift occurs when previously active concepts reappear after a certain period of time. It is noteworthy that the seasonality of the change must be previously unknown, otherwise the core assumption of the uncertainty about the future would be compromised (Žliobaitė, 2010).

Being able to deal with different types of concept drift in the same problem can constitute a relevant challenge and one must be aware that these settings are becoming widespread. Whether in social networks, where users change topics and concepts continuously, or in more professional scenarios, such as news agencies or patent agencies where trends are always evolving, there are multiple applications that would benefit from proper strategies to learn in such dynamic environments.

Nonetheless, a significant part of dynamic environments involves text-based inputs, and that will be the focus of this work. In the following we present the definition of information extraction, detailing document representation, preprocessing methods, learning methods, and performance metrics that will be used in the following sections.

2.3 Information Extraction

The deluge of information we are living in is already *status quo*. However, due to the amount of data that is being produced and propagated, it is today difficult to process it and identify significant information among all data. As an example, imagine a user seeking for a news confirmation. Firstly, the user must choose where to search for the information, for instance in a web search engine, in a news feed, or in a social network. Secondly, it is important to know how to search for the information in the chosen source, not only in order to narrow the search, but also to be understood. Finally, it is also vital to be able to decode the information, as it can be written in a foreign language, or in a digital format uneasy to decode.

There are multiple information extraction problems, like companies searching for information in order to better promote their products, intelligence agencies trying to detect terrorist threats, news agencies trying to perceive important events, or patent agencies trying to find duplicate patents, to mention just a few.

Four inherent characteristics of an information extraction problem were identified by Moens (2006): (*i*) the request for information, (*ii*) the answer for the request to be present in an unstructured data source, (*iii*) the impossibility for a human to process all the information of the data source and finally (*iv*) the inability for a computer to directly query the source of information.

Although these general characteristics are commonly accepted in an information extraction problem, the definition of information extraction varies in literature. Some authors, like Cowie & Lehnert (1996) and Piskorski & Yangarber (2013), rely on the assumption that the source of information is text-based. Others, like Cucchiara *et al.* (2003), also consider image or video as a possible source of information and use a wider definition. Considering the scope of this thesis, which is text-based, we prefer the definition proposed by Cowie & Lehnert (1996), that states that information extraction is any process which selectively structures and combines data which is found, explicitly stated or implied, in one or more texts.

Information extraction is a complex research field that aims to extract fragments of information in documents, like author, place, or date. Another approach, considering text problems, is text classification. In a text classification problem there usually is a previously known set of labels where each document can be categorized as belonging to.

Focusing on extracting information from texts, challenges are multiple. It is quite difficult to provide computers with the ability to understand texts, which requires decoding, i.e., perceiving the concepts beyond the words. When a document is written in natural language, an information extractor finds ambiguity as one of its major challenges. Ambiguity can be at word level, as some words have different meanings, like the word 'root', or at a syntactic level, called amphiboly or amphibology, as the same sentence can be reasonably interpreted in more than one way, like the well-known sentences 'free whales' and 'flying planes can be dangerous'.

Ambiguity may also be caused by a time change, as humans can associate different concepts to the same words during the course of time. For instance, in the sentence:

'the president of United States of America is going to give a press conference' we will immediately associate it with Donald Trump, which is the man in charge, but a few years ago we would associate the same sentence to Barack Obama. It is an important challenge to deal with this ambiguity.

Finally, another interesting challenge in information extraction is to extract more complex information from a document, for instance the underlying sentiments of a given text or sentence (Cambria, 2016; Das & Kalita, 2017), or the gender of the author (Karimi *et al.*, 2016; Wong *et al.*, 2016).

2.3.1 Document Representation

Natural language documents are syntactically structured, but the use of syntactic structures to represent a document is computationally hard. In text classification problems documents are mostly seen as unstructured data. To classify a document, we need to transform it into a suitable computational representation that can be used by a classifier. We present two different document representation schemes: *bag of words* and *natural language processing*.

Bag of Words

One of the most successful and commonly used document representation is the vector space model, also known as *bag of words*. Figure 2.2 depicts the document collection. The collection of features is built as the dictionary of unique terms present in the documents collections ($\mathcal{W} = w_1, w_2, w_3, ...w_z$). Each document of the document collection ($\mathcal{D} = d_1, d_2, d_3, ...d_z$) is indexed with the *bag* of the terms occurring in it, i.e., a vector with one element for each term occurring in the whole collection. Although a binary model represents the word w_i occurring in a document d_j as setting the element w_{ij} of the document vector to 1, one of the most common representations is using the term frequency value, i.e. not setting the w_{ij} to 1 to represent the presence of the word in the document, but to the number of times the term occurs in the document, which can be more informative. This representation is called *Term-Frequency* ($TF(w_{ij})$).

However, one of the major problems of this representation is that longer documents tend to have higher word count values, regardless of the actual importance of that term in the document. To avoid the bias of this representation in larger documents, $TF(w_{ij})$ is usually normalized by using the *Inverse Document Frequency* (IDF) as a

	w_1	w_2	w_3	...	w_z
d_1	w_{11}	w_{21}	w_{31}	...	w_{z1}
d_2	w_{12}	w_{22}	w_{32}	...	w_{z2}
d_3	w_{13}	w_{23}	w_{33}	...	w_{z3}
...	...	...	...	...	...
d_i	w_{1i}	w_{2i}	w_{3i}	...	w_{zi}

Figure 2.2: Document collection representation matrix

measure of whether the term is more or less common across the collection. To obtain the *IDF(*w_i*)*, we use the *Document Frequency* (*DF(*w_i*)*), that is the number of documents in which the word w_i occurs at least one time. *IDF(*w_i*)* can be calculated using the aforementioned *document frequency*, using

$$IDF(w_i) = \log(\frac{|\mathcal{D}|}{DF(w_i)}) \tag{2.2}$$

where $|\mathcal{D}|$ is the total number of documents.

Combining *term frequency* and *inverse document frequency* we have the *TF-IDF* representation, obtained by

$$TF - IDF(w_{ij}) = TF(w_{ij}) \times IDF(w_i) \tag{2.3}$$

Considering that the *inverse document frequency* of a word is low if it occurs in many documents, and high if the word occurs in only one, this word representation heuristic says that a word is an important indexing term for a document if it occurs frequently in it (the term frequency is high). On the counterpart, words that occur in many documents are rated less important indexing terms due to their low inverse document frequency(Joachims, 1997).

Natural Language Processing

Even though *bag of words* is a commonly used document representation scheme, Natural Language Processing (NLP) is also being widely used. One of the major drawbacks of *bag of words* is that the relation between terms in a document is not taken into account. For instance, the words 'United' and 'Nations' would be treated as separate

features, even though they together represent 'United Nations', a stronger and intuitive feature. Another motivation for using NLP is that it considers the importance of a word token as depending on both its type and the specific linguistic context in which it appears (Paola Merlo & Wehrli, 2013).

Natural Language Processing is a vast field of research with multidisciplinary tasks. Nevertheless, a lot of effort is being put in part-of-speech (POS) tagging, Named Entity Recognition (NER), and Semantic Role Labelling (SRL). POS aims at labelling each word with a unique tag that identifies its syntactic role, like noun, verb, or adverb, while NER labels atomic elements of the phrase into categories, such as 'brand', 'location' or 'person' (Collobert & Weston, 2008). SRL aims at identifying all the components in the sentence which fill a semantic role, like 'agent', 'patient' or 'instrument', but also adjuncts such as 'cause', 'manner' or 'locative' (Carreras & Màrquez, 2005).

2.3.2 Preprocessing Methods

In text classification problems the aim is to induce a hypothesis using an algorithm that can predict the label of new examples as accurately as possible. If a vector with one element for each occurring term in the whole collection is used to represent a document, there can be a high dimensional feature space that can bring not only computational problems, but also the over-fitting of data, which can inhibit the classifier to generalize and thus predict the classification of unseen data. As a consequence, preprocessing methods are often applied. They are also referred as dimensionality reduction methods since their goal is to reduce the size of the document representation, controlling the computational burden involved, whilst maintaining or improving the classification performance, as they prevent the mislead in classification.

Besides their importance, it is not an easy task, as potentially useful information can then be disregarded. Preprocessing methods can then be divided in two different types according to: *feature selection* and *feature extraction*.

Feature Selection

Feature selection aims to reduce the document representation by identifying a smaller set of terms that could represent the document more effectively. Some learning algorithms are natively capable of feature selection, like decisions trees, and thus would not need prior feature selection methods to be applied. However, feature selection is

mandatory for others, as a large number of features can not only represent a computational burden but also impair the classification accuracy. Feature selection methods can be divided into *filter methods* and *wrapper methods*.

- **Filter Methods**

  One of the most successful and commonly used document representation is the vector space model, where a feature is defined as a word occurring in a document. Filter methods are simple methods. Some scoring measure is defined so the relative importance of a term can be represented. The score is assigned to each feature independently, then sorted according to the assigned score, and finally a predefined number of the best features is taken to form the solution feature subset (Mladenić, 1998).

  Filter methods consider attributes independently from the algorithm that will use them, relying on general characteristics of the training set to select some features and discard others (Langley, 1994).

  Stopword removal can be considered a basic filter method and rely on the assumption that some words, such as articles, prepositions, and conjunctions, called *stopwords*, are non-informative words, and occur more frequently than informative ones. Those words are then included in a list and filtered in the document representation process, as they could mislead correlations between documents.

  Despite the advantages of being simple and independent of the classifier used, the above mentioned methods also have the disadvantage of totally ignoring the effect of the selected feature subset on the performance of the classifier (John *et al.*, 1994).

- **Wrapper Methods**

  Taking into account the disadvantage of the filter approach, wrapper methods conduct a search for a viable subset taking into account the classifier, as it is used as part of the evaluation process. The general argument for wrapper approaches is that the classifier that will use the feature subset should provide a better estimate of accuracy than a separate measure that may have an entirely different inductive bias (Langley, 1994).

In wrapper methods, a candidate subset is generated and tested against the classifier. The performance obtained by the candidate subset is then used to score that subset among others, and a new subset is generated usually by adding or removing attributes. The process is then continued until some criteria is reached and best fit are chosen.

Although the wrapper approach may obtain better performances, their major disadvantage over filter methods is the former's computational cost, which results from calling the classifier for each feature set considered (Langley, 1994; Sánchez-Maroño *et al.*, 2007).

Feature Extraction

Feature extraction methods propose to generate a new set of terms to represent a document that, differently from the original ones, could be more robust to problems like polysemy, homonymy, and synonymy (Sebastiani, 2002). The rationale for feature extraction is that users in different contexts, or with different needs, knowledge, or linguistic habits will describe the same information using different terms. Indeed, the degree of variability in descriptive term usage is much greater than is commonly suspected (Deerwester *et al.*, 1990).

We will present three different feature extraction methods: *term clustering*, *Principal Components Analysis* and *Latent Semantic Indexing*.

- **Term Clustering**

  Term clustering is based on the idea that multiple words can be so semantically related that they could be grouped together and be represented as a unique, unified, term.

  Stemming is an example of a term clustering method. It consists in removing case and inflection information of a word, reducing it to the word stem. As an example, words like *preprocessing*, *processing* or *processed*, would be all contracted to the same stem, in this case the word *process*. Stemming does not alter significantly the information included, but it does avoid feature expansion. In the above mentioned example, 3 features would be reduced to only 1. By unifying those terms, it also improves their importance, as the word *process* appears 3 times more than each word individually.

There are multiple stemming algorithms for the English language presented in literature and freely available, like those proposed by Porter (1980) and Krovetz (1993). A survey of stemming algorithms in information retrieval is presented in Moral *et al.* (2014), including a compilation of stemming algorithms for non-English languages.

One of the major drawbacks of term clustering is not perceiving the relation between words that are not explicitly semantic related, like 'car' and 'automobile'. If there is a great number of terms which all contribute a small amount of critical information, and are not semantically explicitly related, a term-based solution is unable to identify this evidence (Deerwester *et al.*, 1990).

- **Principal Components Analysis**

  Principal Components Analysis (PCA) is a statistical technique that aims to reduce the dimensionality of data by transforming the original attribute space into a smaller space, deriving new variables that are combinations of the original ones and are uncorrelated (Zareapoor & Seeja, 2015).

  PCA is domain independent and not specifically designed for textual data, even though it is widely used in text-based problems for feature reduction. The advantage of PCA is mainly related to its ability to describe significant information/variation within the recorded data typically by the first few score variables, which simplifies data analysis tasks accordingly (Kruger *et al.*, 2008). However, PCA has also limitations, as it assumes that the relationships between variables are linear (Linting *et al.*, 2007).

- **Latent Semantic Indexing**

  Latent Semantic Indexing (LSI) exploits the existence of some underlying latent semantic structure in the association of terms and documents. It was proposed by Deerwester *et al.* (1990) and it is described as a statistical technique to estimate this latent structure, avoiding 'noise'. It takes a large matrix of term-document association data to construct a 'semantic' space wherein terms and documents that are closely associated can be represented together. A singular-value decomposition is executed to arrange the space to reflect the major associative patterns

in data, ignoring those smaller or less important. The positions in this space are used as the semantic indexing.

One characteristic of LSI, and that is common to PCA, is that the newly obtained dimensions are not, unlike in term selection and term clustering, intuitively interpretable. However, they work well in bringing out the 'latent' semantic structure of the vocabulary used in the corpus and, on the contrary of term-based solutions, consider the relation shown in the scenario where a great number of terms contribute a small amount of critical information (Sebastiani, 2002; Schütze *et al.*, 1995).

2.3.3 Learning Methods

In this section we present a brief overview of learning methods for text classification, namely Rocchio's method, decision trees and decision rules, Naïve Bayes, k-nearest neighbor, support vector machines, and also active learning methods.

Rocchio's Methods

Rocchio's method (Rocchio, 1971) is a classic method for handling text documents originally inspired in information retrieval. It can be used to obtain linear, profile-style classifiers, i.e., linear classifiers that consist of an explicit profile (or prototypical document) of the class, which brings interpretability advantages.

In Rocchio's method, each word w_k is assigned a weight ω_k, that combines the term weight in the original query q with the weights relevant (Pos) and irrelevant (Neg) documents for the specific class:

$$\omega_k = \lambda\,\omega_{qk} + \beta \sum_{i \in Pos} \frac{\omega_{ik}}{|Pos|} - \gamma \sum_{i \in Neg} \frac{\omega_{ik}}{|Neg|}, \tag{2.4}$$

where ω_{ik} is the weight or representation of term w_k in document $\mathbf{d}_i$, Pos are the training documents belonging to the class, Neg are the training documents not belonging to the class and $|.|$ represents the number of elements of the set.

This method relies on an adaptation to text classification of the well-known Rocchio's formula for relevance feedback in the vector space model, and it can be used for text classification (Hull, 1994; Sebastiani, 2002; Joachims, 1997; Schapire *et al.*, 1998),

as baseline classifier (Cohen & Singer, 1999; Joachims, 1998; Lewis *et al.*, 1996; Schapire & Singer, 2000), or member of a classifier committee (Larkey & Croft, 1996).

As in text classification we do not have an initial query, the term weight in the original query, ω_{qk}, is not considered, by setting $\lambda = 0$. β and γ are control parameters that allow setting the relative importance of positive and negative examples.

Decision Trees

Decision trees and decision rules are usually chosen due to being easily interpreted by humans. Such symbolic or non-numeric methods include internal nodes that correspond to labels or words in text classification approaches. Besides the internal nodes, decision trees also include branches departing from each node, that are labelled by tests on the weight that the term has in the testing document.

Each leaf represents the categories according to the branches followed. To classify a testing document $\mathbf{d}_i$, starting from the root of the decision tree, each term weight in the document is tested to determine which branch to follow, until a leaf node is reached. The class label of this leaf node is then assigned to the document $\mathbf{d}_i$ (Silva & Ribeiro, 2010).

Decision trees and decision rules are similar as both can encode any boolean function. A decision rule classifier built by an inductive rule learning method consists of a Disjunctive Normal Form (DNF) rule. The literals in the premise denote the presence (non-negated keyword) or absence (negated keyword) of the keyword in the testing document $\mathbf{d}_i$, while the clause head denotes the decision to classify $\mathbf{d}_i$ under the class (Silva & Ribeiro, 2010).

Naïve Bayes

The Naïve Bayes classifier is a probabilistic classifier based on the Bayes' theorem. The Bayes' theorem, stated mathematically in 2.5, defines the probability of a document $\mathbf{d}_i = (w_{i1}, w_{i2}, \ldots, w_{ik}, \ldots, w_{iN})$ to be classified as a document of class c_j, $P(c_j|\mathbf{d}_i)$.

$$P(c_j|\mathbf{d}_i) = \frac{P(c_j) \cdot P(\mathbf{d}_i|c_j)}{P(\mathbf{d}_i)} \tag{2.5}$$

The estimation of $P(\mathbf{d}_i|c_j)$ and $P(\mathbf{d}_i)$ is usually complex, given that the number of possible document vectors $\mathbf{d}_i$ is very large. However, such value, $\mathbf{d}_i$, does not vary

between classes and can thus be neglected, since it can be interpreted only as a scale factor. To compute $P(\mathbf{d}_i|c_j)$, this classifier makes the naïve assumption that gives the name to the method, i.e., that any two features of the document vector are, when viewed as random variables, statistically independent of each other (Sebastiani, 2002), or, mathematically:

$$P(\mathbf{d}_i|c_j) = \prod_{k=1}^{|\mathcal{W}|} P(w_{ik}|c_j). \tag{2.6}$$

Despite the fact that the naïve assumption of independence is generally not true for word or features appearing in documents, the Naïve Bayes classifier is surprisingly effective (Silva & Ribeiro, 2010).

K-Nearest Neighbor

The k-nearest neighbor is one of the most emblematic instance-based methods (Mitchell, 1997). Instance-based methods are characterized by not constructing an explicit model with the training examples, i.e., they do not use the training phase to delineate boundaries between classes. On the contrary, actions are only taken in the testing phase, when presented to a new instance, which they compare to the previously seen examples. As a consequence, they are usually called *lazy classifiers*.

The K-nearest neighbor method assumes that all instances correspond to points in an n-dimensional space $\mathbb{R}^n$. To classify a given instance, the training examples are represented as vectors in the multidimensional feature space and a class label c is associated with each vector. In the simplest case, it will be either a positive class or a negative class. Then, when a new instance is presented to the classifier, the distance to the training examples is computed. Usually the distance between two instances x_i and x_j is defined to be the Euclidean distance $d(x_i, x_j)$, where $a_r(x)$ denotes the value of the rth attribute A_r of the instance x:

$$d(x_i, x_j) \equiv \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2} \tag{2.7}$$

Finally, a point in the space is assigned to the class c if it is the most frequent class label among the k nearest training samples, being k a user-defined constant.

Although considered a simple classification method, k-nearest neighbor has been successfully applied to a large number of computational tasks, such as text classification, pattern recognition, or intrusion detection.

Support Vector Machines

Support Vector Machines (SVM) is a machine learning method introduced by Vapnik (2000), based on his Statistical Learning Theory and Structural Risk Minimization Principle. The underlying idea behind the use of SVM for classification consists in finding the optimal separating hyperplane between the positive and negative examples. The optimal hyperplane is defined as the one giving the maximum margin between the training examples that are closest to it. Support vectors are the examples that lie closest to the separating hyperplane. Once this hyperplane is found, new examples can be classified simply by determining on which side of the hyperplane they are.

The output of a linear SVM is $u = \mathbf{w} \times \mathbf{x} - b$, where $\mathbf{w}$ is the normal weight vector to the hyperplane and $\mathbf{x}$ is the input vector. Maximizing the margin can be seen as an optimization problem:

$$\begin{aligned} &subjected\ \ to\ \ y_i(\mathbf{w}.\mathbf{x} + b) \geq 1, \forall i, \\ &minimize \quad \frac{1}{2}||\mathbf{w}||^2, \end{aligned} \tag{2.8}$$

where $\mathbf{x}$ is the training example and y_i is the correct output for the ith training example. Intuitively the classifier with the largest margin will give low expected risk, and hence better generalization.

To deal with the constrained optimization problem in (2.8) Lagrange multipliers $\alpha_i \geq 0$ and the Lagrangian (2.9) can be introduced:

$$L_p \equiv \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{l} \alpha_i (y_i(\mathbf{w}.\mathbf{x} + b) - 1). \tag{2.9}$$

In fact, Support Vector Machine (SVM) constitute currently the best of breed kernel-based technique, exhibiting state-of-the-art performance in diverse application areas, such as text classification (Joachims, 2002; Tong & Koller, 2002; Antunes *et al.*, 2011).

Active Learning

The key idea behind active learning is that a machine learning algorithm can achieve greater accuracy with fewer training labels if it is allowed to choose the data from which it learns. An active learner may pose queries, usually in the form of unlabelled data instances, to be labelled by a supervisor (Settles, 2010).

The reason for using active learning is mainly to expedite the learning process and reduce the labelling efforts required by the supervisor, therefore active learning is well-motivated in many modern machine learning problems where data may be abundant but labels are scarce or expensive to obtain (Baram *et al.*, 2003; Settles, 2010). Another strong reason is the possibility of each user defining personal labels, thus constructing a customized learning model that better fits his preferences. The customization of a learning model is particularly important in recommendation applications, like movie, or book recommendation systems.

Active learning methods can be grouped according to the selection strategy, as being committee-based and certainty-based (Silva & Ribeiro, 2007). In the first group the active examples combine the outputs of a set of committee members, by determining those in which the members disagree the most as the candidates to be labelled (McCallum & Nigam, 1998). The certainty-based methods try to determine the most uncertain examples and point them as active examples to be labelled. The certainty measure depends on the learning method used.

Active learning has been successfully applied to a large number of computational tasks, like image recognition, word disambiguation, or text classification.

Other Methods

We have made an effort to present an overview of learning approaches in information extraction literature. However, being aware that others could also be mentioned, we have considered that extending this survey to a more complete one would compromise the outline of this thesis. Nevertheless, the ones most worth mentioning include bio-inspired approaches, e.g., Genetic Algorithms (Uğuz, 2011), Particle Swarm Optimization (Wang *et al.*, 2007), and Artificial Immune Systems (Antunes *et al.*, 2011), and also fuzzy approaches (Ayeldeen *et al.*, 2013). Ensemble models are also worth

mentioning but, considering their suitableness to dynamic environments, they will be further described in the Section 2.4.2.

2.3.4 Evaluation

To evaluate the performance of a classifier, evaluation strategies must be defined. One of the most common evaluation strategy is to calculate the accuracy of a classifier, i.e., what is the portion of documents that are correctly assigned to their class. The simplest way of doing this is to split the labelled portion of the dataset into training data and test data. Training data is used in model construction, while test data is used to evaluate the performance of the model. The predictions provided by the model in the test data are then compared with the correct labels and thus the accuracy of the classifier is calculated. Some benchmarks have a predefined ratio between training and test sets, however, when not defined, the typical ratio is 70:30. When validation mechanisms are used, a small amount of documents is put aside and called validation data. The validation process occurs between training and testing with the aim of tuning the classifier before it is used for testing purposes. The usual ratio in these cases is 60:20:20.

Another approach that tends to compensate the rigidity of splits is *cross validation*, where experiments are often repeated with different random splits into training, validation, and testing datasets. Usually the dataset is randomly split and evaluated from 5 to 30 times, and the mean and variance of the criteria are reported. Techniques like cross validation and k-fold validation help guard against randomness in particular data splits, and allow for sounder results. The k-fold validation involves splitting the data in k parts (folds), using $(k-1)$ parts for training and the remaining part for testing. This is repeated k times, considering all possible testing sets, one at a time. There are multiple advantages, like considering that every example from the original dataset has the same chance of appearing in the training and testing set, or the ability to perform validations when data is scarce. For instance, if we reserve data for model construction, but we lack a sufficient portion for testing purposes, or, on the counterpart, we can reserve data for model testing, but we might lack relevant data for model construction. Despite these potential advantages, cross-validation is usually not appropriate for problems with time dynamics, since in those cases the test examples should always be

pooled from latter examples than training examples, which is not guaranteed in cross validation.

Focusing on text classification, we can simplify it into a binary decision problem, as each document can be classified as being in a given class, or not.

In order to evaluate the binary decision task, a contingency matrix can be defined to represent the possible outcomes of the classification, as shown in Table 2.1.

	Class Positive	Class Negative
Assigned Positive	a	b
	(True Positives)	(False Positives)
Assigned Negative	c	d
	(False Negatives)	(True Negatives)

Table 2.1: Contingency table for binary classification

Traditional measures can be defined based on this contingency table, such as error rate ($\frac{b+c}{a+b+c+d}$) and accuracy ($\frac{a+d}{a+b+c+d}$). However, for unbalanced problems, i.e., problems where the number of positive examples is rather different among classes, or in the case of a binary problem, the number of positive examples is rather different from the negative examples, as it often happens in text classification, more specific measures should be defined to capture the performance of each model. Such measures usually take into consideration not just the error, but distinguish between positive and negative errors. Typical examples include recall ($R = \frac{a}{a+c}$) and precision ($P = \frac{a}{a+b}$). Although they can be seen as complementary, recall emphasizes the false negatives, while precision is sensitive to false positives.

Additionally, combined measures that give a more holistic view of performance in just one value, yet considering false negatives and false positives as separate problems, are very helpful. Paradigmatic to such measures is the van Rijsbergen F_β measure (van Rijsbergen, 1979), which combines recall and precision in a single score and thus is one of the best suited measures for text classification:

$$F_\beta = \frac{(\beta^2 + 1)P \times R}{\beta^2 P + R}. \tag{2.10}$$

F_β can comprise values from F_0 to F_∞. F_0 is the same as precision, while F_∞ is the same as recall. The most common value assigned to β is $\beta = 1$, i.e. F_1, an harmonic

average between precision and recall (2.11).

$$F_1 = \frac{2 \times P \times R}{P + R}. \tag{2.11}$$

Identical to F_β is F_α that only changes the way to control the break-even between precision and recall (2.12), e.g., the harmonic average is achieved with $\alpha = 0.5$.

$$F_\alpha = \frac{1}{\alpha \frac{1}{P} + (1 - \alpha) \frac{1}{R}} \tag{2.12}$$

In a multi-class problem, a binary decision setting can be applied by using a one-against-all approach.

Considering the time series and the aforementioned one-against-all strategy, a classifier can be trained for each batch of the time series that is composed by $|Y|$ binary classifiers, being Y the collection of possible labels. To perceive the performance of the classification for each class, all the binary classifiers that were created in all the time series batches must be considered. To evaluate the performance obtained across time, it is possible to average the obtained results. Two conventional methods are widely used, specially in multi-label scenarios, namely macro-averaging and micro-averaging. Macro-averaged performance scores are obtained by computing the scores for each learning model in each batch of the time series and then averaging these scores to obtain the global means. Differently, micro-averaged performance scores are computed by summing all the previously introduced contingency matrix values (a,b,c and d), and then use the sum of these values to compute a single micro-averaged performance score that represents the global score.

In the next section, we will discuss current approaches that try to tackle learning in dynamic environments.

2.4 Learning in Dynamic Environments

Effective learning in non-stationary environment with hidden contexts and concept drift requires learning algorithms with the ability to detect context changes without being explicitly informed about them, quickly recover from the context change and adjust its hypothesis to the new context. Such algorithms should also make use of previous

experienced situations when old context and corresponding concept reappear (Widmer & Kubat, 1996).

One of the major challenges that is posed to a learning model in non-stationary environments is to distinguish between an effective concept drift and slight variations that are due to noise in the training data. If a learning algorithm is designed to quickly react to the first signs of concept drift, it can be easily mislead into overreacting to noise and erroneously interpreting it as concept drift. On the other hand, if it is designed to be highly robust in the face of noise, it will probably take too long to react to real changes, turning it less sensitive than what is expected to learn under drift conditions (Widmer & Kubat, 1996; Tsymbal, 2004). This problem is known as the *stability-plasticity dilemma* (Grossberg, 1988). The underpinning idea of the dilemma is that a learning model to operate in a dynamic environment requires plasticity for coping with new knowledge, but also stability to prevent forgetting the previously acquired. If a model has too much plasticity it will adapt easily to new knowledge but, as a counterpart, it will lack stability, and thus it will also easily forget. For algorithms to cope with drift, they must converge quickly and accurately to new target concepts, while being efficient in time and space (Kolter & Maloof, 2003).

Different approaches exist for learning in non-stationary environments that can be casted as active or passive approaches. Table 2.2 describes and summarizes both approaches that will be further described.

2.4.1 Active Approaches

Active approaches for learning in dynamic environments are used to detect changes in the environment and react adaptively, updating or building a new classifier. Features are extracted for change detection and, once a change is detected, the classifier model is updated or rebuilt by discarding the obsolete knowledge and adapting to the new environment. The whole process involves change detection and adaptation methods (see Table 2.2)(Ditzler *et al.*, 2015).

Change detection approaches inspect extracted features and variations in the underlying distribution data using theoretically-grounded statistical techniques and include (Ditzler *et al.*, 2015):

Category	Type	Approaches	References
Active	Change Detection	Based on theoretically-grounded statistical techniques: hypothesis tests, change-point methods, sequential hypothesis tests, and change detection tests.	(Patist, 2007; Nishida & Yamauchi, 2007; Ross *et al.*, 2011; Wald, 1992; Armitage, 1960; Harel *et al.*, 2014; Haque *et al.*, 2015)
	Adaptation	Adapting the classifier by learning from the newly available information and discarding the obsolete one. Main mechanisms: windowing, weighting and random sampling.	(Alippi *et al.*, 2013, 2012; Bifet & Gavalda, 2007; Alippi & Roveri, 2008; Cohen *et al.*, 2008b; Koychev, 2000; Datar & Motwani, 2016; Alippi *et al.*, 2009; Klinkenberg, 2004; Vitter, 1985)
Passive	Single Models	Applied in big data scenarios. Paradigmatic models include decision trees (DT), very fast decision trees (VFDT). Other approaches include extreme learning machine (ELM).	(Domingos & Hulton, 2000; Hulten *et al.*, 2001; Liu *et al.*, 2009; Cohen *et al.*, 2008a; Ye *et al.*, 2013)
	Ensemble Models	Extremely appropriate for learning in dynamic environments. Reduction in the variance of the error and attain the flexibility. Approaches include boosting, bagging or random forests.	(Freund & Schapire, 1997; Breiman, 1996, 2001; Bagul & Phulpagar, 2016; Ditzler & Polikar, 2013; Tabassum & Ahmed, 2016; Ren *et al.*, 2016; Jr., 2011; Elwell & Polikar, 2011; Karnick *et al.*, 2008)

Table 2.2: Summary of comparison between active and passive approaches

- *Hypothesis Tests* assess the validity of a hypothesis by controlling the false positive rate in change detection based on predetermined confidence calculations and using statistical techniques. The confidence threshold can be based on the mean value with which a set of samples has been drawn from a specific distribution as in Patist (2007) and Nishida & Yamauchi (2007);

- *Change-Point Methods* use a fixed data sequence to verify if a given sequence contains a change-point, by analysing all possible partitions of the data stream. This statistical technique is highly computationally bound, nevertheless it has the ability to detect the presence of a change and estimate the instant when the change occurred, as in Ross *et al.* (2011);

- *Sequential Hypothesis Tests* inspect sequentially incoming examples, one at a time, until there are enough examples to determine the presence of a change or not. Some examples of this technique are the probability ratio test (Wald, 1992) and the repeated significance test (Armitage, 1960);

- *Change Detection Tests* overcome limitations of the previous technique by sequentially analysing the statistical behaviour of data streams. This method consists in a change detection based on a threshold as in Harel *et al.* (2014) and Haque *et al.* (2015). The limitation of this method is the difficulty to set the threshold to an optimal value with which we may have a reasonable classification performance.

The **Adaptation** phase occurs after a change in the environment is observed and detected. It consists in adapting the classifier to the change, by learning from the new available information and discarding the obsolete one (Gama *et al.*, 2014). Adaptation mechanisms can be grouped into the following three main categories (Ditzler *et al.*, 2015):

- *Windowing* is the most used and easiest mechanism. It is based on a sliding-window that includes, at each given moment, the most recent and up-to-date examples, while the obsolete examples are discarded. With this mechanism the up-to-date examples are used to retrain the classifier and thus enhance its performance for the next batch(es). The choice of the appropriate window length is a critical issue and can itself be adaptively calculated (Bifet & Gavalda, 2007; Alippi *et al.*, 2012, 2013) or determined by the expected change ratio (Alippi

& Roveri, 2008; Cohen *et al.*, 2008b). Just-In-Time (JIT) adaptive classifier, a new generation of adaptive classifiers that are able to operate in non-stationary environments is proposed in Alippi & Roveri (2008).

- *Weighting*, unlike windowing mechanisms, takes into account all the examples weighted according to some rule, like their age or relevancy with respect to the recent classification accuracy performance (Koychev, 2000). Several approaches can be found in literature regarding the weighting mechanisms used: gradual forgetting (Koychev, 2000), time-based weighting (Datar & Motwani, 2016), change index, which measures the variation of data processing over time (Alippi *et al.*, 2009), and based on the accuracy/error calculated in the last batch of supervised data (Klinkenberg, 2004);

- *Sampling*, more precisely reservoir sampling (Vitter, 1985), uses randomization and is able to select a subset of unique examples from the data stream.

Active approaches in dynamic environments, like those related with change detection in temporal data streams, can be easily observed in some real-world applications, like network intrusion (Kim *et al.*, 2007a; Haque & Alkharobi, 2015) and spam detection (Ahsan *et al.*, 2016; Lughofer & Mouchaweh, 2015).

2.4.2 Passive Approaches

Passive approaches, unlike active approaches, do not aim at detecting the presence of changes or drifts in the environment, but take a more natural-inspired path of continuous adaptation of the model parameters every time new data arrives (Ditzler *et al.*, 2015). The complexity of such adaptation methods varies, but the main goal is to keep the final model as close to the state of reality brought out by current data. Passive approaches can be divided into single models and ensemble models:

Single models are constituted by only one model, presenting a lower computational burden that is often appropriate for massive data streams. As a consequence, less complex models, e.g. decision trees (DT), can be used. In fact, decision trees are the mostly common classifiers used for data stream mining with the very-fast decision tree (VFDT) learner being one of the most popular (Domingos & Hulton, 2000). A

sliding-window approach was also proposed to take different options into consideration at each tree node split (Hulten *et al.*, 2001; Liu *et al.*, 2009).

Other examples of single models include a fuzzy-logic-based approach that also exploits a sliding-window over the training data stream (Cohen *et al.*, 2008a) and an online extreme learning machine (ELM) combined with a time-varying neural network for learning from non-stationary data (Ye *et al.*, 2013).

Ensemble models are one of the best researched methods for adaptive learning in dynamic environments. Ensembles of classifiers integrate multiple classifiers and are based on the idea that, given a task that requires expert knowledge, k experts (baseline classifiers) may perform better than one, if their individual judgements are appropriately combined. The underlying idea of using an ensemble of classifiers is based on the human nature of seeking more than one opinion when a major decision comes along. Searching for multiple opinions increases confidence and allows a more informed answer to the facing challenge (Polikar, 2006). A classifier committee is then characterized by (i) a choice of k classifiers, and (ii) a choice of a combination function, sometimes denominated a *voting algorithm*. The classifiers should be as independent as possible to guarantee a large number of inductions on the data. By using different classifiers, we can exploit diverse patterns of errors, making the ensemble better than just the sum (or average) of the parts.

The simplest combination function is just a majority voting mechanism with an odd number of baseline classifiers. However, more advanced strategies have been pursued. In Kuncheva (2002), a theoretical study on six classifier fusion strategies is presented. The authors analyse the classification error for different fusion methods: average, minimum, maximum, median, majority vote, and oracle.

There are many approaches for ensemble of classifiers, such as *boosting* (Freund & Schapire, 1997), *bagging* (Breiman, 1996), or *random forests* (Breiman, 2001), but their original form is usually applied to static environments. Nevertheless, ensembles are specially adequate to tackle dynamic evolving settings, given their modular nature. Ensembles are cutting-edge solutions to many different learning challenges and different researchers have been studying ensembles and their applications in various fields (Tabassum & Ahmed, 2016; Ditzler & Polikar, 2013; Ren *et al.*, 2016; Jr.,

2011; Bagul & Phulpagar, 2016). In Karnick *et al.* (2008) an approach of incremental learning of concept drift in non-stationary environments is presented. The authors describe ensemble-based approaches of classifiers for incrementally learning from new data, drawn from a distribution that changes in time, and generate a new classifier using each additional dataset that becomes available from the changing environment. The classifiers are then combined by a modified weighted majority voting, where the weights are dynamically updated based on the classifiers' current and past performances, as well as their age. This will permit to track changes and respond accordingly, even in a cyclical environment. They combine a base classifier with Learn++, proposed in Polikar *et al.* (2001). Experiments were done using three base classifiers, namely, multilayer perceptron (MLP), SVM, and Naïve Bayes with promising results in dealing with concept drift. Another popular batch-based learning algorithm for non-stationary environments is Learn++.NSE (NSE stands for Non-Stationary Environments) (Polikar *et al.*, 2001) that will be detailed further in this thesis.

2.5 Conclusion

In this chapter we have introduced the background in dynamic environments. We started with a formal definition and carried on by detailing their characteristics, highlighting information extraction problems in such drifting scenarios. Considering information extraction, we have presented fundamentals on document representation, preprocessing methods, learning methods and performance metrics. Finally, we have discussed current learning approaches in dynamic environments.

Dynamic environments are extremely challenging and are included in cutting edge research topics. The methods we have detailed are classified as active and passive approaches. Up-to-date references were also provided.

In the dynamic text-based settings that we will tackle in this work, concepts vary significantly with time. Learning strategies to cope with such dynamics often include passive approaches that adapt to new concepts, not necessarily detecting explicit drifts, which could be unattainable given the pace of change. Ensemble learning is usually implemented by building several models that are retained, aiming at obtaining multiple expert views of the underlying data distribution through time.

Among the most significant examples of text-based dynamic environments are social networks, like Twitter or Facebook. Considering their dynamic nature and their potential in information spread, it becomes imperative to find learning strategies able to cope with these environments, since they are an unquestionable and valuable information source, as will be explored in the next chapter, by introducing the concept of *crowdsourcing*.

Part II

Learning from Crowds

Chapter 3

Crowdsourcing

In this chapter we introduce the *crowdsourcing* concept, from the definition of crowd to different perspectives that have been proposed. We also detail its characteristics and main applications. Crowd-based platforms are also explored and specific examples are described. Finally, further challenges are discussed.

3.1 Introduction

Over the last few years, with the burst of communication technologies, people are becoming easily connected and can thus communicate, share, and group together rapidly without effort, giving room for virtual communities to emerge. Considering this new reality, industries and organizations discovered that crowdsourcing, an innovative low-cost work force, could save time (and money) in problem solving.

The underpinning idea behind crowdsourcing is that, under the right circumstances, groups can be remarkably intelligent and efficient. Although the concept is fairly new, as the term was coined in 2006, it is now familiar, making often part of our daily routine. Examples are multiple: we are used to be asked to give opinions about a product or a brand in social networks or in TV shows, or even to contribute to multiple crowdfunding projects.

In machine learning, besides the intelligent use of a group as an ensemble, there is another potential noteworthy advantage of using crowdsourcing, as there are tasks that are notoriously difficult for an algorithm to perform and quite simple for humans,

like speech or image recognition, language understanding, text summarization, and labelling (Barr & Cabrera, 2006).

Hence, crowdsourcing has been widely studied for the last few years, being the focus of research in many fields, like economics, biology, social sciences and engineering, among others (Leimeister, 2010).

3.2 Definition

Since the seminal work of Surowiecki (Surowiecki, 2004), the concept of crowdsourcing is expanding, mainly through the work of Jeff Howe (Howe, 2006), where the term crowdsourcing was definitely coined. The main idea behind crowdsourcing is that groups can be remarkably intelligent. They do not need to be dominated by exceptionally intelligent individuals in order to be smart or to fulfil a purpose. On the contrary, they are often smarter than the smartest individual in the group, i.e., group decisions are usually better than the decisions of the brightest party.

According to Estellés-Arolas & González-Ladrón-De-Guevara (2012), there are more than 40 original definitions of crowdsourcing in at least 209 scientific documents, like books, journal articles, conference papers, and others. As there are multiple definitions, they do not all fully agree, and thus a definition is proposed trying to cover the most relevant aspects of crowdsourcing. The definition is as follows. *Crowdsourcing is a type of participative online activity in which an individual, an institution, a non-profit organization, or a company, proposes to a group of individuals of varying knowledge, heterogeneity, and number, via a flexible open call, the voluntary undertaking of a task. The undertaking of the task, of variable complexity and modularity, and in which the crowd should participate bringing their work, money, knowledge and/or experience, always entails mutual benefit. The user will receive the satisfaction of a given type of need, be it economic, social recognition, self-esteem, or the development of individual skills, while the crowdsourcer will obtain and utilize to their advantage that what the user has brought to the venture, whose form will depend on the type of activity undertaken (Estellés-Arolas & González-Ladrón-De-Guevara, 2012).*

Another important and well accepted definition of crowdsourcing is proposed by Daren Brabham (Brabham, 2013), and consists in *an online, distributed problem-solving and production model that leverages the collective intelligence of online communities to*

serve specific organizational goals. Online communities, a.k.a. crowds, are given the opportunity to respond to crowdsourcing activities promoted by the organization, and are motivated to respond for a variety of reasons (Brabham, 2013).

The conscience of the potential of the crowd results in new economical concepts, like *crowdserving* and *crowdfunding*. Crowdserving is based on the idea of "everyone as a service" (Petrie, 2010), which means that every connected individual can be a service provider by individually promoting the services s/he can manage. This idea is also explored by Davis (2011), who futures that cloud computing will enable flash companies, that scale up fast and achieve striking results by dramatically reducing costs and completion task times.

On the other hand, crowdfunding has probably its roots in the idea of Muhammad Yunus, the Bangladeshi economist and founder of the Grameen Bank, winner of the Nobel Peace Prize in 2006. Yunus improved the Bangladeshi economy by establishing a micro credit system that permitted small amounts of money to be loaned to those who had potential, but no money to start a viable business.

With these broad definitions, it becomes necessary to focus on different perspectives that try to classify crowdsourcing according to the methodologies and goals defined, as will be explained in the following section.

3.3 Crowdsourcing Perspectives

Multiple authors have proposed different typologies for crowdsourcing. Stanoevska-Slabeva & Schmid (2001) proposed the typology of *community supporting platforms for task - and goal - oriented communities* in 2001. It was proposed a few years before the term crowdsourcing appeared, but it was related posteriorly. Those task-oriented communities, where the driving force for participation is the achievement of a common goal, could be divided in three subtasks:

- **Transaction communities**, where the emphasis is on the performance of market transactions, which arise around electronic commerce. These communities are meeting places where buyers and sellers meet to discover the price of goods and services;
- **Design communities**, where the aim is the common design and development of a product, as for example open source communities, like Linux or Wikipedia;

- **Online learning communities**, dedicated to collaborative online learning, to establish a learning space for a certain subject, where participants can get both degrees and knowledge, as well as support for long-lasting learning.

Another possible categorization was later given by Yuen *et al.* (2011) that proposed four crowdsourcing categories: *voting system*, *information sharing system*, *game*, and *creative system*, as detailed in the following:

- **Voting system**, where voting tasks are proposed to a crowd, and the answer that the majority selected is considered to be correct. Voting can be used as a tool to evaluate the correctness of an answer from the crowd;

- **Information sharing**, where the aim is to share various types of information among the crowd. According to the authors, Wikipedia is an example of an information sharing system, because it is written by a crowd and the result is also distributed to a crowd;

- **Game**, a game system that produces useful metadata as a product. By taking advantage of people's desire to be entertained, problems can be solved efficiently by online game players. Image labelling or music annotation are possible examples;

- **Creative system**, based on the role of a human in creativity, considering that it cannot be replaced by advanced technologies. Creative tasks, such as drawing and coding, can be requested in creative systems.

Finally, Brabham (2013) proposed four dominant crowdsourcing types, based on the kind of problems being solved: *knowledge-discovery and management approach*, the *broadcast-search approach*, the *peer-vetted creative-production approach*, and the *distributed-human-intelligence tasking approach*, as detailed in the following:

- **Knowledge-discovery and management approach** is where online communities are challenged to uncover existing knowledge in the network, thus amplifying the discovery capabilities of an organization with limited resources;

- **Broadcast-search approach** is oriented towards finding a single specialist, who probably is outside the direct field of expertise of the problem, and who has the time and the skills to adapt previous work to produce a solution;

- **Peer-vetted creative-production approach** can be seen as the creative phase of a designed product opened to a network of Internet users, who send in a flood of submissions, including some superior ideas. A *good* solution is also seen as the popular solution that the market will support;

- **Distributed-human-intelligence tasking approach** is appropriate when a corpus of data is known and the problem is not to produce designs, find information, or develop solutions but to process data. It is similar to large-scale distributed-computing projects.

Another relevant aspect of crowdsourcing is to understand what can be crowdsourced, i.e., what kind of tasks can be suitable to be carried out by a crowd. According to Schenk & Guittard (2011), it is possible to divide those tasks in:

- **Simple tasks**, that can be carried out cheaply, but their implementation becomes an issue when the scale increases. In this case, crowdsourcing becomes relevant, since it makes it possible to reach a large number of individuals, and thus a powerful work force. Another characteristic of simple tasks, is that they do not stem from individual abilities, but from the low-cost realization of tasks on a large scale;

- **Complex tasks**, differently from simple tasks, require expert abilities to be accomplished. There are multiple reasons for a request of a complex task, like lack of either skills or satisfactory in-house solutions. The idea is to expect to receive a set of candidate solutions from the crowd, and then to select the solution that seems best suited;

- **Creative tasks**, refers to cases where creativity and uniqueness have a value *per se*. The point here is not to have a problem solved, but rather to benefit from the creative power of the crowd.

3.4 Crowd Characteristics

A crowd is, in its essence, a group of persons, crowdsourcers, that in different manners participate in a crowdsourcing task. A crowd can be set together explicitly or

implicitly, whether each participant is aware of the final goal of the crowd or not. Examples of explicit crowds include Wikipedia and crowdfunding, while social networks are paramount examples of implicit crowds, since users contribute with their posts without being aware of possible future uses of that information.

When setting a crowd, one of the common goals is to define a *wise* crowd, i.e. a crowd that can achieve the proposed goals. Different conditions have been identified that characterize wise crowds, i.e., what should be the characteristics of crowdsourcers, like those proposed by Surowiecki (2004):

1. **Diversity of opinion**, as each person have some personal information, even if it is just an eccentric interpretation of the known facts. For example, if a crowd of individuals thinks in the same exact way, they are unable to provide the variability that is needed to cancel the errors each one makes;

2. **Independence**, related to the fact that people's opinion is not determined by the opinions of those around them, as persuasive individuals can sway others to think in a certain way and null the diversity of opinion;

3. **Decentralization**, in which people are able to specialize and draw on local knowledge. Otherwise, the centralization can narrow and guide the course of information, turning the crowd less wise;

4. **Aggregation**, related to the existing mechanisms for turning private judgments into a collective decision. By using multiple sources to provide a collective decision, mechanisms that combine information are required.

As an example, if you ask a large enough group of diverse, independent people, to predict or estimate a probability, and then average those estimates, the errors each one of them makes in coming up with an answer will probably cancel themselves out, i.e., virtually anyone has the potential to plug in valuable information (Surowiecki, 2004; Greengard, 2011).

Notice the similarities between crowdsourcing and ensemble learning (see Section 2.4.2). Diversity, independence, and aggregation can as easily be associated with baseline classifiers in an ensemble, as with crowdsourcers.

3.5 Applications

Different fields like biology, social sciences, engineering, and computer science have been application areas of crowdsourcing (Leimeister, 2010). In computer science, and particularly in machine learning, crowdsourcing applications are booming. In Welinder & Perona (2010) people contribute to image classification and are rated to obtain cost-effective labels. Another interesting application is presented in Chen *et al.* (2011), where facial recognition is carried out by requesting people to tag specific characteristics in facial images. In Franklin *et al.* (2011) crowdsourcing is used to process queries that neither database systems nor search engines can adequately answer, like ranking pictures by subject areas, and in Tarasov *et al.* (2014) a novel approach is proposed to dynamic estimation of rater reliability, specifically suited for real-life crowdsourcing scenarios, where the task at hand is labelling or rating corpora to be used in supervised machine learning. In Liu *et al.* (2016) it is proposed to efficiently recognize astonishing regions of landmarks for exploring nice visiting and photographic points of interest in these landmarks.

Crowdsourcing is also integrated with mobile devices like in Lee *et al.* (2016), where a mobile crowdsourcing platform prototype is introduced aiming to help cities to improve public engagement with their residents, and in Reichenbacher *et al.* (2016), where two methods are proposed for assessing the relevance of geographic entities in a mobile context, considering users increasingly retrieve information in mobile situations and that relevance is often related to geographic features in the real-world.

Considering software engineering, in LaToza & Van Der Hoek (2015) it is proposed the decomposition of software development into microtasks, enabling crowds of developers to immediately and effectively contribute by generating, distributing, and coordinating software microtasks. An important survey of the use of crowdsourcing in software engineering is presented in Mao *et al.* (2017).

There are also a few applications of crowdsourcing for text classification. In Brew *et al.* (2010) economic news articles are classified using supervised learning and crowdsourcing, and in Mollá *et al.* (2016) a corpus is designed for the development and testing of text processing tools for evidence-based medicine with the use of crowdsourcing. In Costa *et al.* (2011c), Costa *et al.* (2011a) and Costa *et al.* (2011b) crowdsourcing is used to improve the classification performance in a humour classification problem.

Finally, in Costa *et al.* (2013b), crowdsoucing is used to fit user preferences with the use of a custom crowd.

The combined use of crowdsourcing with social media is also being explored. In Xu *et al.* (2016), a knowledge base model is proposed to detect and describe real-time urban emergency events by using crowdsourcing on social media. In Sun *et al.* (2016) a novel crowdsourcing algorithm is proposed for efficient identification of human groups in social networks, which addresses the challenge of querying suitable collaborative experts for cross organizational business processes. A different approach is proposed in Karataev & Zadorozhny (2016), where a novel social learning framework is introduced that allows anybody to author educational content in a form of mini-lessons, learn lessons by following adaptive learning pathways, as well as interact with their peers as in any social network, combining crowdsourcing, online social networks, and complex adaptive systems to engage users in efficient learning throughout the teaching process.

3.6 Crowd Platforms

Taking advantage of the crowdsourcing capabilities, many platforms emerged to provide means between requesters and crowdsources, such as the now widely used `Amazon Mechanical Turk` and `Yahoo! Answers`.

Amazon Mechanical Turk

Amazon Mechanical Turk[1], commonly abbreviated as MTurk, was launched publicly on November 2005. The name refers to the Mechanical Turk fake chess-playing machine constructed and unveiled in 1770 by Wolfgang von Kempelen. Despite its success as an automaton chess-player for more than 80 years, it was in fact a hoax. It was later discovered to be an illusion artefact where a human chess master was hiding and operating the machine, and not an automaton at all. Like the original Mechanical Turk, the Amazon Mechanical Turk proposes a solution based on humans to tasks that could not be easily done by an automaton. Amazon provides a service where requesters can ask for workers to complete tasks, and workers are allowed to complete them and potentially get paid. Tasks are usually known as Human Intelligence Tasks (HITS) and workers as providers in Mechanical Turk's Terms of Service, or more informally

[1] `https://www.mturk.com`

turkers. According to the Amazon Mechanical Turk Documentation[1], MTurk is suited to complete jobs that humans can do better than computers, like finding objects in photos, writing reviews of restaurants, movies, or businesses, translating text passages into foreign languages, getting the hours of operation of the business center within a hotel, determining if a hotel is family-friendly, or telling you the most relevant search results for a given phrase.

An Application Programming Interface (API) is provided by Amazon, so users can programmatically access the MTurk system, and thus be used in larger scale. Scripting, along with software applications, can then be developed.

Despite the success of MTurk, some authors point that it came at a price: its crowd lost capabilities, like naiveness and honesty. In Peer *et al.* (2017), authors compared MTurk with two similar and less popular platforms, namely CrowdFlower and Prolific Academic, and they have concluded that participants on both platforms were more naïve and less dishonest when compared to MTurk participants. They have also concluded that CrowdFlower and Prolific Academic participants were also much more diverse than those from MTurk. However, it is important to note that CrowdFlower and Prolific Academic are designed to be exclusively used in research crowdsourcing tasks, like those related with data annotation, and that the comparison done by the authors is based on those specific tasks.

Yahoo! Answers

Yahoo! Answers (YA) was launched on June, 2005, although in a private testing environment. The first beta version was made public in December 2005. It is considered a large and diverse question-answer forum, acting not only as a medium for sharing technical knowledge, but as a place where one can seek advice, gather opinions, and satisfy one's curiosity about a countless number of things (Adamic *et al.*, 2008).

Yahoo! Answers' users are ranked in seven levels: the greater the user level, the more privileges has the user, like the ability to post or answer more questions. It is similar to a level of confidence based on the user's previous activity.

Researchers have found potential in YA, and multiple studies have been done. In Kim *et al.* (2007b), the authors propose to identify the selection criteria people employ when they select best answers in YA, in the context of relevance research. They

[1] https://aws.amazon.com/documentation/mturk/

analysed the comments people left upon the selection of best answers to their own questions, and through an iterative process of evaluating the types of comments, the best-answer selection criteria were inductively derived and grouped into seven value categories: *content value*, *cognitive value*, *socio-emotional value*, *information source value*, *extrinsic value*, *utility*, and *general statement*. The *socio-emotional value* was particularly prominent in this study, especially when people ask for opinions and suggestions. In Liu & Agichtein (2008), the temporal evolution of YA is investigated, with respect to its effectiveness in answering three basic types of questions: factoid, opinion, and complex questions. They have crawled 96,000 questions and 1,150,000 answers during 2006 and 2007. Authors state that even though YA kept growing rapidly, its overall quality as an information source for factoid question-answering degraded. However, they suggest that instead of answering factoid questions, YA might be more effective to answer opinion and complex questions.

Crowdfunding platforms

Crowdfunding, according to the Oxford Living Dictionaries[1], the practice of funding a project or venture by raising money from a large number of people who each contribute a relatively small amount, typically via the Internet, has also dedicated platforms.

The idea that some people may decide to pay for producing and promoting a product (instead of buying it), and bear the risk associated with that decision, represents a further step in the evolution of consumers' roles and requires distinctive platforms that are able to handle the financial transactions involved (Fisk *et al.*, 2011).

In late 2005 a non-profit organization called Kiva was founded. Inspired in Yunus ideas, Kiva.org[2] continues to be an important platform of crowdfunding by connecting people that lend small amounts of money to permit the development of projects in poor countries. Recently, there has been a boost of crowdfunding platforms, like profounder[3], catarse.me[4], or eppela[5]. These platforms are used by common people to invite the crowd to finance their projects, like recording a CD, editing a magazine, or buying material to improve their jobs.

[1] https://en.oxforddictionaries.com/
[2] http://www.kiva.org
[3] http://www.profounder.com
[4] http://catarse.me
[5] http://www.eppela.com

3.7 Challenges

Despite the enormous potential for problem solving, specially if one can benefit from a wise crowd, crowdsourcing encompasses a set of issues that should be brought into discussion. In Doan *et al.* (2011), four key challenges were identified: *how to recruit contributors*, *what can they do*, *how to combine their contributions*, and *how to manage abuse*. These challenges become even more relevant when dealing with implicit crowds as in social networks.

It is important to evaluate the motivations of the contributors, if this contribution should (or should not) be gratified, how can contributors be retained and encouraged to pursue, if the contributions are somehow limited by the contributor, and how can these limitations be controlled or avoided.

Combining the contribution of each individual can also be complex, as there are many possibilities, like the average agreement of users, the ranking of users and the concept of most reliable, or combined metrics, where users are rated and a combined metric defines the result of the crowd (Rodrigues *et al.*, 2013).

It is also relevant to detect and manage malicious users, specially with wide accessible crowds as in social networks. Such users can not only contribute maliciously, decreasing the potential of the crowd, as they can influence other users, propagating their malicious consequences. In Cox (2011) the concept of truth in crowdsourcing is analysed. The author uses Wikipedia[1] as an example to identify major threats, pinpointing the absence of self-policing and the fast spread of news and ideas, which can be inaccurate or promoted by dishonest people to find their way to mainstream. There can be obscure actions combined with ignorant or apathetic individuals inside the crowd.

3.8 Conclusion

In this chapter we have presented an overview on crowdsourcing, showing that a growing number of real-world problems have taken advantage of this technique, e.g. Wikipedia, Linux, and social networks in general. With this mindset, and given the enormous challenges and opportunities, a strong stimulus exists to join the advantages of crowdsourcing with the advantages of social networks. Differently from a crowdsourcing platform, like Amazon Mechanical Turk, one can use an implicit crowd, i.e., crowdsources

[1] http://www.wikipedia.org

that are considered because they provide relevant information rather than because they profiled themselves as crowdsources.

Additionally, in machine learning, the scope of this thesis, crowdsourcing can be seen as a distributed classification method in which a crowdsourcer submits a complex task to groups of people, termed crowds, in order to obtain different solutions for further analysis and evaluation. In crowdsourcing the distributed problem solving model is based on interoperability between humans (crowd) and computers (evaluation analysis) that work together to solve complex tasks, like those related to annotation, recommendation, and classification of contextual examples. Regarding classification settings, the crowd has to deal with intrinsically subjective tasks, that usually include contextual, semantic, and sentiment analysis. The use of crowsourcing as a data source for learning settings is the main subject of the following chapter.

Chapter 4

Active Learning and Crowdsourcing

In this chapter we focus on active learning and crowdsourcing to improve classification performance. We start by presenting the case study used in our experiments, a humour classification problem. We then discuss an active learning strategy to enhance classification performance presenting three contributions. A baseline model is set up to define which examples should be used for active learning, and then an active learning strategy is put forward comprising the use of a supervisor versus the use of crowdsourcing. Finally, we present a crowd customization strategy to overcome previously identified limitations in the use of crowdsourcing.

4.1 Introduction

Crowdsourcing, introduced in the previous chapter, emerged as a new paradigm for using all the information and opinion shared among Internet users. Hence, this technique is capable of aggregating talent, leveraging ingenuity, while reducing the cost and time needed to solve problems. Moreover, crowdsourcing is enabled through the technology of the web, which is a mode of user interactivity, and not merely a medium between messages and people (Brabham, 2008).

Along with crowdsourcing, active learning can also extend the capabilities of a learning model, particularly when more labelled examples are needed, but are also scarce or expensive to obtain. The promise of active learning is that when the examples

to be labelled are properly selected, data requirements for some problems decrease drastically (Schohn & Cohn, 2000). Active learning is then based on the idea that a machine learning algorithm can achieve greater accuracy with fewer training labels if it is allowed to choose the data from which it learns. Thus a fewer number of examples could be required to be labelled by a supervisor, an oracle, or even a crowd if we merge active learning with crowdsourcing capabilities.

Dynamic environments can also benefit from the above mentioned potential. Using a spam classifier as an example, current generic classifiers are often deemed insufficient to fulfil user's expectations, since they can vary tremendously among users. An email can be legitimate for a given user, but undoubtedly spam for another user. To tackle the problem of user dynamics, an active learning strategy that uses crowds as source of annotated information can be used. A crowd with similar users can be used to train the model, and thus benefiting from both techniques. Such a system, able to adapt to user drifts, can be specially relevant in the customization process of a recommendation system, where usually highly subjective issues arise, like book or movie recommendations.

4.2 Crowdsourcing: Case Study

Crowdsourcing has some interesting features: the implicit heterogeneity provided by the crowd members, the intrinsic ability to cope with highly subjective tasks, and the fact that it can be a low cost working force turn it into a valuable technique that is worth exploring in classification problems. This is specially relevant for problems dependent on social, cultural, or emotional backgrounds, like book or movie recommendation, and thus difficult to a machine learning algorithm to cope with.

In classification scenarios, a large number of tasks must deal with inherently subjective labels and there is a substantial variation among different annotators (refers to a person labelling a set of examples). One of such scenarios is text classification (Raykar *et al.*, 2010) and particularly humour (jokes) classification, as it is one of its most interesting and difficult tasks. The main reason behind this subjectivity is related to the contextual meaning of each joke, as they can have religious, racist, or sexual comments. However, in spite of the attention it has received in fields such as philosophy, linguis-

tics, and psychology, there have been few attempts to create computational models for automatic humour classification and recommendation (Mihalcea & Strapparava, 2005).

The applications are multiple, as nowadays people often search for humour as a relaxing proxy to overcome stressful and demanding situations, having little or no time to search contents for such activities. Hence, the definition of personal models that allow the user to access humour with more confidence on the precision of her/his preferences are of major interest. Besides, while it is merely considered a way to induce amusement, humour also has a positive effect on the mental state of those using it, and has the ability to improve their activity (Mihalcea & Strapparava, 2005; Stock & Strapparava, 2003).

Nevertheless, considering related work in humour classification, we must mention that humour research in computer science has two main research areas: *humour classification* (Mihalcea & Strapparava, 2006, 2005; Reyes *et al.*, 2010) and *humour generation* (Stock & Strapparava, 2003; Binsted & Ritchie, 1994). With respect to the latter, we will not enter into as it is not in the scope of this thesis.

Humour classification is intrinsically subjective. Each one of us has its own perception of fun, hence automatic humour recognition is a difficult learning task that is gaining interest among the scientific community. Classification methods used so far are mainly text-based and include diverse classifiers, like Support Vector Machines (SVM), *Naïve Bayes* and decision trees. The research carried out so far considers mostly humour in short sentences, like *one-liners*, that are jokes with only one sentence, and the improvement of interaction between applications and users.

In Mihalcea & Strapparava (2006) a humour recognition approach based on *one-liners* is presented. A dataset was built by grabbing *one-liners* from the web, using web search engines. This humourous dataset was then compared with non-humourous datasets like headlines from news articles published in the Reuters newswire and a collection of proverbs.

In Reyes *et al.* (2010) another interesting approach is proposed to distinguish between an implicit funny comment and a not funny one. The authors used a 600,000 web comments dataset, retrieved from the Slashdot news website. These web comments were tagged by users into four categories: *funny*, *informative*, *insightful*, and *negative*. The dataset was then split in humourous and non-humourous comments.

We will further detail the jokes classification problem and then present the Jester dataset, that was used in our experiments regarding jokes classification.

4.2.1 Problem Setting

Jokes classification can be handled as a binary task formalized as approximating an unknown target function $f : \mathcal{J} \times C \longrightarrow \{-1, 1\}$ that corresponds to learning a mapping function that is able to classify jokes as a human. The function f is the jokes classifier, $C = \{c_1, c_2, \cdots, c_{|C|}\}$, and $\mathcal{J}$ is a set of jokes. Each joke $\mathbf{j}$ has a simple document representation, which is the vector space model also known as bag of words (see Section 2.3.1). The joke is represented as a set of features, usually words, $\mathcal{W} = \{w_1, w_2, \cdots, w_{|\mathcal{W}|}\}$ with each one as a vector $\mathbf{j}_i = (w_{i1}, w_{i2}, \cdots, w_{i|\mathcal{W}|})$ where w_{ik} describes each feature representation for the specific joke. In this representation, each joke is indexed with the bag of the terms occurring on it, i.e., it is a vector with one component for each term occurring in the whole collection. When the mapping function takes the value 1, i.e., $f(\mathbf{j}_i, c_j) = 1$, then $\mathbf{j}_i$ is a positive example or member of class c_j, otherwise it is a negative example of c_j. Given the binary classification problem the cardinality of classes is two ($|C| = 2$).

4.2.2 Jester Dataset

To validate and evaluate the humour classification strategy we have used the Jester dataset. The Jester dataset contains 4.1 million continuous ratings (-10.00 to +10.00) of 100 jokes from 73,421 users, and is available at: `http://eigentaste.berkeley.edu`. Figure 4.1 presents an example of a joke in the dataset. It was generated from Ken Goldberg's joke recommendation website, where users rate a core set of 10 jokes and receive recommendations from other jokes they could also like. As users can continue reading and rating, and most of them end up rating all the 100 jokes, the dataset is quite dense.

The dataset is provided in three parts: the first one contains data from 24,983 users, the second one from 23,500 users, whilst the third one contains data from 24,938 users. The users from part one and part two have rated 36 or more jokes, while the users from the third part have only rated between 15 to 35 jokes. The experiments were carried out using the first and the second part as they contain a significant number of users that rate all jokes.

```
When my three-year-old son opened the birthday gift from his
grandmother, he discovered a water pistol.  He squealed with delight
and headed for the nearest sink.  I was not so pleased.  I turned to
Mom and said, "I'm surprised at you.  Don't you remember how we used to
drive you crazy with water guns?"

Mom smiled and then replied...
"I remember."
```

Figure 4.1: Example of a joke from the Jester dataset

For classification purposes it was considered that a joke classified on average as 0.00 or above is a recommendable joke, and a joke below that value is not recommendable. Jokes were split into two equal size disjoint sets: training and test. The data from the training set is used to select learning models, and the data from the testing set to evaluate performance.

Considering jokes representation, we have used the already mentioned bag of words, along with preprocessing methods, namely stemming and *stopword* removal (see Section 2.3.2), to reduce feature space.

4.3 Enhancing Classification with Active Learning

Traditional classification algorithms can be limited in their performance when the problem is highly subjective, as the above mentioned humour classification. As a consequence, more training examples are needed. However, in some problems, there is a significant amount of unlabelled data, but labelled data is scarce or expensive. To cope with these constraints in these kind of problems we propose a strategy to take advantage of active learning, described in Section 2.3.3, aiming to improve classification when labelled data is scarce, like in our case study, humour classification.

The underpinning idea behind active learning is that a machine learning algorithm can achieve greater accuracy with fewer training labels if it is allowed to choose the data from which it learns. An active learner may pose queries, usually in the form of unlabelled data instances to be labelled by a supervisor. The number of active learning examples can not be large, since the supervisor is usually asked to manually classify them. After being correctly classified, they can be integrated in the training set. Hence,

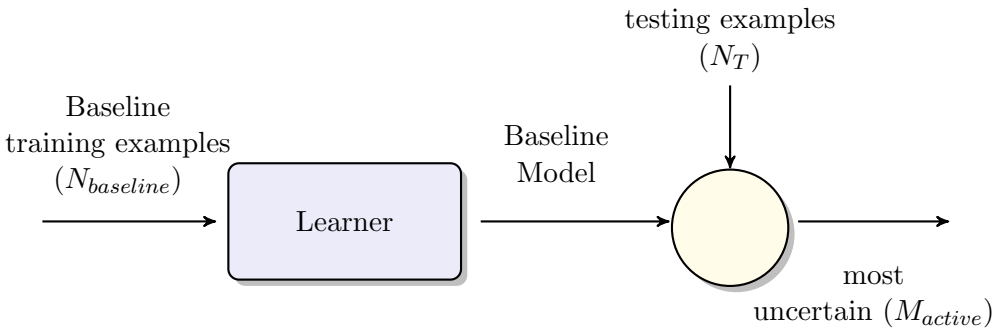

Figure 4.2: Baseline Approach

the definition of a subset of active examples is crucial to the active learning approach, as the proposed strategy will be put forward based on this subset.

In this section we will describe our strategy to evaluate the feasibility of enhancing classification with active learning. We will start by defining the baseline approach that will determine which examples will be used in the active learning process.

4.3.1 Baseline Approach

The baseline approach is then the starting point of the methodology used in this chapter. It is depicted in Figure 4.2. As already mentioned, it will not only be used for comparative purposes, but also to define the examples that will be considered for active learning. The confidence in the classification of data points in the testing set with the baseline model is used to identify which are the most informative examples. Therefore, those examples classified with less confidence will be used as active examples in the subsequent approaches.

A certainty-based strategy is put forward, by using a baseline model to determine the most uncertain examples (M_{active}), also referred from now on as active learning examples, and pointing them to be used in an active learning strategy. The certainty is dependent on the learning method. For instance, using margin-based algorithms, as the SVM classifier we have chosen, we can use the classification margin provided by the baseline model as the determining factor. When an SVM model classifies new unlabelled examples, they are classified according to which side of the Optimal Separating Hyperplane (OSH) they fall. As can be gleaned from Figure4.3, not all unlabelled points are classified with the same distance to the OSH. In fact, the farther they are

from the OSH, the larger the margin, i.e., more confidence can be put on their classification, since slight deviations from the OSH would not change their given class. Thus, an example $(\mathbf{x}_i, y_i)$ will be included in the active learning strategy if Equation (4.1) holds.

$$(\mathbf{x}_i, y_i) : \rho(\mathbf{x}_i, y_i) = \frac{2}{\|w\|} < \Delta \tag{4.1}$$

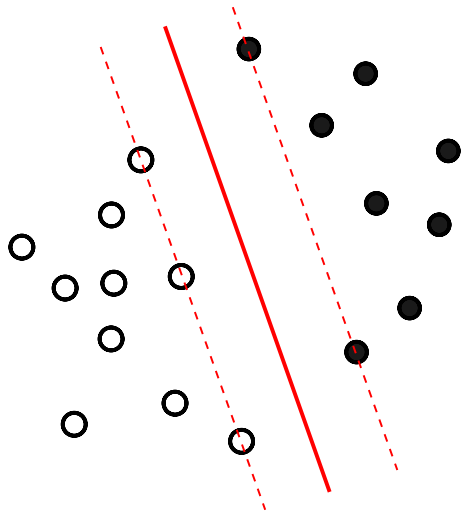

Figure 4.3: SVM Optimal Separating Hyperplane (OSH) representation

The information introduced by each example in the classification task is inversely proportional to its classification margin, as those located far from the margin are those in which the classifier does not have doubts, while those located near the margin can more easily be assigned to the opposite class. As a consequence, if more informative sample data points, i.e., more relevant information regarding the classification task, can be provided to the classifier, more efficiently the classification performance can be improved, with fewer labelled examples, based on the idea that if the learning algorithm has the ability to choose the learning examples, they can be more adequate to the learning process (McCallum & Nigam, 1998; Tong & Koller, 2002; Dan, 2004).

To construct the baseline model, the dataset examples are equally split into training ($N_{baseline}$) and testing examples (N_T). Then, the baseline model is built using the training examples, referred from now on as baseline training examples, and tested using the testing examples. In this baseline approach the training examples are generic

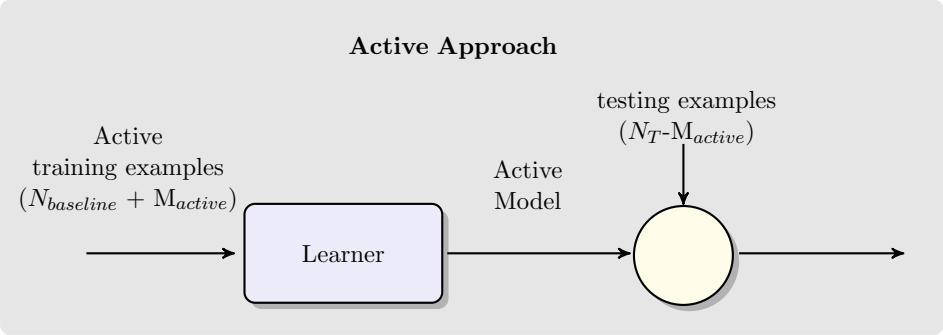

Figure 4.4: Active Approach

in the sense that they include the contribution of all the users that rated the Jester dataset examples, and each user participating equally to the final decision.

4.3.2 Research Design with Active Learning

The key factor in any active approach is the determination of the active examples. Figure 4.4 depicts the active approach. We adopt a certainty-based strategy to determine the most uncertain examples (M_{active}), named active learning examples. These active examples are then added to the baseline training examples ($N_{baseline}$) and removed from the testing training examples. The newly defined testing set is defined as N_T - M_{active}. The classification of these new training examples is still generic, in the sense that it is not customized for any given user and is obtained with the contribution of all the users that rated the Jester dataset examples.

However, the active learning approach we proposed is in fact a three-fold active learning approach. We have conducted experiments in order to compare the above mentioned baseline approach with three different active learning strategies. We have thus defined:

1. Active Learning SVM using a set of arbitrary examples (Random AL SVM)
2. Active Learning SVM using a set of the most relevant examples, a margin-based dataset, correctly classified by a supervisor (Margin AL SVM)
3. Active Learning SVM using a set of the most relevant examples, the same margin-based dataset used in (Margin AL SVM), classified by crowdsourcing instead of the supervisor (Crowd AL SVM)

For the first experiment, Random AL SVM, 30 runs were carried out, by randomly selecting 10 active examples, and average values are presented. The number of examples used can not be large, since in the next experiments the supervisor and the crowd will be asked to manually classify them. For the second and third experiments, namely Margin AL SVM and Crowd AL SVM, we have used the margin-based strategy we have previously described. After being classified, they are integrated in the training set.

4.3.3 Experimental Results and Analysis

We have used the SVMLight[1] package with linear kernels and default parameters.

The obtained results of our three-fold approach regarding the use of active learning are presented in Table 4.1.

	Precision	Recall	F_1
Baseline SVM	81.40%	92.11%	86.42%
Random AL SVM	84.36%	84.74%	83.81%
Margin AL SVM	87.80%	94.74%	91.14%
Crowd AL SVM	81.82%	94.74%	87.80%

Table 4.1: Comparative results considering precision, recall and F_1

Analysing the table we can see that active learning does not necessarily improve the classification performance. When we randomly choose the examples presented as active learning the performance can even decrease, as can be seen by comparing the Baseline SVM approach, with 86.42% regarding F_1, and the result obtained by the Random AL SVM approach, 83.81%. It is also important to note that more examples were presented to the classifier in the Random AL SVM, because instead of $N_{baseline}$ we have used $N_{baseline} + M_{active}$, which turns out to be a more costly solution. However, when the active learning examples are correctly chosen, a remarkably important performance boost is noticed. Both recall, precision, and F_1 were improved, and the enhancements are robust regarding false positive and false negative examples.

Considering only the use of active learning, and discarding for now the use of crowdsourcing, we can see that there is a trend for improvement in precision: 81.40%, 84.36%

[1] http://svmlight.joachims.org/

and 87.80%. This can become a determining factor in some problems where precision is more relevant than recall, like in humour classification, since users are typically more interested in a strong confidence of amusement (low false positive values) than in the guarantee of getting all jokes (low false negative values).

Comparing random active approach and margin-based active approach, we can see that although both present improvements in precision, the random approach achieves it at the expense of recall values, while in proposed margin-based active learning the improvement is on both recall and precision.

Another important analysis must be done regarding the use of crowdsourcing in the active learning process. When crowdsourcing is used we were able to verify a slight improvement considering baseline, as we obtained 87.80% F_1 score against the 86.42% of the baseline approach. Although only a slight improvement over F_1, slightly more than 1%, this method improves across all metrics.

There are a few explanations for this minor improvement when compared with the obtained results of using the margin-based active learning approach, which means comparing the use of active learning classified by a supervisor or by a crowd:

1. Some problems are subjective, and humour is undoubtedly one of them, as it is influenced by the contextual meaning of the joke, and can vary accordingly with culture, region, race, or sex.

2. The definition of crowd is subjective, as it is difficult to acquire if the correct size of the crowd is used, or that the crowd is diverse enough to provided substantial results.

3. The supervisor does not fail, although in some real-world problems it is impossible to use correct label data as active learning examples, others exist where the use of active learning is considered just to limit the number of examples given to a supervisor, who can easily label correctly the presented data.

4. The crowd is not sure, as it is impossible to acquire if annotators can really correctly classify the given examples. In this example the crowd used was mostly Portuguese, i.e., non-English native, and some jokes were intrinsically related to the American culture, which could explain the results, as just 6 jokes (out of the 10 initially chosen) were correctly classified.

Considering the experiments done and the obtained results regarding the use of crowdsourcing, and bearing in mind the successful accomplishments in many fields, we can conclude that crowdsourcing can be seen as a promising technique. However, the presented results show that it can be fallible in accomplishing specific tasks, even though we have compared it with an assertive supervisor, which can be infeasible in many real-world problems. Yet, it is somewhat difficult to evaluate the appropriateness of a crowd in such a subjective classification problem, and more research must be done in order to confirm its aptness. In the next section we will detail experiments to evaluate if it is possible to customize a crowd to improve the classification performance and address the observed limitations.

4.4 Crowd Customization

It is important to define the appropriate scope of a crowd to tackle the spotted above debilities of using a crowdsourcing strategy in highly subjective tasks, like humour classification. The problem of defining the best suited crowd for improving the classification performance of a model can be seen as a crowd customization problem, and it is particularly important in recommendation applications where users' preferences might be dependent on social, cultural, or emotional backgrounds. In this case, a model can be fitted to recommend a book to a group of people that share common interests or the same cultural background, but fails to acknowledge the recommendation in a different context.

The rationale behind crowd customization is the possibility of models customization based on user preferences. It can be casted as a classification problem, since the classification model must adapt to the preferences of each new user. As previously referred, those preferences can depend on age, cultural context, geographic location, etc. Model customization can be a hard task, since ground-truth is often difficult to determine, as it is quite improbable that all users have the same opinion about a given item, drastically reducing the possibility of having labels that are applicable in all possible scenarios and/or users. Hence, there are two problems that must be taken into account: (i) the subjectivity and (ii) the reliability of crowd users (annotators).

To adapt the model for a given user it is necessary to have complementary information that allows the learning machine to identify the user profile. This information

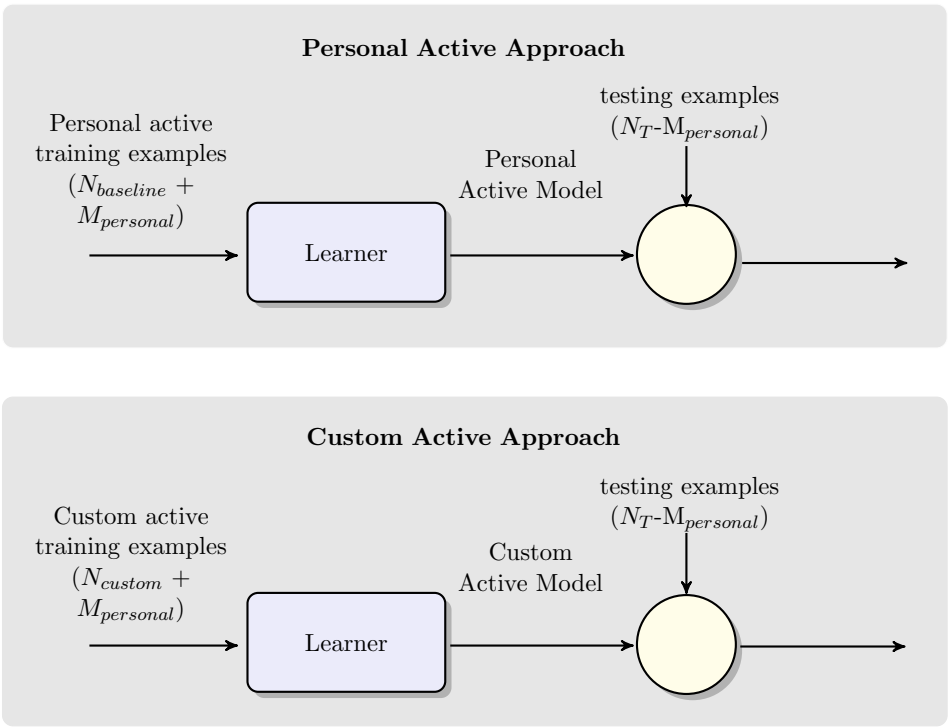

Figure 4.5: Personal Active Approach and Custom Active Approach

can be supplied explicitly by user profiles, or implicitly by using user feedback.

In Figure 4.5 we illustrate the two learning approaches to crowd customization, namely *personal active approach* and *custom active approach*, that we will detail further.

4.4.1 Personal Active Approach

We propose dealing with crowd customization using user feedback to characterize each user and choosing the appropriate crowd to minimize the personal feedback required. To integrate the user feedback into the learning process, we have used an active subset of examples chosen with the same strategy described in Section 4.3. By either asking each new user, or a *customized* crowd, to classify those examples, we aim to construct the appropriate user profile, just before building the customized model.

Differently from the strategy proposed in Section 4.3.2, in the *personal active approach*, the new user presented to the recommendation system must classify the active learning examples according to his preferences, in order to define his profile and promote the customization. These examples are then named personal active training examples

($M_{personal}$) and, along with the generically training examples, i.e., the $N_{baseline}$ examples, are used to train a more customized model.

In this approach the number of personal active training examples has to be necessarily small, since the user is required to provide a classification. Nonetheless, the adequacy of this number is user and task dependent.

Using a recommendation system as example, the active examples are directly classified by the user instead of being classified by the crowd, providing an obvious advantage in terms of representativeness of the training dataset.

4.4.2 Custom Active Approach

In the SVM *custom active approach* we take the strategy one step further. On one hand we use the active examples personally classified by the user ($M_{personal}$) as in the previous approach, but we also customize the baseline examples that were used so far. To achieve this customization, instead of using the crowd contribution to determine the classification of the baseline examples ($N_{baseline}$), we choose a customized crowd, i.e., a crowd with closer preferences to our target user, resulting in a customized set of examples (N_{custom}).

As already stated, the new user profile is defined by his/her classification in the active learning examples, therefore the closeness between individuals can just take into account the classification of this subset. The baseline training examples classified by this customized group from the crowd is referred as custom examples (N_{custom}).

The main idea behind this approach is to use not only the classification of the user, but to adjust the baseline training examples by restricting the contribution of the previously seen individuals to those that are closely related to the new user. The underpinning idea is that the information provided by them can be more valuable, as it avoids the bias provided by using remarkably distinct users when compared to the one we intend to customize our model for. It is also important to refer that this approach also avoids asking the new user to manually classify the whole training example set, specially when it is sometimes infeasible to ask for such contribution, and thus decreasing variance.

In the next section we will present the appropriate closeness metrics that can be applied to determine the customized crowd.

4.4.3 Closeness Metrics

Considering a generic recommendation system, we usually have a large set of ratings for each example. When comparing two users, a straightforward technique is to use the *sum of the absolute errors* between examples being classified.

Taking I as the collection of items, a and b as two different users, one can estimate the closeness using:

$$\sum_{i \in I} |C_i^a - C_i^b|, \tag{4.2}$$

where C_i^a is the classification of item i given by user a.

Using such a similitude measure to determine the closeness between two users, we can then choose a subset of the users that compose a crowd, using only the k users closer (more similar) to a given user. Such a customized crowd is then suited to provide information for customizing a model to classify documents matching the user's preferences.

4.4.4 Experimental Results and Analysis

Table 4.2 shows the recall and precision results for both levels of crowd customization, *personal active approach* and *custom active approach*. We considered that 10 jokes were deemed sufficiently non-intrusive for a user to classify, and the closer crowd in the *custom active approach* was heuristically defined to include the $k = 1000$ users with closest preferences to the user (see Section 4.4.3).

We may also observe that whilst precision values are rather similar, there is a relevant difference of circa 5% in recall values. This difference results in more relevant examples being discovered. Considering a recommendation system as a possible application, one may argue that this can make the difference in the user's evaluation of the service provided, as we are providing more examples close to his/her preferences.

Regarding F_1 we have macro-averaged values of 77.42% for the *personal active approach* and 79.13% for the *custom active approach*. As expected by the difference in recall values, the *custom active approach* presents a better overall performance. While the *personal active approach* only uses the 10 active examples, it is outperformed by the *custom active approach* that takes customization one step further by using the

	Precision	Recall
Personal Active Approach	72.78± 0.17%	82.69± 0.13%
Custom Active Approach	72.35± 0.18%	87.32± 0.20%

Table 4.2: Precision and recall performances for active approaches

similarity measure, defined in Section 4.4.3, to choose the crowd that is closer to user preferences.

Finally, notice that the values of Table 4.2 and the results already presented in Table 4.1 (Section 4.3.1) are not directly comparable. While the active learning results are generic, in the sense that we are building a model that aims to classify the examples regarding the sense of the whole population, the personal and custom results are personalized. The goal in the active learning approach was set as the average of the classification of each joke, while the goal in the custom and personal approaches is distinct for each user, thus much harder to learn.

The results obtained revealed the usefulness of using crowds in the adjustment of user preferences. More precisely, we determined that specially chosen crowds are able to retrieve customized user preferences. This is an interesting insight considering that crowds can be tuned to model customization, specially in highly subjective and difficult tasks, where traditional algorithms can be limited in their performance.

4.5 Conclusion

In this chapter we firstly introduced the crowdsourcing case study we have used, a humour classification problem. We started by the problem setting along with the characterization of the Jester dataset, a freely available dataset for humour classification. We then detailed our approach to enhance classification with the use of active learning, by designing a baseline model to define which examples are more informative, and thus should be used as active learning examples. A three-fold experiment was then put forward, using random active learning examples, classifying the most informative results with the use of a supervisor, and finally classifying the same set of examples with the use of crowdsourcing.

The obtained results attest the importance of correctly defining the subset of active learning examples, but had some pitfalls regarding the use of crowdsourcing. Considering those limitations, we then propose a different strategy based on crowd customization. We presented two distinct models, along with the obtained results and the final analysis. The results obtained revealed the usefulness of using crowds in the adjustment of user preferences in a humour recommendation case study.

We have shown that both active learning and crowdsourcing techniques, due to their intrinsic properties, can be used to improve the classification performance in complex problems. However, accessing a crowd and manipulating its contribution for problem solving is not straightforward, as it can be costly and time consuming. In the next chapter we will introduce social networks, as their massive use, dynamic nature, and potential in information spread can be seen as important characteristics for problem solving if considering they might constitute a potential implicit, and low-cost, crowd.

Part III

Learning in Dynamic Environments

Chapter 5

Social Networks

In this chapter we focus on social networks. We firstly present a background on social networks, describing the most accessed ones, along with their history and features. We detail Twitter and its characteristics as it is the case study we have used in the context of this thesis. We then present the applications of social networks and the current research areas that they include, detailing not only research paths but also pinpointing the major challenges in the field. Afterwords, a Twitter classification solution is proposed to classify Twitter messages based on their *hashtag*, and finally we extend the contribution by defining the concept of *meta-hashtag*, providing a study on the benefits of clustering similar messages in Twitter.

5.1 Introduction

The constant growth of platforms like Twitter, or Facebook, shows the importance of social media in nowadays people's lives. We are now all connected to friends, family, and communities, all around the world, and share information on a daily basis.

In recent years different social networks have been created and popularized, like the above mentioned Twitter and Facebook, but also Instagram, LinkedIn, Pinterest, Orkut, or Google+. And even though the first social networks were mostly text-based, it has now become popular to share multimedia content, too, like photos, music, or video. More recently live sharing is also being implemented and used in several social networks.

Although mostly considered as an entertainment tool, some networks can be an important source of information, since relevant and valuable information is publicly shared among users. Three major conditions can be put forward for the success of social networks and their reliability as source media: (*i*) the variety of contexts in which they can arise, (*ii*) their ability to grow naturally; and (*iii*) their intrinsic capability for fast spread (Doerr *et al.*, 2012).

Considering their relevance, specially in information spread, they have gained increasing importance and are being widely studied in many fields of research, specially due to their multiple applications.

Crowdsourcing through social networks turns out to be a quite appellative application. The potential of crowdsourcing, described in Chapter 3 is undoubted, not only because under the right circumstances groups can be remarkably intelligent and efficient, but also because there are tasks that are notoriously difficult for an algorithm to perform and quite simple for humans, like speech or image recognition, language understanding, text summarization and labelling (Barr & Cabrera, 2006). Considering the above, and that, in social networks, users have the ability to post, to comment, or to *like* posts, according to their personal status or preferences, it is possible to perceive the potential of social networks as source of implicit crowds. By stating their opinion about a post or picture, each user can, for instance, feed a recommendation system, promoting a particular post to be seen by the user's connections. Social networks are thus fed with data originated from a crowd, whose members are its users. This means that data processed by crowd-based social networks is a valuable source of information.

5.2 Background

The first social networks as we know today started to appear in the beginning of the 2000s. They were firstly based on user profiles and forums, a merged solution between personal web pages and the already popularized news servers. The interaction between users was not meant to be synchronous, as it was not common for users to be online all day long. However, during the 2000s, the Internet became widespread to people's homes, and more importantly, mobile devices started to appear in people's lives.

Soon, Internet and mobile devices were combined, and nowadays reality is that everything is connected, everywhere and every time. As a consequence of this phe-

nomenon, along with the improvements on data communication technologies, social networks not only boomed, but also changed their paradigm, turning them more real-time based. They have then settled definitely in the daily routine of Internet users, as a mean for real-time connection with friends, family, and access to broad information.

In the following, we will present a short overview of relevant social networks, namely LinkedIn, Orkut, Facebook, Youtube, Pinterest, Instagram, Google+, and Twitter. This list is organized in a chronological order of their creation, except for Twitter that will appear lastly as it constitutes an important foundation of the techniques proposed in this thesis, and must be more profoundly contextualized.

5.2.1 Relevant Social Networks

LinkedIn

LinkedIn[1] was created in 2002 and launched publicly in the subsequent year. Unlike all the other networks, the purpose of LinkedIn is professional advertisement and not sharing personal text, photos, or video. It is the most relevant business-oriented social networking service, and the eighteenth most accessed website according to Alexa.com[2]. This might not seem impressive considering Facebook or Twitter, but one must take into account the context of LinkedIn as a professional advertisement tool for workers and employers.

The *modus operandi* of LinkedIn is to allow users (workers and employers) to create profiles and connections between each other that represent real-world professional relationships. It is a relevant tool for those who are interested in finding a job, as it allows users to search for companies in which they may be interested in working for, and for those who are recruiting, as it is today used by recruiters as a source for finding potential candidates. A recommendation system is also put forward so users can recommend the professional characteristics of those who they have worked with.

Orkut

Orkut was an important social network but it no longer exists. It was created in 2004 and it is only mentioned because it was one of the most relevant social networks, and one of the most accessed ones in the beginning era of social networks. The success

[1] http://www.linkedin.com
[2] http://www.alexa.com

of Orkut was mainly due to the amount of users from Brazil and India, and due to that success Google, who owned Orkut, announced in 2008 that Orkut would be fully operated in Brazil.

A significant difference from Orkut and the other social networks is that by default all profiles are public and any user could see all the other profiles. Still, the concept of friends and connections between users was always present. Another feature of Orkut was communities, that were groups of discussion in which users could post about a particular issue.

Due to Facebook and Twitter, who rapidly gained interest worldwide, the number of Orkut users started to decrease, and in 2014 Google announced that the social network would be shutting down completely.

Facebook

Facebook[1] was created in 2004, being almost a private network. In 2006 it started to accept users with at least 13 years of age and since then its popularity has grown considerably. It became the third most accessed website in the world according to Alexa.com, and the first most accessed social network if one does not consider YouTube as a social network, as Google is the rank leader, followed by YouTube.

Facebook's concept is different from the rest, with a wide range of sharing possibilities, like photos, events, and videos. It also supports pages for companies, institutions, organizations, games, and applications. Live video broadcast is a recently added feature for mobile and it is becoming popular among users.

Youtube

Youtube[2] can be seen as a controversial social network. It was created in 2005 as a video-sharing website. The features were limited, and by then Youtube could not be considered a social network as there was not any relation between users. However, the website evolved, and today we can see Youtube as a social network as we have now user profiles and following mechanisms that reflect the social relations between users. We are now connected to those profiles we intend to watch videos from, and we can also maintain a Youtube channel to broadcast for our audience.

[1] http://www.facebook.com
[2] http://www.youtube.com

According to Alexa.com, Youtube is today the second most accessed website in the world, and it is now used for streaming media, not only by private users, but also by important media companies like BBC, CBS, or Vevo.

Pinterest

Pinterest[1] was launched in 2010 as a photo sharing social network. Users can save and sort images, also known as pins, through collections that are known as pinboards. A pinboard is usually organized as a theme, sorting images by similar characteristics, though this is not mandatory, as users can manage pinboards as they intend to. There is also a pin feed, where users are presented with their last chronological activity. These pin feeds, that represent users' activity, can be follow by others, managing that way the relation between users and pinboards.

It has more than 100 million active users and, differently from other social networks, in Pinterest there is a significant difference in the percentage of female users, 85%, and the percentage of male users, the remaining 15%. Among the most used categories that are pinned in Pinterest there are food related pins, arts, DIY, travel, and fashion.

According to Alexa.com, Pinterest is ranked 35 in the most accessed websites in the world.

Instagram

Instagram[2] was created in 2010 as a free mobile application, but rapidly gained interest of millions of users. Differently from all the others, it was created not to be a web-based social network, but as a mobile application where a photo sharing service was provided. Users could share square framed photos to their peers with simple text messages as comments. The square framed photo was the brand footprint and a different aspect ratio was only considered in 2015, almost 5 years from its debut.

It is, according to Alexa.com and by now, the fifth most accessed website in the world. As an example of the importance of Instagram, we can identify the popularization of the concept of *selfie*, that became not only an accepted word in the Oxford English Dictionary, but a concept we all identify nowadays. Though the concept was not created by Instagram, the strong bond between them might be explained by being

[1] http://www.pinterest.com
[2] http://www.instagram.com

the most popular image-based social network with a close relation to front cameras in mobile phones.

Google+

Google+[1] is today's Google social network. Launched in 2011 it has more than 100 million active users and it was the fourth social network created and managed by Google, after Google Buzz, Google Friend Connect, and Orkut.

Google+, sometimes abbreviated as G+, was initially created as a Google social networking service, but later integrated with the Google account and all the other Google services, probably due to the inability to defeat the popularity of Facebook. From then on, every user that uses another service from Google, like Gmail, Drive, or even Youtube (as Youtube is also owned by Google), started to have a Google+ account and a user profile. As a consequence, the number of active users on Google+ grew significantly, but most of the users are not aware of the G+ website, and the fraction of time users spend using this service, when compared to other social networks, is small.

Considering Google+ features, and besides user profile, Google+ created the concept of circles, a core feature in the social network that enables users to organize themselves into groups and lists for sharing the same interests. There is also the feature Hangouts, that includes free video conference calls and instant messaging. Though this might not be seen as a Google+ feature, as it is integrated with all Google services, Google tightly integrated Hangouts in the Google+ website so it can be mostly seen as a Google+ feature.

As Google is the top ranked website to Alexa.com, and Google+ is fully integrated with Google services, the Google+ social network is implicitly one of the most used social networking services.

5.2.2 Twitter

Twitter[2], created in 2006, rapidly gained popularity with more than 328 million monthly active users in August 2017. According to the company, its mission is to give everyone

[1] http://plus.google.com/
[2] http://www.twitter.com

the power to create and share ideas and information instantly, without barriers. Nevertheless, the concept can be easily described as a network where a user can share a simple message, which is immediately made public.

Twitter took advantage of the worldwide implemented Short Message Service (SMS), and promoted the idea of sharing simple day life events in order to stay connected with friends and family. Although this was the initial concept, Twitter has changed and is now allowing multimedia content, like photos and video.

One of the most significant differences of Twitter as a social network is that the relation between users in not necessarily reciprocal, i.e., in Twitter a user can follow another user without being followed back. Another distinctive characteristic of Twitter is that, by default, all messages are public. According to Alexa.com, Twitter is today the ninth most accessed website in the world.

Tweet

A *tweet* is any message posted to Twitter which may contain photos, videos, links and up to 140[1] characters of text. There is also the concept *retweet*, described as a tweet that you forward to your followers. Tweets can also include *mentions*, a user name started with the symbol "@", and *hashtags*, detailed in the following. An example of a tweet is shown in Figure 5.1.

Figure 5.1: An example of a tweet

Hashtag

Twitter provides the possibility of including a *hashtag*. A hashtag is a single word starting with the symbol "#", as represented in Figure 5.2. It is used to classify the content of a message and improve search capabilities. This can be particularly important considering the amount of data produced in the Twitter social network. Besides improving search capabilities, hashtags have been identified as having multiple

[1] Twitter announced in September 2017 that the limit will double to 280 characters

and relevant potentialities, like promoting the phenomenon described in Huang *et al.* (2010) as *micro-meme*, i.e. an idea, behavior, or style, that spreads from person to person within a culture (Merriam-Webster, 2004). By tagging a message with a trending topic hashtag, a user expands the audience of the message, compelling more users to express their feelings about the subject (Zappavigna, 2011). Although users with different languages tag tweets in different ways (Weerkamp *et al.*, 2011), e.g. German language users tag 25% of the produced tweets while Japanese language users tag only 4%. Kwak *et al.* (2010) presents a study that states that 15% of Twitter users have recently participated in trending topics.

Figure 5.2: Tweet and hashtag representation

The importance of the hashtag in Twitter is already identified in multiple applications, like bringing a wider audience into discussion (Johnson, 2009), spreading an idea (Tsur & Rappoport, 2012), get affiliated with a community (Yang *et al.*, 2012), or bringing together other Internet resources (Chang, 2010).

Regarding Twitter hashtags, and particularly hashtag recommendation, we have identified the study presented in Zangerle *et al.* (2011), where an approach for hashtag recommendation is introduced. This approach computes a similarity measure between tweets and uses a ranking system to recommend hashtags to new tweets. A different approach is proposed in O Ozdikis (2012), where an event detection method is described to cluster Twitter hashtags based on semantic similarities between the hashtags. Two methods for tweet vector generation are proposed, and their performance evaluated on clustering and event detection in comparison to word-based vector generation methods.

Twitter also implemented *cashtags*, a company's stock ticker symbol preceded by the symbol "$" (e.g. $TWTR for Twitter's cashtag), that allows financial searches. Though this feature was implemented in 2012, it is less used and almost unknown for most Twitter users.

5.3 Paramount Applications

The applications of social networks are endless. Considering machine learning applications, one can mention event detection (Doulamis *et al.*, 2016; Atefeh & Khreich, 2015; Cordeiro & Gama, 2016), information spreading (Doerr *et al.*, 2012; Lu *et al.*, 2016), community mining (Tantipathananandh & Berger-Wolf, 2011; Attea & Khoder, 2016), crowdsourcing (Treiber *et al.*, 2011) and sentiment analysis (O'Connor *et al.*, 2010; Pozzi *et al.*, 2016), among others.

One of the important achievements of our decade is the rapid pace at which we notice and propagate events. As information travels fast and worldwide, and everyone can use social networks to publish easily and fastly, the occurrence of an event, like a catastrophe as an earthquake, can be announced in seconds. But due to the amount of information propagated, it is not easy to identify trustworthy information about events, so event detection is mandatory.

This particular case of event detection, popularized through social networks, is also referred as citizen's journalism. Social networks are not only used to describe the occurring events, but also to ask for help and to advise others in similar situations. They are also used to bring people together in rebellions, forcing governments to react. Another worth mentioning capability is to promote transparency, participation, and collaboration, as many organizations started to publish consolidated account reports. These movements can also appear naturally, as people easily engage together for causes considered significant, specially within small communities. This can obviously endorse noteworthy changes in news spreading, community engagement, disaster response, and community safety.

Considering communities, social networks were also responsible for another challenge concerning graph mining, the community mining. Humans tend to aggregate themselves in communities based on common interests, an idea of a "reciprocity" phenomenon called *homophily*. This phenomenon, identified by McPherson *et al.* (2001), structures network ties of every type, including marriage, friendship, work, advice, support, information transfer, exchange, co-membership, and other types of relationship, which result in people's personal networks becoming homogeneous with regard to many sociodemographic, behavioral, and intrapersonal characteristics. This is a particularly

important knowledge to mine as it is useful in many domains, like recommendation systems, advertising, sociology, or epidemiology. Community mining is sometimes referred in literature as a particular graph matching problem.

Still, graph mining in social media do not only include community mining. In recent years, the problem of graph mining, also called graph-based knowledge discovery, has become an active research area in the field of data mining and aims to find novel knowledge in graph data structures, like repetitive patterns or substructures. The World Wide Web and the increasing importance of social networks have played a significant role, not only because they are responsible for new arising paradigms that are important to mine, like new communication models already referred, but also because for the first time there is an easily accessible large scale graph with tons of data.

Sentiment Analysis is another relevant machine learning field that gained significance in social networks. It concerns with the perception of sentiments in data, particularly in text data. The importance of sentiment analysis is explained by the impact it can have for instance in the market, as an enterprise is fully interested in perceiving the acceptance a product can have with their customers. This can be done by analysing what is being written about the product in social media, i.e., if customers are committing to a product or rejecting some of its characteristics. However, considering the urge of social media, its information spreading ability, it is not only mandatory to predict the acceptance of a product, as it is important to analyse the political impact of decisions being taken by governments, and thus sentiment analysis is one of the paradigms of social networks applications.

We have pinpointed some of the paramount applications of social networks. Being aware of the potential of social networks, we are conscious that others could also be mentioned. Nevertheless we tried to reach a trade-off between the outline of this thesis and the most relevant applications one can mention considering its scope.

5.4 Current Research

Social networks are being widely studied in many fields, specially due to their multiple applications referred in the previous section. Modern challenges in social networks involve not only computer science matters but also social, political, business, and economical sciences.

In this section, we will detail the research paths that are being taken considering social networks and the relevant challenges in the field. It is our purpose to present the efforts being done, but also the major challenges that still persist, laying a foundation to settle our contributions in the field.

5.4.1 Research Paths

There is a growing interest in social networks concerning machine learning tasks. The research paths are multiple, in line with all the possible applications. It is infeasible to present all the research paths, but we will try to mention those related with the scope of this thesis.

Considering event detection in Twitter, a machine learning task where a lot of effort is being put through, one must mention a survey presented in Atefeh & Khreich (2015). The authors classify the presented techniques according to the event type, detection task, and detection method, and discuss commonly used features. They also highlight the need for public benchmarks to evaluate the performance of different detection approaches and various features, which is in line with the challenges that will be presented in the next section. A more recent survey on event detection in social networks is presented in Cordeiro & Gama (2016). In this survey, besides Twitter, other social networks are considered, and an overview of the common detection methods is presented, along with a taxonomy of event detection systems, taking into account the type of detection (supervised, unsupervised, and hybrid) and the type of the event that the system tries to detect (specified or unspecified event). Recent works in event detection in Twitter include Doulamis *et al.* (2016), Nguyen & Jung (2017) and Alsaedi *et al.* (2017).

A lot of research is also being done comprising sentiment analysis in social networks, and particularly in Twitter. In Agarwal *et al.* (2011), authors examine sentiment analysis on Twitter data. According to the authors, they have tentatively concluded that sentiment analysis for Twitter data is not that different from sentiment analysis for other genres. A survey regarding sentiment analysis in Twitter is presented in Giachanou & Crestani (2016), where an overview of the topic is presented, by investigating and briefly describing the algorithms that have been proposed for sentiment analysis in Twitter. A broader and extensive survey is documented in Pozzi *et al.* (2016).

Regarding crowdsourcing and social networks, we have already presented some research works in Section 3.5. In Costa *et al.* (2013c) we have proposed the use of meta-classes to boost the performance of Twitter messages' classification. This preliminary study, that will be further presented in Section 5.6, shows the possibility of evaluating message content in order to predict hashtags. Regarding Twitter hashtags, and particularly hashtag recommendation, we have also identified the study presented in Zangerle *et al.* (2011), where an approach for hashtag recommendation is introduced. This approach computes a similarity measure between tweets and uses a ranking system to recommend hashtags to new tweets. In Duan *et al.* (2012) the use of hashtags to classify Twitter messages is done by clustering similar tweets in a graph-based collective classification strategy. The presented results are promising, despite the fact that this is not an adaptive strategy.

A different approach is proposed in O Ozdikis (2012), where an event detection method is described to cluster Twitter hashtags based on semantic similarities between the hashtags. This work is in line with our previously referred work, except for the fact that we have based the similarities on the message content rather than being based on semantic hashtag similarities.

Another research path is learning in the presence of concept drift in dynamic scenarios like social networks, and particularly in Twitter, where important information can be mined, but this will be further described in Chapter 7. Despite the potential of these techniques, major challenges arise, which will be further described.

5.4.2 Challenges

One of the major challenges regarding machine learning tasks in social networks is the inability to find the adequate benchmark dataset. Most social networks platforms provide an Application Programming Interface (API), a collection of routines, protocols, and tools in which one can access the social network in a programmatic way. The major advantage of those API is to be able to implement software to interact with the social network and thus, for instance, obtain data or publish documents. Though this is relatively easy to implement, when there is a need for a labelled dataset, like, to classify social networks posts concerning sentiment analysis, or event detection, major problems arise, as data is abundant, but not labelled. One of the possible solutions, used by some authors, like in Agarwal *et al.* (2011) and in Finin *et al.* (2010), is to

manually annotate a social network dataset. Despite this being a solution, it can be costly, specially concerning the amount of available data in social networks. Another strategy is used by Bifet & Frank (2010), where emotions are used to classify a Twitter dataset for sentiment analysis. Considering we want to be able to cope with drift, there is a special challenge when we want to represent dynamics in a social network dataset.

Social networks are, in their essence, a time series, and posts are always timestamped, but drift patterns are not identified, i.e., even though there is no doubt that in social networks drifts are present, we do not know where they are, and what are their characteristics. One of the aims of this thesis is to classify text-based messages in a dynamic environment. Both challenges, on one hand those related with text classification, and on the other hand those related with dynamic environments are usually seen separately. Regarding datasets to text classification, one must mention the Reuters-21578 Text Categorization Test collection [1] and the Twenty Newsgroups Dataset[2]. But, despite the importance of these datasets as text classification datasets, they do not represent a dynamic environment, and their aim is exclusively related with text classification, not comprising the effect that time can have in evolving documents and their classification.

Considering dynamic environments, popular datasets include the Streaming Ensemble Algorithm (SEA) dataset and the Gaussian dataset. The SEA dataset, presented in Street & Kim (2001), was artificially generated with 60,000 random points in a three-dimensional feature space. All three features have values between 0 and 10, but only the first two features are relevant. Those points were then split into four blocks in order to create 4 different concepts, and a threshold was set so the sum of both features can define if the example belongs to a concept or not. The Gaussian dataset, presented in Elwell & Polikar (2011), features multi-class data, each drawn from a Gaussian distribution. Each class experiences gradual but independent drift, with class means and variances changing according to parametric equations. Differently from other drift datasets, in the Gaussian dataset, class addition and removal are supported. They are widely used, but they are not text-based datasets and, obviously, they can not be used for text classification purposes. We have identified some drift datasets that are text-

[1] http://www.davidlewis.com/resources/testcollections/reuters21578/
[2] https://archive.ics.uci.edu/ml/datasets/Twenty+Newsgroups

based, but we have also identified some limitations, and they will be further discussed in Section 6.4.

Besides datasets, there are some frameworks that can generate datasets with drift. One of the most well-known frameworks is Massive Online Analysis[1] (MOA). MOA is open source and used for data stream mining. It includes not only a collection of machine learning algorithms for classification, regression, or clustering, but also evaluation tools. It is based on the Waikato Environment for Knowledge Analysis (WEKA) project and users can easily implement their own algorithms in MOA for testing purposes. Although it has a lot of potential, it also does not include text-based features.

In addition to the above mentioned problem, the particular semantic of social networks can also pose challenges, as it can be difficult to mine in those environments, specially in social networks that use short messages like Twitter, where users are limited to only 140 characters per post. It is also important to refer that commonly social media messages are written informally (with no controlled vocabulary), without predefined rules or a well defined structure, and in mobile devices. These characteristics, along with the massification of Short Message Service (SMS), and other equivalent short messaging services, are responsible for the arising of a particular semantic, where users tend to shorten the length of the words to be as much informative as possible in the space available for a single message, like making use of acronyms. As a result, traditional text mining techniques are not suitable, not only because of the short length of tweets but also because of the large number of spelling and grammatical errors, and the frequent use of informal and mixed language. There are also large amounts of meaningless messages and polluted content, which negatively affect the classification performance (Atefeh & Khreich, 2015). One must also mention that some words, specific in the Twitter context, like 'rt' (from retweet), can also be considered stopwords, and thus preprocessing methods adapted to social networks can also be a matter of study.

Some characteristics of social networks can also be difficult to deal with. Shneiderman *et al.* (2011) identified four important characteristics: *(1) polarized discussions*, as users tend to selectively view only material aligned with their world view, *(2) reduced credibility of online resources*, as rumours and mislead information spread fast

[1] http://moa.cms.waikato.ac.nz/

in social networks, and are not filtered or verified by traditional journalistic means, *(3) possible distraction from deep reflection*, as individuals tend to respond to frequent interruptions, privacy, security issues, identity theft, online bullying and *(4) disclosure of potential or embarrassing personal information*. All the above mentioned can pose questions to validate what is being written, for instance, if a user bias through an aspect can affect the document itself if it is used for classification purposes.

Another challenging issue is privacy. Millions of photos are posted online each day in Twitter, Blogger, or Facebook, containing neighbourhoods, houses, families, and children. Besides online publishing contents, people are invited to tag the photo with more information, such as who is in the photo or where it was taken. In Facebook people can tag someone else in a photo that it is not even theirs, which means someone can tag us in a photo we did not known it existed. Although it can be argued that we can easily untag our name in a photo or even ask Facebook to delete the photo, it is impossible to always be aware if (and when) content about us is published online. Content posted online remains so forever and as a consequence all participants should be conscious and aware of every share. As a consequence, users tend to perceive the lack of privacy as a frightening issue, which is perfectly acceptable, and one can mention a simple exercise to be conscious of how scary this can be by trying to make a timeline of someone's life using online information. By searching for a name in Google, trying to find it in Twitter or Facebook, finding out addresses, dates, personal achievements and friend and family connections, one can easily perceive most daily routines of a user. In order to deal with all these privacy issues, and being aware each day that this is a relevant matter for its own users, constant changes in platforms, particularly in Facebook, are being implemented. Although this might not be seen as a challenging issue regarding the scope of this thesis, the sense of lack of privacy, along with the constrains that are being implemented by social network platforms, can lead to difficulties regarding data access, which can compromise future studies regarding the learning in these environments.

Even though privacy can be a concern for social networks' users, there is a growing interest in identifying social behaviours that permit information to spread, as this important characteristic of social networks can be profitable. Regarding this challenge, two works must be mentioned: Doerr *et al.* (2012) and Nekovee *et al.* (2007).

It is not the intent of this thesis to convey all the above mentioned challenges regarding social networks. Nevertheless, we will try to address some of them, making a

contribution in the field, specially regarding the lack of benchmarks for text classification in the social network dynamic environment and proposing a solution to cope with learning in this environment. We propose a two-fold contribution regarding the above mentioned challenge, not only a text-based dataset with drift, but also a framework that can generate text-based datasets with drift. In the next chapters we will detail both contributions and in the next section we will present a Twitter classification strategy. The aim of this approach is to be able to avoid the need for manual annotation in the Twitter environment, by using the hashtag to classify the Twitter messages.

5.5 Twitter Classification

Given the widespread use of social networks, research efforts to retrieve information using tagging from social networks communications have increased. Particularly in Twitter, hashtags are widely used to define a shared context for events and topics, as mentioned in Section 5.2.2, and have multiple and relevant potentialities, like improving search capabilities and information spreading (Costa *et al.*, 2013c). While the use of hashtags is a common practice, the hashtags are often introduced by the user without any particular rule and resulting in a natural bias.

Considering the importance of the hashtag in Twitter, it is relevant to study the possibility of evaluating message contents in order to predict its hashtag. If we can classify a message based on a set of hashtags, we can, for instance, promote a product or recommend similar content to the users' preferences. This is also particularly important for machine learning purposes, considering that it is infeasible to manually annotate the amount of data collected from social networks, as detailed in Section 5.4.2.

In this section, we propose an approach to classify Twitter messages based on their hashtag. We will start by defining the Twitter classification problem, and then describe how have we collected Twitter data in order to test and evaluate our approach. We will then present the concept meta-hashtag, its uses and how can meta-hashtags improve classification performance.

5.5.1 Definition

The classification of Twitter messages can be described as a multi-class problem that can be cast as a time series of tweets. It consists of a continuous sequence of instances,

in this case Twitter messages, represented as $\mathcal{D} = \{d_1, \ldots, d_t\}$, where d_1 is the first occurring instance and d_t the latest. Each instance occurs at a time, not necessarily in equally spaced time intervals, and is characterized by a set of features, usually words, $\mathcal{W} = \{w_1, w_2, \ldots, w_{|\mathcal{W}|}\}$. Consequently, the instance d_i is represented by the feature vector $\{w_{i1}, w_{i2}, \ldots, w_{i|\mathcal{W}|}\}$.

If d_i is a labelled instance it can be represented by the pair (d_i, y_i), where $y_i \in \mathcal{Y} = \{y_1, y_2, \ldots, y_{|\mathcal{Y}|}\}$ is the class label for instance d_i.

Our classification strategy will be using the Twitter message hashtag to label the content of the message, which means that y_i represents the hashtag that labels the Twitter message d_i.

Notwithstanding being a multi-class problem in its essence, it can be decomposed in multiple binary tasks in a one-against-all binary classification strategy. In this case, a classifier h^t is composed by $|Y|$ binary classifiers.

5.5.2 Dataset

To evaluate and validate our strategy, and considering the lack of a labelled dataset with the needed characteristics, we have built our own dataset. The dataset was constructed by requesting public tweets to the Twitter API[1]. We have collected more than 230,000 messages during four days, since 24 November 2012 to 27 November 2012, and, considering the worldwide usage of Twitter, tweets were only considered if the user language was defined as English. All the messages that did not have at least one hashtag were discarded, as the hashtags are assumed as the message classification. Finally, tweets containing no message content besides hashtags were also discarded and all the hashtags are removed from remaining tweets. From the 230,000 collected messages, we reach 10,000 tweets that have a body part and at least one hashtag.

As users are able to define their own hashtags, which increases the number of classification classes, only the ten most used hashtags were considered. In Figure 5.3 we show the word cloud representation of the most frequent hashtags present in the dataset.

The tweets were then split into two equal and disjoint sets: training and test. The data from the training set is used to select learning models, and the data from

[1] https://dev.Twitter.com/

Figure 5.3: Most frequent hashtags present in the dataset

	Precision	Recall	F_1
#NP	44.16%	48.92%	46.42%
#TEAMFOLLOWBACK	37.50%	48.00%	42.11%
#OOMF	36.86%	72.50%	48.88%
#REPLACESONGTITLEWITHTOMCATS	96.34%	94.05%	95.18%
#WHENIWASLITTLE	36.79%	48.75%	41.94%
#NOWPLAYING	27.27%	47.83%	34.74%
#SEX	80.00%	74.07%	76.92%
#PORN	88.37%	67.86%	76.77%
#NF	21.65%	87.72%	34.72%
#KCA	100.00%	43.18%	60.32%

Table 5.1: Comparative results considering precision, recall and F_1

the testing set to evaluate performance. Preprocessing methods were applied, namely stopword removal and stemming.

5.5.3 Experimental Results and Analysis

The proposed Twitter classification approach was implemented using Support Vector Machines as the classifier. Table 5.1 summarizes the performance results obtained in classifying the ten most used hashtags in the dataset. The results are ordered by the classification of the most frequent hashtag, #NP, to the least frequent one, #KCA.

Considering the main goal of the proposed approach, which is to evaluate the suitability of the Twitter message content to predict its hashtag, it is possible to see that

the hashtag #REPLACESONGTITLEWITHTOMCATS was almost correctly classified in all the examples given, with an F_1 score of 95.15%. The explanation is that this particular hashtag was a trending topic that became popular through the concerted effort of many users, a recurrent phenomenon in Twitter. These users not only replicated the hashtag but also the content, or form, of the tweet message, thus enhancing the classification. It is also important to note that the obtained results in the classification of this hashtag indicate that the content of a Twitter can be informative of its hashtag. Besides the excellent performance in classifying the above cited hashtag, the classification of a Twitter message is difficult. There are a few explanations for this:

1. Twitter is a multi-language platform, so users can write the same information in different languages and use the same hashtag. Different languages have different structures, different words for the same information, and thus can inhibit the correct classification of the Twitter message. We have tried to minimize this problem by requesting tweets from users who have set their Twitter language to English, but there are multiple non-native English users that set their preferences to English and still write in their own language.

2. Twitter messages tend to be written quickly and without special rules.

3. With only 140 characters it is not easy to be informative, i.e., without a predefined rule, or without a concerted effort, it is hard in free text to compose a message informative enough with just a few words (including hashtags).

4. People can post the same kind of information and choose different hashtags, e.g., posting the song they are playing at the moment: they can use the #NP or #NOWPLAYING, or even #IAMPLAYING. Although referring the same content, the hashtags are different for classification purposes.

5. With the use of different hashtags when posting the same information, in terms of classification purposes, we are saying that the same content is classified in different ways, and thus we are inhibiting the classifier to generalize both classes.

We then propose an approach to deal with the bias resulting from the freely user-defined hashtags, by defining semantic meta-hashtags to identify clusters of similar messages, in order to improve their classification. In the next section we will detail the proposed approach, presenting the obtained results, and the subsequent analysis.

5.6 The Use of Meta-hashtags in Twitter Classification

In this section we describe the proposed approach to define meta-hashtags and to use them in a classification application to improve the overall classification score. Our approach is two-fold, resulting in two final models, the baseline model, that considers the user-defined hashtags, and the meta-hashtags model, that considers the clusters of similar messages grouped by a single meta-hashtag. In Figure 5.4 we depict the proposed framework.

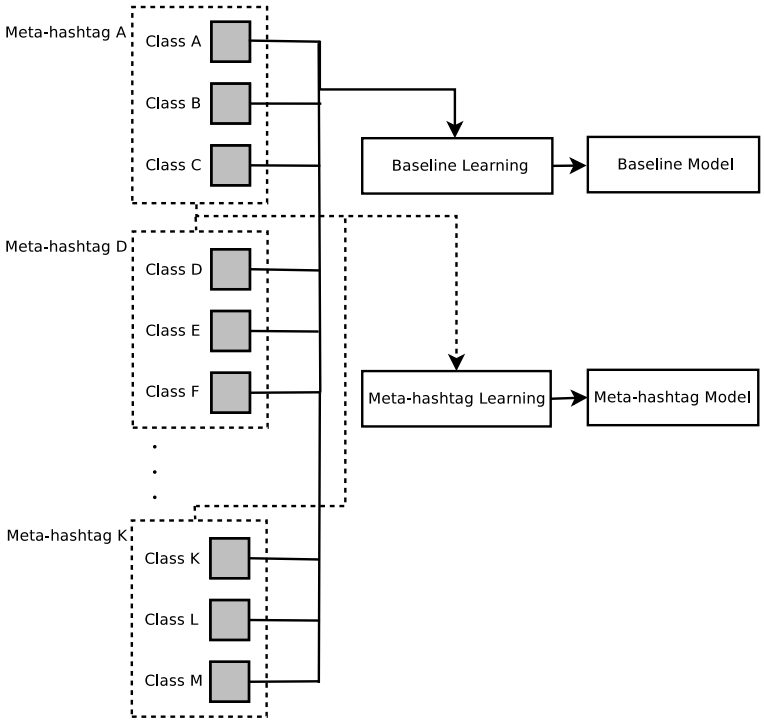

Figure 5.4: Meta-hashtags Approach

The baseline model is constructed and trained with labelled examples that use the user-defined hashtags as a one-against-all two-class problem. In the meta-hashtags model, semantic meta-hashtags were heuristically defined, by clustering similar hashtags in a meta class. Related classes are then relabelled according to the new defined meta-hashtags and a similar training process occurs in order to construct the new proposed model.

The underpinning idea behind the use of meta-hashtags is to combine the class

label of similar messages in order to mitigate the effects of the bias introduced by freely user-defined hashtags, and thus improving the overall classification of Twitter messages according to their hashtags.

5.6.1 Dataset

We have used the dataset previously collected and described in Section 5.5.2, and have used the crowdsourcing platform *http://tagdef.com/* to discover the hashtag meaning and the related hashtags. Considering most used ones, we have chosen those which have at least two or more related hashtags, so a meta-hashtag class may empirically be defined. A total number of 15 hashtags were found to match this presumption and a total number of 1,230 tweets were considered as being labelled and suited for classification purposes. The individual hashtags were semantically clustered in 5 meta-hashtag classes, as depicted in Table 5.2.

The tweets were then split into two equally sized and disjoint sets: training and testing. The training dataset is used to build classification learning models, and the testing dataset to evaluate performance.

Table 5.2 describes the positive documents of each class and the corresponding meta-class of the dataset. As can be observed from the table, there is an heterogeneous distribution of hashtags in the dataset. For example, class TEAMFOLLOWBACK has 274 documents, while class NOWFOLLOWING has only 17 documents. The amount of positive documents in the training and testing datasets is balanced because it is obtained by the equal split of training and test sets.

5.6.2 Experimental Results and Analysis

We evaluate the performance obtained on the Twitter dataset using the two approaches previously defined, namely, the baseline approach and the meta-hashtag approach. Table 5.3 summarises the performance results obtained by each approach when classifying both dataset versions: the initial one considering the 15 initial hashtags, and the one considering the meta-hashtags.

Analysing the table we can observe that the use of a meta-hashtag outperforms the overall classification of the initial hashtags when they are considered individually. For example, the class NP has an F_1 score of 49.50% and the class NOWPLAYING 28.05%, both classified by the baseline model. However, with the use of meta-hashtags,

	Training	Testing
NP	138	139
NOWPLAYING	72	68
meta-hashtag NP	**209**	**207**
SEX	70	54
PORN	65	56
XXX	23	19
HOT	14	6
meta-hashtag SEX	**104**	**76**
JOB	32	36
JOBS	41	37
meta-hashtag JOB	**59**	**61**
NW	32	32
NOWWATCHING	7	4
meta-hashtag NW	**39**	**36**
TEAMFOLLOW	17	18
TEAMFOLLOWBACK	126	148
FOLLOWBACK	14	24
NF	58	57
NOWFOLLOWING	7	10
meta-hashtag NF	**207**	**238**

Table 5.2: Amount of positive documents in the training and testing phases

the new proposed model presents an F_1 score for the meta-hashtag NP class of 64.45% F_1. This might be related to the fact that the content being classified in the meta-hashtag version of the dataset is more diverse than the one from the initial dataset, thus misleading the classifier. These improvements on the results obtained may also be observed in other cases, like in class JOB with F_1 of 59.26% and class JOBS with 64.00%, while the corresponding meta-hashtag JOB has 71.17%.

With the use of a meta-hashtag we unify the labelling process by grouping similar messages and placing them in the same classification class, thus boosting the performance of the overall classifier. This analysis is in line with the work presented in O Ozdikis (2012), where a pseudo-meta-hashtag approach was presented to be beneficial in a clustering problem, even though they have clustered based on hashtag similarities rather than message content similarities.

	Baseline Model			Meta-hashtags Model		
	Precision	Recall	F_1	Precision	Recall	F_1
NP	45.73%	53.96%	49.50%	35.44%	92.81%	51.29%
NOWPLAYING	23.96%	33.82%	28.05%	15.11%	80.88%	25.46%
meta-hashtag NP				**50.55%**	**88.89%**	**64.45%**
SEX	70.18%	74.07%	72.07%	38.69%	98.15%	55.50%
PORN	80.00%	71.43%	75.47%	40.88%	100.00%	58.03%
XXX	21.05%	21.05%	21.05%	11.45%	100.00%	20.54%
HOT	7.14%	16.67%	10.00%	3.61%	100.00%	6.98%
meta-hashtag SEX				**69.23%**	**94.74%**	**80.00%**
JOB	53.33%	66.67%	59.26%	34.31%	97.22%	50.72%
JOBS	50.79%	86.49%	64.00%	33.33%	91.89%	48.92%
meta-hashtag JOB				**56.86%**	**95.08%**	**71.17%**
NW	17.65%	9.38%	12.24%	15.38%	12.50%	13.79%
NOWWATCHING	0.00%	0.00%	-	3.85%	25.00%	6.67%
meta-hashtag NW				**19.23%**	**13.89%**	**16.13%**
TEAMFOLLOW	100.00%	100.00%	100.00%	26.09%	100.00%	41.38%
TEAMFOLLOWBACK	40.80%	55.41%	46.99%	32.91%	87.84%	47.88%
FOLLOWBACK	0.00%	0.00%	-	5.57%	91.67%	10.50%
NF	25.00%	5.26%	8.70%	13.92%	96.49%	24.34%
NOWFOLLOWING	-	0.00%	-	2.53%	100.00%	4.94%
meta-hashtag NF				**54.94%**	**91.18%**	**68.56%**

Table 5.3: Comparative results considering precision, recall and F_1

Other noteworthy results are obtained when using the meta-hashtag model to classify the initial classes. Although the F_1 measure decreases considering the baseline model, the recall increases in a higher proportion. As an example, the class XXX classified by the baseline model presents a precision and a recall of 21.05%. Classified by the meta-hashtag model, the precision falls to 11.45% and the recall raises up to 100.00%, which means that precision proportionally decreases less than the increase of recall. This is due to the false positive increase being less than the increase of true positives. The increase of false positives, and thus the decrease of the F_1 measure, was expected, as we mislead the classifier by training it with the meta-hashtag examples, which means we used as positive examples not only the initial class messages, but also the related messages that belong to semantically similar hashtags.

In the baseline model, classes like NOWFOLLOWING or FOLLOWBACK have no F_1 measure. This occurs because the classifier did not identify any true positive document, probably due to the lack of information in the training phase, so precision and recall are 0.00% and F_1 can not be calculated. In these classes the use of the meta-hashtags approach, more than increasing the classifier performance, permits the identification of these classes' documents.

The preliminary results are very promising. It is possible to observe that the proposed approach outperforms the F_1 measure of each initial class included in its composition, with the exception of the initial class TEAMFOLLOW, that is already correctly classified in the initial approach. It is also important to note the overall improvement of the recall metric when using the meta-hashtag model to classify the initial classes. This improvement sustains the use of meta-hashtags and makes it possible to infer that clustering similar messages can improve the classification performance.

5.7 Conclusion

Social networks are part of the daily routine of millions of users. They can be considered as entertainment tools, where people express personal feelings (and thoughts) or make a diary style journal. On the other side, they can also be a relevant source of information as users share not only daily status, but rapidly propagate news and events that occur worldwide. Besides, social networks are also used for professional advertisement, promoting services, or earning income. Moreover, they are also a mean

for new business models as people can provide services, promote themselves, and easily interact with costumers. The presence of companies and organizations in social networks is not only to gain visibility and self promotion, but also for different new areas as market sensing, since people often share information concerning products and organizations. The applications are endless, like spam, email filtering, intrusion detection, recommendation systems, event detection, or search capabilities improvement.

Considering their potential, specially in information spread, they have become an important research field. Even though our main interest is computer science, and particularly machine learning, one must mention that modern challenges in social networks involve not only computer science issues, but also social, political, business, and economical sciences.

In this chapter we have presented a background on social networks, describing the most popular, along with their history and main features, and detailing Twitter and its characteristics, as it is the case study we have used in the context of this thesis. We have then presented the scope of paramount applications regarding social networks, along with the current research that has been done. We have detailed some of the challenging issues in the field and presented two major contributions: a study regarding the use of the hashtag for Twitter classification, and the use of the concept meta-hashtags for improving the classification of Twitter messages.

In the next chapter, we will present further contributions regarding learning in dynamic environments.

Chapter 6

Learning in Dynamic Social Environments

In this chapter we focus on learning strategies to apply in dynamic social environments. We start by presenting the Drift Oriented Tool System (DOTS), a framework that allows the definition and generation of text-based datasets for dynamic environments. Three learning models to tackle such environments are then proposed, namely, the time-window model, the incremental model, and the ensemble model. We continue by detailing a dataset we are going to use to test and evaluate learning strategies in dynamic environments, more precisely in the social networks' environment, and finally, we study the effect of longstanding messages in dynamic environments, proposing a batch learning model to cope with the infeasibility of storing all the previously acquired examples in problems like social networks' data streams.

6.1 Introduction

Current challenges in machine learning include dealing with temporal data streams, drift and non-stationary scenarios, often with text data, whether in social networks or in business systems. This dynamic nature tends to limit the performance of traditional static learning models and dynamic learning strategies must be put forward. Considering the particular case of social networks, and their dynamic nature and potential as information networks, there is an ever-growing interest in the extraction of complex information used for trend detection, promoting services, or market sensing, and it is

imperative to find learning strategies able to learn in these environments and cope with their dynamic nature.

Learning in social networks is particularly difficult, not only because of its dynamic nature, but also due to the amount of data produced. In social networks, documents, usually known as posts, are organized from more recent to older and users tend to perceive the newly posted material as more relevant. Considering the posted material by millions of users, everyday, we realize that time plays an important role, by easily and fast out-dating information. Moreover, in this extremely dynamic environment, concepts appear and reappear, as users concentrate their focus on newly occurring events, forgetting old ones forever or during an unpredictable amount of time. For instance, during a terrorist attack there is a sudden burst of related messages during a few days that will probably fade away as time passes by, but might reappear if new information surfaces, or if a year or a decade has passed and the event is again mentioned.

To learn in this kind of environment, and considering how infeasible it might be to use all the data produced, it is essential to understand how informative past events can impact current learning models. This influences how long it is relevant to store previously seen information, to reduce the computational burden.

However, in order to test the effectiveness of learning models that aim to cope with different drift patterns in social networks, adequate benchmarks are needed. This is another challenging issue in dynamic social environments, as it is difficult to find benchmarks that relate learning in dynamic environments to social networks.

6.2 Drift Oriented Tool System

The usual challenge for machine learning approaches is to build models that can perform well in classifying new data in production settings. Research efforts are usually put forward in proposing, implementing, and testing innovative algorithms and techniques, but to build drift aware datasets with blended temporal distributions is also a challenging task.

It is not straightforward to find acceptable benchmarks in dynamic environments and for that reason we developed the Drift Oriented Tool System (DOTS), a framework that allows for the definition and generation of text-based datasets and can be

used to simulate a set of different drift patterns with a temporal basis. The datasets obtained can then be used to evaluate and validate learning strategies used in dynamic environments, including preprocessing strategies.

DOTS is a simple-to-use freeware application with a friendly interface as shown in Figure 6.1. It can be downloaded at `http://dotspt.sourceforge.net/`.

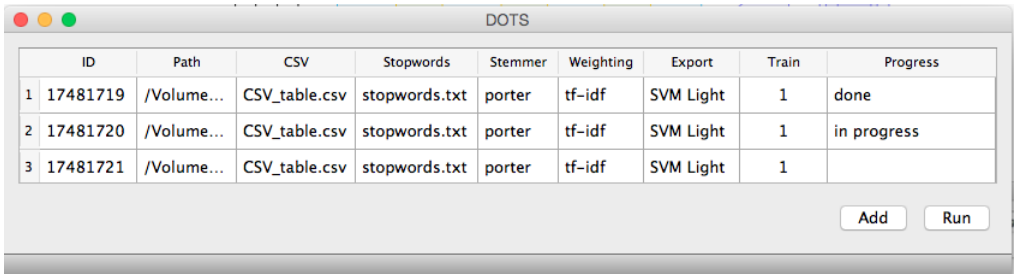

Figure 6.1: The DOTS interface

The main purpose of the DOTS framework is to represent drift patterns in a text-based dataset. Therefore, the framework input is a set of text document files, each representing a class, and a frequency table representing the drift patterns. Figure 6.2 depicts the framework. There are three processing phases that will be further described: *INDRI index*, *preprocessing* and *data generation*. The initial phase of DOTS includes building an INDRI index, provided via the INDRI API, from the Lemur Project[1]. By building an INDRI index we extend the potentialities of DOTS to those provided by the INDRI API, like a text search engine and a rich structured query language.

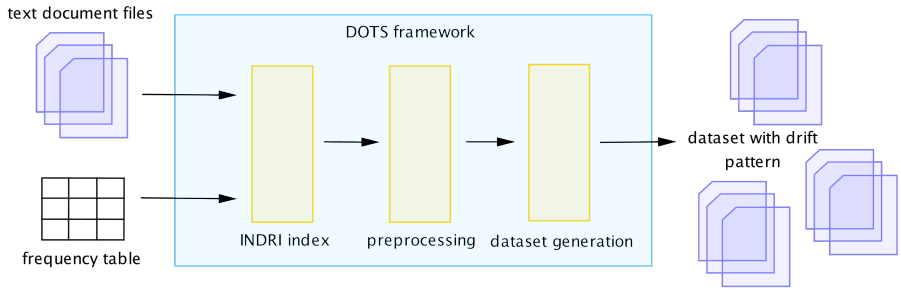

Figure 6.2: The DOTS framework

The DOTS framework represents each document with the *bag of words* document representation.

[1] `http://www.lemurproject.org/`

Two problems arise when a vector with one element for each term occurring in the whole collection is used to represent a document: space high-dimensionality and overfitting, as previously mentioned in Section 2.3.2. To tackle both problems, preprocessing methods were also integrated in the DOTS framework and are the second phase of the processing. Those methods are part of the INDRI API and aim at reducing the size of the document representation and prevent misleading classification. Besides stopword removal, DOTS also permits stemming. Two important stemming algorithms for the English language were included: the Porter algorithm (Willett, 2006) and the Krovetz algorithm (Krovetz, 1993).

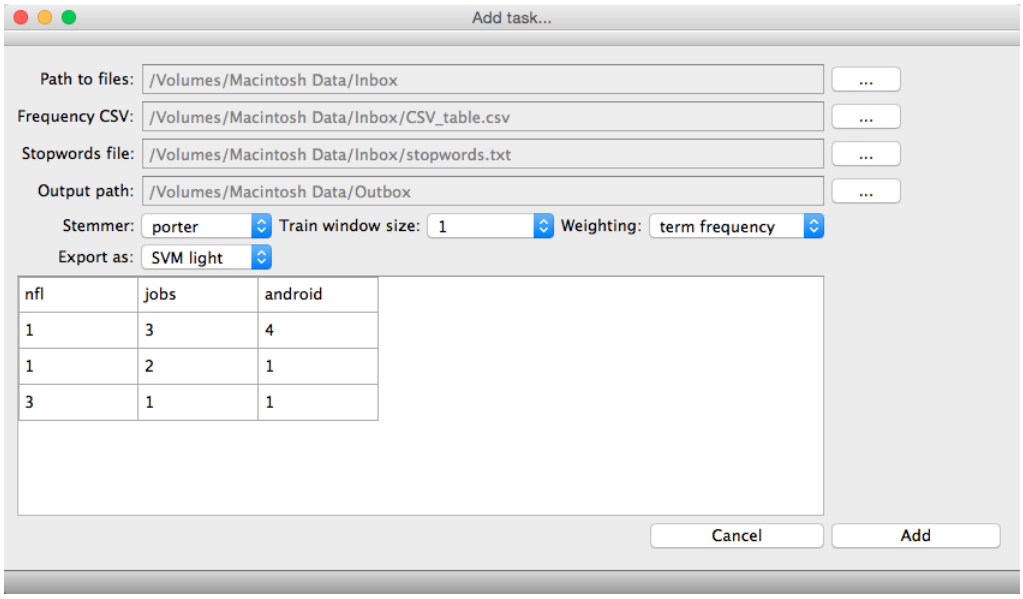

Figure 6.3: DOTS: add task feature

It is also possible to define the weighting scheme used to represent each word of a document, that is, the weight of each feature of a document. Two weighting schemes were defined, namely term frequency (*tf*) and term frequency-inverse document frequency (*tf-idf*). Considering the defined input, DOTS creates a word index that allows users to use different strategies of filtering and data analysis.

A major characteristic of DOTS is the possibility of defining the exact time, more precisely the exact time-window, where each document appears, being thus possible to define time drifts. This is done by the frequency table that is also an input of the DOTS framework (see Figure 6.2). DOTS takes no regard of the real counterpart dimension

of the time-window, since it constitutes an abstract realization of an amount of time (in social networks streams it can represent seconds, but in other applications it can represent hours, or even days). The main idea is to use the frequency to reproduce artificial drifts.

DOTS output is thus a set of datasets, including the defined strategies related to vector space model, preprocessing strategies, feature representation, and document division in time-windows.

Three output formats were implemented: the Comma-Separated Values (CSV) file format, the Attribute-Relation File Format (ARFF), widely used in WEKA software, and the SVMLight file format. The possibility of using the concept of training window size, which will be further described in Section 6.5, is also possible, as users can define for each time-window the amount of previous data that should also be considered. By defining different training window sizes one can represent the memory properties of the training models, because it mimics a storage mechanism.

6.2.1 Specific Features of DOTS

The input of the DOTS framework is two-fold, as seen in Figure 6.2, and is composed of text documents and a frequency table. Each text document file represents the documents of the same class, and the frequency table is used to define the drift patterns of the scenario. The frequency table must be in the CSV format, and each row corresponds to a time instance. It is not important if a time instance represents a minute, an hour, or a day, but it is assumed that all of them correspond to the same amount of time. The first row contains the names of the classes, and each cell of all the other rows contain the number of documents of the given class that occur in a given time instance.

Consider Figure 6.3 that represents a task to be added to the framework. By using as input a frequency table as in the example given, we represent three classes: `nfl`, `jobs` and `android`, and 3 time-windows. As depicted in this example, in the first time-window there is 1 document of the class `nfl`, 3 from the class `jobs` and 4 of the class `android`.

Additional parameters can also be defined, like a stopwords file and a stemmer algorithm. The stopwords file is a text file containing stopwords that will not be considered in the documents' representation, and the stemmer algorithm will be used

to reduce the document inflected words to their root form. Two stemmer algorithms were implemented: *porter* and *korvetz*.

It is also possible to define multiple training window sizes, multiple weighting schemes, and multiple export file formats. The training window size defines in each time-window how many previous time-windows will be considered for training purposes, as this can be important for testing learning models with memory capabilities. For instance, to perceive for how long it is relevant to keep previously gathered information and how that can affect the learning model capabilities. The weighting scheme will be used to define the document representation weighting scheme, like *term-frequency* or *tf-idf*. By exporting in multiple file formats, DOTS permits the creation of datasets that can be used in different classification frameworks, like *SVMLight* and *WEKA*.

As it is often relevant to define various testing scenarios, DOTS also permits adding tasks using INI files. INI files are structured files with `‘‘key=value’’` pairs, which will allow for the definition of multiple tasks at once. A complete tutorial about DOTS can be downloaded at `http://dotspt.sourceforge.net/`.

The DOTS framework attempts to fill an existing gap in machine learning research for text applications, by making it possible to generate benchmark datasets with thoroughly controlled drift characteristics. We will now present an example of the potential use of the framework, with Twitter stream as case study.

6.2.2 Using DOTS in Twitter Classification

There are few works regarding the learning process in the occurrence of drift in the particular field of social networks, and little is known about the types of drift that can occur. Hence, the importance of having a dataset that simulates different types of drift in Twitter in order to evaluate and validate learning strategies in this scenario. To accomplish this goal we used DOTS to artificially timestamp Twitter messages in order to induce different types of drift with controlled features. The aim of this thesis is to identify the learning characteristics needed to deal with those dynamic features, and thus define the best tailored learning mechanisms.

The drifts we intend to represent are based on the four different major types proposed in Žliobaitė (2010), namely (i) *sudden*, (ii) *gradual*, (iii) *incremental*, and (iv) *reoccurring*. Thus, we have made use of these four types of drift and have included a

#syrisa	#airasia	#isis	#android	#sex
0	100	30	0	40
0	100	60	200	40
1000	0	90	200	40
1000	0	120	0	40
1000	200	150	0	40
0	200	180	200	40
0	0	210	200	40
0	0	240	0	40
0	300	270	0	40

Table 6.1: Representation of the CSV table

new one called *normal*, that aims to represent normality, so we have defined five different drift patterns which cover different behaviours corresponding to classes (hashtags). DOTS receives a document set for each class of tweets containing the same hashtag. A CSV table with different drift patterns was also defined, reproducing the artificial drifts, namely: *sudden*, *gradual*, *incremental*, *reoccurring* and *normal*. As an example, a sudden drift might be represented by tweets from a hashtag that in a given temporal moment starts to appear with a significant frequency. We have requested 6410 tweets, and considered the hashtags *#syrisa*, *#airasia*, *#android* and *#sex* to represent, respectively, sudden, gradual, incremental, reoccurring, and normal drift. We tried to use mutually exclusive concepts to avoid misleading the classifier, since two different tweets may represent the same concept. Table 6.1 represents the frequency of tweets in the 9 moments we have defined, and it is the representation of the CSV table required by DOTS.

Each tweet was represented by DOTS as a vector space model and preprocessing methods were applied, like stopword removal and stemming. We have exported, for our convenience, in SVMLight format, as we decided to use Support Vector Machines (SVM) as the learning model of this case study. Two different weighting schemes in document representation were tested, term-frequency and *tf-idf*. Table 6.2 presents the obtained results using both weighting schemes. In order to evaluate the possible outcomes of the classification, we used the van Rijsbergen F_β measure with $\beta = 1$, presented in Section 2.3.4. The results shed light on the use of term-frequency to classify Twitter messages in the presence of drift. However, the aim of this experiment was to demonstrate the use of the framework in a Twitter stream case study and further

analysis should be done to explain the obtained results. As document representation in Twitter streams is not in the scope of this thesis, we will use *tf-idf* in future experiments, considering the work by Salton & Buckley (1988), that states that term frequency alone cannot ensure acceptable retrieval performance and that a reasonable measure of term importance may be obtained by using the product of the term frequency and the inverse document frequency, the *tf-idf* measure.

Drift	Hashtag	F_1 using tf	F_1 using tf-idf
Sudden	#syrisa	79.37%	75.32%
Gradual	#airasia	57.88%	56.08%
Incremental	#isis	83.49%	81.75%
Reoccurring	#android	60.66%	59.49%
Normal	#sex	74.88%	74.56%

Table 6.2: Performance results obtained with Twitter stream

Equipped with this framework it becomes possible to propose and test different approaches, as presented in the following.

6.3 Dynamic Learning Models

For classifying dynamic time series like the Twitter stream, or other text-based social networks, we have to devise proper learning models. As an example, if we want to forecast which topics will become trends on Twitter, we have to carefully choose our guiding hypothesis for this setting.

We present three models to tackle learning in such dynamic environments: a time-window model, an ensemble model, and an incremental model. The time-window model is a batch learning model unable to retain all the previously seen examples. The incremental model learns batches of documents and has a memory mechanism that allows awareness of previously seen examples. Unlike the previous ones, the ensemble model has a modular structure which enables temporal adaptation to new incoming tweets on the basis of the data sampling real distribution over time. In a way, the built-in memory mechanism is inherited from the (recent) past. It is noteworthy to refer that the models we are proposing aim to be independent of the classifier used.

In the following sections we will detail the three proposed models and compare them in the Twitter classification problem.

6.3.1 Time-window Model

Algorithm 6.1 defines the basic steps of the time-window model. For each collection of documents $\mathcal{T}$ in a time-window t, $\mathcal{T}^t = \{x_1, \ldots, x_{|T^t|}\}$ with labels $\{y_1, \ldots, y_{|T^t|}\} \rightarrow \{-1, 1\}$, the dataset $\mathcal{D}^t$ is updated with the newly seen documents, $\mathcal{T}^t$. Previously seen documents are not stored in $\mathcal{D}^t$ and thus the $\mathcal{C}^t$ classifier is always trained with the examples of the most recent time-window.

Algorithm 6.1: Time-window Model

Input:
For each collection of documents $\mathcal{T}$ in a time-window t, $\mathcal{T}^t = \{x_1, \ldots, x_{|T^t|}\}$ with labels $\{y_1, \ldots, y_{|T^t|}\} \rightarrow \{-1, 1\} \quad t = 1, 2, \ldots$ T

1 **for** *t=1,2,... T* **do**
2 | $\mathcal{D}^t \leftarrow \mathcal{T}^t$
3 **end**
4 BaseClassifier $\mathcal{C}^t$: Learn ($\mathcal{D}^t$), obtain: h^t: $\mathcal{X} \rightarrow \mathcal{Y}$
5 Time-window Classifier $\mathcal{C}^t$: Classify ($\mathcal{T}^{t+1}$), using: h^t: $\mathcal{X} \rightarrow \mathcal{Y}$

The rationale of this model is to use the most recent information, based on the implicit idea that the most recent is also the most relevant. Another important insight about this model is that by discarding the information prior to the most recent batch it can be computationally lightweight.

6.3.2 Incremental Model

The incremental model, unlike the time-window model, uses all the previously seen examples, as can be illustrated in Algorithm 6.2, by updating the documents collection $\mathcal{D}^t$ in an incremental manner. One of the major advantages of this model is that it retains all the information gathered over time, so no relevant information is lost. On the other hand, one can argue that continuously increasing $\mathcal{D}^t$ would lead to storage problems, and to an unavoidable computational burden. A different strategy should be put forward to avoid the drawback of this ever-growing problem.

Algorithm 6.2: Incremental Model

Input:
For each collection of documents $\mathcal{T}$ in a time-window t, $\mathcal{T}^t = \{x_1, \ldots, x_{|T^t|}\}$ with labels $\{y_1, \ldots, y_{|T^t|}\} \rightarrow \{-1, 1\} \quad t = 1, 2, \ldots$

1 **for** *t=1,2,...* T **do**
2 $\quad \mathcal{D}^t \leftarrow \mathcal{D}^t \cup \mathcal{T}^t$
3 **end**
4 BaseClassifier $\mathcal{C}^t$: Learn ($\mathcal{D}^t$), obtain: h^t: $\mathcal{X} \rightarrow \mathcal{Y}$
5 Incremental Classifier $\mathcal{C}^t$: Classify ($\mathcal{T}^{t+1}$), using: h^t: $\mathcal{X} \rightarrow \mathcal{Y}$

6.3.3 Ensemble Model

The ensemble model, presented in Algorithm 6.3, proposes to store all the information gathered in a different way. For each collection of documents $\mathcal{T}$, that contain both positive and negative examples and occur in a time-window t, a classifier $\mathcal{C}^t$ is trained and stored. When a new collection of documents in the subsequent time-window occur, all the previously trained classifiers are loaded, and will classify the newly seen examples. The prediction function of the ensemble, composed by the set of classifiers already created, is a combined function of the outputs of all the considered classifiers. Several strategies can be used herein, for instance a majority voting strategy, where each model participates equally. If the sum of all votes is a null value, which means a tie, the classification of the most recent classifier is used to untie. The major advantage of this approach is that documents of the previously seen time-windows are not stored. The information is still retained as the classifiers trained with those examples contribute to the decision of the ensemble.

6.3.4 Dataset

In line with the use of DOTS presented in Section 6.2.2, we have requested Twitter messages so different drift patterns could be represented to better evaluate and validate the proposed models. It is important to note that since a Twitter labelled dataset is missing so far, the strategy to label the Twitter messages is to use the hashtags enclosed in the message as the message classification, as previously introduced in Section 5.5. Differently from what we have done in Section 6.2.2, where we have only used 5 different hashtags, we are now proposing 10 different hashtags. We have doubled the number

Algorithm 6.3: Ensemble Model

Input:
For each collection of documents $\mathcal{T}$ in a time-window t, $\mathcal{T}^t = \{x_1, \ldots, x_{|T^t|}\}$ with labels $\{y_1, \ldots, y_{|T^t|}\} \rightarrow \{-1, 1\} \quad t = 1, 2, \ldots$ T

for *t=1, 2, ... T* **do**
  $\mathcal{D}^t \leftarrow \mathcal{T}^t$
  BaseClassifier $\mathcal{C}^t$: Learn ($\mathcal{D}^t$), obtain: h^t: $\mathcal{X} \rightarrow \mathcal{Y}$
end
for *k=1, ... , t* **do**
  ModuleClassifier $\mathcal{C}^k$: Classify ($\mathcal{T}^{t+1}$), using: h^k: $\mathcal{X} \rightarrow \mathcal{Y}$
end
Ensemble $\mathcal{E}^t$: Classify ($\mathcal{T}^{t+1}$), using: $e^t = \begin{cases} \frac{\sum_t h^t(\mathcal{T}^{t+1})}{|\sum_t h^t(\mathcal{T}^{t+1})|} & \text{if } \sum_t h^t(\mathcal{T}^{t+1}) \neq 0 \\ h^t(\mathcal{T}^{t+1}) & \text{if } \sum_t h^t(\mathcal{T}^{t+1}) = 0 \end{cases}$

Drift	Hashtag
Sudden #1	#bradpitt
Sudden #2	#realmadrid
Gradual #1	#ryanair
Gradual #2	#literature
Incremental #1	#twitter
Incremental #2	#ferrari
Reoccurring	#syria
Regular #1	#jobs
Regular #2	#sex
Regular #3	#nowplaying

Table 6.3: Correspondence between type of drift and hashtag

Time-window	Sudden#1	Sudden#2	Gradual#1	Gradual#2	Incremental#1	Incremental#2	Reoccurring	Normal#1	Normal#2	Normal#3
1	0	0	0	60	0	60	0	20	50	20
2	0	0	10	60	0	60	0	20	50	50
3	0	0	10	0	0	60	0	20	50	20
4	0	0	0	0	0	60	0	20	50	50
5	0	50	0	50	0	60	0	20	50	20
6	0	50	20	50	4	56	50	20	50	50
7	0	50	20	0	8	52	50	20	50	20
8	0	50	0	0	12	48	50	20	50	50
9	0	50	0	40	16	44	0	20	50	20
10	0	50	30	40	20	40	0	20	50	50
11	0	0	30	0	24	36	0	20	50	20
12	0	0	0	0	28	32	0	20	50	50
13	0	0	0	30	32	28	0	20	50	20
14	0	0	40	30	36	24	50	20	50	50
15	50	0	40	0	40	20	50	20	50	20
16	50	0	0	0	44	16	50	20	50	50
17	50	0	0	20	48	12	0	20	50	20
18	0	0	50	20	52	8	0	20	50	50
19	0	0	50	0	56	4	0	20	50	20
20	0	0	0	0	60	0	0	20	50	50
21	0	0	0	10	60	0	0	20	50	20
22	0	0	60	10	60	0	50	20	50	50
23	0	0	60	0	60	0	50	20	50	20
24	0	0	0	0	60	0	50	20	50	50
Total:	**150**	**300**	**420**	**420**	**720**	**720**	**450**	**480**	**1200**	**840**

Table 6.4: Frequency of hashtags over time

of classes so we could represent some variability inside each drift pattern, like a soft sudden drift or a more deeper one, or an incremental drift with an ascending pattern and an incremental drift with a descending one. Table 6.3 shows the chosen hashtags and the corresponding drift and Table 6.4 shows the different types of drifts sorted by the artificially generated timestamps and their corresponding time-windows.

Time is represented as 24 continuous time-windows, in which the frequency of each hashtag is changed in order to represent the defined drifts. Each tweet is then timestamped so it can belong to one of the time-windows we have defined. Table 6.4 also represents the frequency we have defined for each hashtag over time.

Analysing Table 6.4, one can also notice that, as stated above, there are 2 instances of sudden, gradual, and incremental drifts, 1 instance of a reoccurring drift and 3 with normal feeds. For instance, in the reoccurring drift we introduce 50 tweets in time-windows 6, 7, and 8, and later in time-windows 14, 15, and 16. Normality is represented here to show tweets that occur in a continuous frequency, i.e., without drift.

The Twitter API (`https://dev.Twitter.com`) was used to request public tweets that contain the defined hashtags. The requests were done between the 19^{th} and the 31^{st} of October 2013. Tweets were only considered if the user language was defined as English. We have requested more than 10,000 tweets, even though some of them were discarded, like those tweets containing no message content besides the hashtag.

6.3.5 Experimental Results and Analysis

In this section we evaluate the performance obtained on the Twitter dataset using the three approaches described in previous sections. The evaluation of our approach was done using the support vector machines as the base classifier of each model. We have used the SVMLight[1] package with linear kernels and default parameters.

Table 6.5 summarizes the performance results obtained by classifying the dataset, considering the F_1 measure. Analysing the table we can observe that the use of the incremental approach outperforms the overall classification of the time-window model and the ensemble model, except in the *Sudden #1* drift. In this particular drift, the ensemble model outperforms the incremental model with an F_1 score of 58.42% against the F_1 score of 54.85%. Nevertheless, this is the only drift in which this occurs. The

[1] `http://svmlight.joachims.org/`

	Time-window	**Ensemble**	**Incremental**
Sudden #1	55.93%	58.42%	54.85%
Sudden #2	60.22%	80.12%	88.84%
Gradual #1	49.88%	40.45%	65.21%
Gradual #2	45.08%	74.53%	82.41%
Incremental #1	41.41%	30.69%	60.35%
Incremental #2	52.01%	61.72%	79.45%
Reoccurring	73.59%	82.92%	89.74%
Normal #1	55.78%	55.53%	84.44%
Normal #2	57.69%	88.05%	93.23%
Normal #3	23.71%	30.65%	65.35%
Average:	**51.53%**	**60.31%**	**76.39%**

Table 6.5: Comparative results considering F_1 measure

explanation might be related to being a fast occurring sudden drift, as it appears and disappears rapidly, differently from the *Sudden #2* that occurs during a longer period. The incremental model fails to identify small and fast occurring drifts. This is due to having a broader view of the whole time collection. In this particular case, a model with less memory can be appropriate as there is no gain in retaining the information about this drift.

When considering the average of the F_1 score, the time-window model scores 51.53%. The ensemble model with 60.31% outperforms the time-window model. Finally, the ensemble model is outperformed by the incremental model with an F_1 score of 76.39%. These results were expected as the incremental model decides according the knowledge of the whole collection, differently from the time-window model and even the ensemble model. One can argue that the ensemble model, by using the time-window models created in each time-window, could still have the information of the previously seen examples; however, as the outcome decision is combined, errors can also arise as all the models previously created contribute to the decision and can induce errors in the final decision.

It is also important to note that the ensemble model performs better than the time-window model in the majority of drifts, nevertheless, in the drift *Gradual #1* and in the drift *Incremental #1*, the ensemble scores 40.45% against 49.88% and 30.69% against 41.41%, respectively, which are significant results. These drifts have the particularity of being the only ones that increase their frequency over time, which seems to denote

that there is a relation between their nature and the performance obtained by the ensemble model. The explanation for this phenomenon is that in the first occurring time-windows, considering the timeline, the time-window models that are created to compose the ensemble tend to fail, as they have not seen enough positive examples. In the last time-windows they contribute equally to the output of the ensemble and influence in a negative way the classification provided by the ensemble. This does not occur with decreasing frequency drifts because when the models that have seen less positive examples start to participate in the ensemble decision, the examples they have to identify are in lesser number (as the frequency is decreasing) and thus the ensemble fails in a smaller proportion. This also seems to denote that the ensemble model tends to take more time to adapt to a changing environment.

Besides the mentioned drifts, in *Normal #1* the ensemble model is also outperformed by the time-window model, but in this case with less significant results, 55.53% against 55.78%. We believe that this is related to the tie-break mechanism, as the examples misclassified are just a few (when comparing with the time-window model) and are those in which there was a tie and the last model, that is called to untie, fails the decision. Finally, the results observed in the classification of *Normal #3* were unexpected, as normality should be easily identified. The results might be explained by the hashtag we chose to represent it, *#nowplaying*. This hashtag is commonly used to refer to the songs that users are playing in their computers or mobile devices, usually just posting the song name and the corresponding artist. Considering the spectrum of musics and artists, the diversity of those tweets, along with its short encoding, might compromise the performance of the classifier. One can also notice that *Normal #1*, with the hashtag *#jobs*, can suffer from the same characteristics, still, in this particular case, tweets have linking words like hiring, recruiting, or opportunity that might be informative of the tweet content.

6.4 Dataset for Drift in Twitter

One of the major difficulties in dynamic environments, and specially in the Twitter scenario, is the inability to find adequate benchmarks to test the effectiveness of learning models that aim to cope with different drift patterns. Even if we do not consider the special case of Twitter, we have identified two text-based datasets that represent drift:

the *Emailing List dataset* and the *Spam Filtering dataset*. They were both presented by Katakis *et al.* (2010). At a first glance, they both represent a text-based dataset with drift.

The Emailing List dataset simulates a stream of email messages from different topics, like science/medicine or science/space, that are sequentially labelled by a user as interesting, or junk, according to the user's personal interests. In email filtering, the idea is to take advantage of this user feedback in order to train a model that will automatically classify messages as interesting or junk. This dataset has been also referred in literature as the Usenet dataset, as the email messages were collected from usenet posts.

The Spam Filtering dataset contains messages from the Spam Assassin Collection[1]. The corpus comes in four parts: *spam*, *spam2*, *ham* (legitimate), and *easy ham*. The spam ratio of the Spam Assassin collection is approximately 20 %, and a timestamp was added to the dataset so time could be represented. Nevertheless, they both represent text-based datasets with drift but they do not represent the full spectrum of drift patterns. In the Emailing List dataset only sudden and reoccurring drifts are represented, while in Spam Filtering the only represented one is gradual drift.

So far, we have not identified any Twitter-based dataset that includes drifts representation, so we propose a new contribution: a dataset for drift in Twitter.

We have presented so far two preliminary datasets created using DOTS to evaluate and validate learning strategies in Twitter. We have used them in the previously mentioned studies, but we found out they could be improved, specially because some of the used hashtags, like #nowplaying, could have intrinsic characteristics that could induce misclassification, as suggested in the previous section. We have also extended not only the number of time moments, but as well the number of tweets, as we intended to study the impact of time, so longer drifts would need to be represented. For future analysis, and considering the above mentioned lack of a social network dataset with drift, we then propose a dataset for learning in the Twitter environment, considering the existence of drift.

The dataset we have defined was carried out by defining 10 different hashtags that would represent our drifts, based on the assumption that they would denote mutually exclusive concepts, like *#realmadrid* and *#android*. As already mentioned, the use

[1] http://spamassassin.apache.org/

of mutually exclusive concepts is mandatory to avoid misleading a classifier, as two different tweets could represent the same concept, and that way introducing a new variable to our scenario that could mislead the possible obtained results. In order to achieve a considerable amount of tweets, and consequent diversity, we have chosen trending hashtags like *#syrisa* and *#airasia*.

The Twitter API[1] was then used to request public tweets that contain the defined hashtags. The requests have been taken care of between 28^{th} of December 2014 and 21^{st} of January 2015 and tweets were only considered if the user language was defined as English. We have requested more than 75,000 tweets concerning the given hashtags, even though some of them were discarded, like for instance those tweets containing no message content besides the hashtag. The hashtag was then removed from the message content in order to be exclusively used as the document label. The tweets matching these presumptions were considered labelled and suitable for classification purposes, and were used in their appearing order in the public feed.

Table 6.6 shows the chosen hashtags and the corresponding drift they represent. This correspondence was done arbitrarily and does not correspond to any possible occurrence in the real Twitter scenario since, as stated above, no information is known about the occurrence of drifts in Twitter.

We have simulated the different types of drift by artificially defining timestamps to the previously gathered tweets. Time is represented as 100 continuous time-windows, in which the frequency of each hashtag is altered in order to represent the defined drifts. Each tweet is then timestamped so it can belong to one of the time-windows we have defined. For instance, Sudden #1 is represented by the appearance of 500 tweets with the hashtag *#syrisa* in each time-window from 25 to 32, and in any of the other time-windows this hashtag appears. Differently from *Sudden #1*, *Sudden #2* is represented with only 200 tweets with the hashtag *#airasia* in each time-window from 14 to 31; we tried to simulate a more soft occurring drift, but with a more long-standing appearance. By making both concepts disappear, in time-windows 32 and 31, respectively, we also intended to simulate the opposite way of the Žliobaitė (2010) proposed sudden drift. Due to space constraints it is unbearable to present a table with the frequency of each hashtag in each time-window, but it is important to state that *Incremental #2* and

[1] https://dev.Twitter.com/

Gradual #2 are represented by the same number of tweets in an equal number of time-windows, but in reverse order than represented in *Incremental #1* and *Gradual #1* and *Normal #1*, *Normal #2* and *Normal #3* differ in the number of tweets that appear in a constant way in all the time-windows. Our final dataset contains 34,240 tweets and will be used in the contributions we will detail further.

Drift	Hashtag
Sudden #1	#syrisa
Sudden #2	#airasia
Gradual #1	#isis
Gradual #2	#bieber
Incremental #1	#android
Incremental #2	#ferrari
Reoccurring	#realmadrid
Normal #1	#jobs
Normal #2	#sex
Normal #3	#nfl

Table 6.6: Mapping between type of drift and hashtag

Considering that in most problems it is infeasible to use all the data produced, it is essential to understand how informative past events can be to current learning models. In the next section we will detail the experiments done to understand the impact of longstanding messages in Twitter, the case study we have chosen in this thesis to represent dynamic environments.

6.5 Time in Dynamic Environments

Time plays an important role by easily out-dating information, being crucial to understand how informative can past events be to current learning models and for how long it is relevant to store previously seen information, to avoid the computational burden associated with the amount of data produced.

In order to understand the importance of past examples in the classification process, we present a batch learning model that retains previously seen examples during a defined period. By retaining examples during different periods we aim to evaluate for how long it is relevant to keep information according to the different types of drift, and thus best tailoring the memory mechanism needed for classification purposes.

6.5.1 Batch Learning Model

Algorithm 6.4 defines the basic steps of our batch learning model. For each collection of documents $\mathcal{T}$ in a time-window t, $\mathcal{T}^t = \{x_1, \ldots, x_{|T^t|}\}$ with labels $\{y_1, \ldots, y_{|T^t|}\} \rightarrow \{-1, 1\}$, and considering the training window size j, the dataset $\mathcal{D}^t$ is updated incrementally until the training window size is complete. By updating the documents collection $\mathcal{D}^t$ based on a training window we retain the information during a defined amount of time, discarding the examples that occur before that moment.

Algorithm 6.4: Batch Learning Model

Input:
For each collection of documents $\mathcal{T}$ in a time-window t, $\mathcal{T}^t = \{x_1, \ldots, x_{|T^t|}\}$ with labels $\{y_1, \ldots, y_{|T^t|}\} \rightarrow \{-1, 1\} \quad t = 1, 2, \ldots$

Training window size j

```
for i=1,2,... j do
    D^t ← D^t ∪ T^{t-i}, if T^{t-i} exists, otherwise D^t ← T^t
    i++
end
Classifier C^t : Learn (D^t), obtain: h^t: X → Y
Classifier C^t : Classify (T^{t+1}), using: h^t: X → Y
```

For this experiment we have also used the Twitter case study and the dataset presented in Section 6.4, as Twitter is a particular case of a time series in dynamic environments. The rationale of this learning strategy is to divide the classification task in multiple batch models, each one trained to classify a time-window. When a new collection of documents in the subsequent time-window occur, we will create a new batch learning model as proposed above to classify the newly seen examples. As in previous experiments, support vector machines were also used as the base classifier of the batch learning model.

6.5.2 Experimental Results and Analysis

In this section we evaluate the performance yielded on the Twitter dataset using the batch learning model described in the previous section. Table 6.7 summarises the

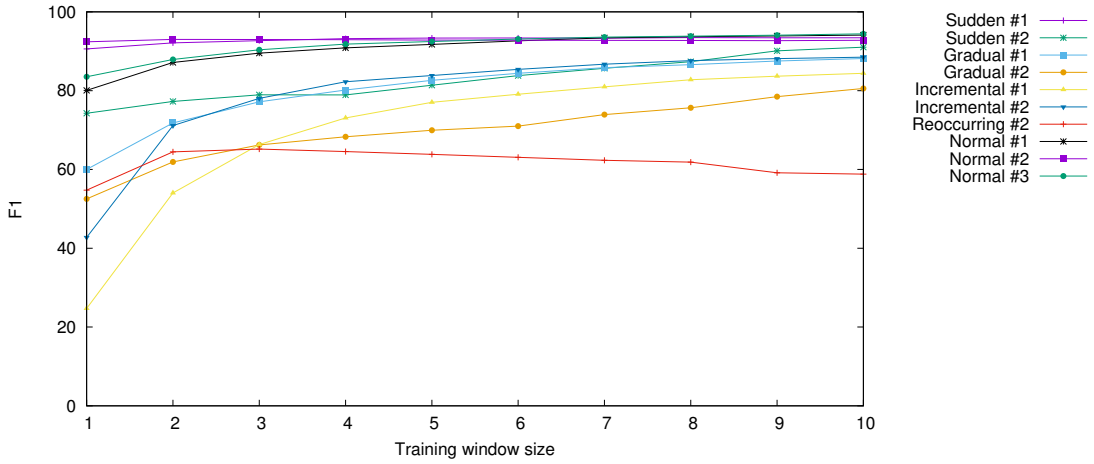

Figure 6.4: Micro-averaged F_1 obtained by training window

performance results obtained by classifying the dataset, considering the micro-averaged F_1 measure.

Analysing the table we can observe that in global terms, and considering the average of the micro-averaged F_1, the increase of the training window size improves the classification performance. This is normal and expected, as the learning models are trained with more examples and this leads to a better performance. Nevertheless, and considering that it is unreasonable to store all the examples for training purposes, it is important to determine the best relation between performance and the computational burden associated with storing and processing the training examples.

As depicted in Figure 6.4, the increase in the average of micro-averaged F_1 seems to slow down above a training window size of 4, which means that above that value the cost benefit relation is less substantial. It is also important to note the greater performance increase from training window size of 1 to training window size of 2. This happens because using two training windows instead of one implies the training window size was doubled, which is a major improvement. From then on the proportion is less substantial, for instance from training window 2 to training window 3 there is only a 50% increase, while from training window size of 3 to training window size of 4 we have an increase of 33%, and so forth. One can argue that this is highly dependent on the computational complexity of processing one time-window and no values are shown in this study about this complexity, but we have used similar sized time-windows,

Drift	Training window size 1	2	3	4	5	6	7	8	9	10
Sudden #1	92.40%	93.01%	92.99%	92.92%	92.83%	92.80%	92.75%	92.73%	92.69%	92.78%
Sudden #2	90.60%	92.12%	92.70%	93.19%	93.34%	93.36%	93.33%	93.46%	93.43%	93.41%
Gradual #1	52.53%	61.91%	66.22%	68.29%	69.95%	71.00%	73.91%	75.64%	78.49%	80.56%
Gradual #2	74.27%	77.26%	78.94%	78.93%	81.39%	83.82%	85.67%	87.35%	90.10%	91.04%
Incremental #1	83.53%	87.90%	90.37%	91.82%	92.46%	93.11%	93.58%	93.83%	94.08%	94.41%
Incremental #2	60.01%	71.79%	77.17%	80.15%	82.62%	84.41%	85.85%	86.60%	87.52%	88.15%
Reoccurring	54.74%	64.47%	65.17%	64.53%	63.83%	63.08%	62.33%	61.85%	59.14%	58.81%
Normal #1	24.72%	54.03%	66.32%	73.07%	77.03%	79.08%	80.97%	82.78%	83.68%	84.40%
Normal #2	80.07%	87.15%	89.53%	90.87%	91.74%	92.64%	93.40%	93.68%	93.89%	94.11%
Normal #3	42.75%	71.14%	78.03%	82.25%	83.85%	85.40%	86.72%	87.61%	88.12%	88.50%
Average of micro-averaged F_1	71.85%	80.01%	83.16%	84.91%	86.09%	87.04%	87.92%	88.53%	89.10%	89.54%

Table 6.7: Comparative results considering micro-averaged F_1

		Training window size				
Time-window	Sudden #1	1	2	3	4	5
20	0	0	0	0	0	0
21	0	0	0	0	0	0
22	0	0	0	0	0	0
23	0	0	0	0	0	0
24	0	0	0	0	0	0
25	500	0	0	0	0	0
26	500	500	500	500	500	500
27	500	500	1000	1000	1000	1000
28	500	500	1000	1500	1500	1500
29	500	500	1000	1500	2000	2000
30	500	500	1000	1500	2000	2500
31	500	500	1000	1500	2000	2500
32	500	500	1000	1500	2000	2500
33	0	500	1000	1500	2000	2500
34	0	0	500	1000	1500	2000
35	0	0	0	500	1000	1500
36	0	0	0	0	500	1000
37	0	0	0	0	0	500
38	0	0	0	0	0	0
39	0	0	0	0	0	0
40	0	0	0	0	0	0

Table 6.8: Effect of increasing the training window size in the training set

which means that in proportional terms one can define the time-window based on the computational complexity that it can handle.

Although the overall performance seems to improve with the increase of the training window size, there are particular cases in which the performance decreases. There are losses that are so small, and do not define a pattern, that one can consider as insignificant, e.g., in the drift *Gradual #1* from training window size of 3 to training window size of 4, respectively 78.94% to 78.93%. But there are also performance losses that, small or not, seem to define a pattern that might be related with the nature of the drift pattern that is represented, like *Sudden #1*, *Sudden #2* and *Reoccurring*.

The performance in the identification of tweets from *Sudden#1* increases from training window size of 1 to training window size of 2. As explained above, this is related to the doubled size training window, but then on, even in small amounts, the perfor-

mance starts to decrease. As mentioned in the Section 6.4, drift *Sudden #1* occurs only in time-windows from number 25 to 32, with a constant amount of tweets per time-window, 500 in each time-window. Table 6.8 represents the time-windows from 20 to 40, i.e., the frequency of *Sudden #1* in those time-windows, along with the number of positive examples in the training set that is used to train the model that classifies the examples of that time-window considering different training window sizes. As an example, to classify the examples in time-window 27, with training window size of 2, the classifier receives as training samples all the examples from time-windows 25 and 26, 1000 positive examples, which means that it sees more positive ones than it would if trained with only examples from time-window 26 (500 examples, if the training window size is 1). However, when we increase the training window size to 5, the classifier in the training phase sees the same 1000 positive examples, despite the inclusion of examples from time-windows 22, 23, 24, 25, and 26, as in time-windows 22, 23, and 24 there are only negative ones, as the drift only starts to happen in the time-window 25. This explains why in sudden drifts the increase of the training window size might lead to a decrease in the performance, because past events hardly contribute with positive examples, as the drift appeared in a sudden way in a specific temporal moment. Another important insight is the effect that occurs when the drift deceases. *Sudden #1* occurs until time-window 32, but when we enlarge the training window size we maintain positive examples for a longer period, as can be seen in Table 6.8. As an example, if the training window size is 2, positive examples cease to be used in time-window 33, but if we increase the training window size to 5 they only cease in time-window 37.

However, considering *Sudden #2*, one must take a note on the results obtained in the classification performance, because whilst it is a drift with the same nature, one should expect the same pattern in the classification performance, but it did not occur, specially until training window size of 7. The major difference from drift *Sudden #1* to *Sudden #2* is that *Sudden #1* is more abrupt than *Sudden #2*. As referred in Section 6.4, the number of examples that appear in each time-window where *Sudden #2* is represented is less than a half that *Sudden #1*, from 500 to 200 tweets, but it also happens that *Sudden #2* is much longer in time than *Sudden #1*. Being a much longer drift past events contribute differently to the performance of the classifier, because in *Sudden #2* past windows might more easily contribute with more positive examples than in *Sudden #1*. This attests the difficulty of the problem, as it is not only the

nature of the drift pattern, but also its particular characteristics that might affect the performance of the classification.

There is also a decreased performance pattern in the drift *Reoccurring*. Firstly, it is importance to explain the characteristics of *Reoccurring* in order to understand the obtained results. *Reoccurring* occurs in 5 consequent time-windows, for instance from time-window 12 to 16, or from time-window 28 to 32, and then disappears during 11 time-windows. As a consequence, from time-window 17 to 27 (both inclusive) there are no positive examples from this class. The same occurs from time-window 33 to 43, but in 44 until 48 the drift pattern in again represented with positive examples. The performance decrease above training window size of 3 might be explained as *Reoccurring* can be seen as having the same characteristics of the sudden drifts, specially because it always disappears for 11 time-windows. As we do not use in our experiments training window sizes bigger than 10, we do not reach the point at which we provide the classifier with positive examples from the previous burst in which the drift occurs, and thus increasing the training window size, and not reaching that moment, could always lead to the same result that happens with the sudden drift.

The results revealed the usefulness of our strategy, specially because it is easy to identify a major slowdown in the increase of performance from training window size of 4 to the subsequent training window sizes. More precisely, we have identified that there is a major improve from training window size of 1 to training window size of 2. Even though in average enlarging the training window size is echoed in an increase in the classification performance, the cost benefit decreases from then on, and specially above training window size of 4.

It is also important to highlight the effect of enlarging the training window size in the classification performance considering drifts with the same nature, specially in drifts like sudden drift. In these cases, a different strategy must be put forward, as enlarging the training window size will not always lead to a performance increase, and can rather have a counter-productive effect, depending on the abruptness and the longstanding of the represented drift pattern.

6.6 Conclusion

In this chapter we have focused on learning strategies in dynamic environments. We presented and detailed the Drift Oriented Tool System (DOTS), a framework that allows for the definition and generation of text-based datasets for dynamic environments. We then proposed three learning models, namely the time-window model, the incremental model and the ensemble model, and then we presented a dataset to be used to test and evaluate learning strategies in the social networks' environment. Finally, we have studied the effect on longstanding messages in dynamic environments, proposing a batch learning model to tackle the infeasibility of storing all the previously acquired examples in problems like social network data streams.

Regarding the presented results, we have shown that memory, or the ability to keep the information already gathered, is important in the adaptability to drift in the learning process, as the incremental model tends to perform better than the other two models. We have also recognized training windows of size 4 as presenting an equilibrium between the computational burden of storing and processing huge amounts of data, and the usefulness of storing those examples. Nevertheless, as storing can be a constraint in the Twitter data stream, it is important to identify if there are examples better than others, and what is an outdated example, which means to understand which examples are considered most significant, the ones with interest to keep, and those which might not be relevant to identify future drifts of the same nature. Those could be discarded to reduce the computational effort.

In the next chapter we will introduce the Drift Adaptive Retain Knowledge (DARK) framework, which uses an ensemble of Support Vector Machines (SVM) with dynamic weighting schemes and variable training window sizes for model adaptation in incremental learning, and therefore aims at effectively learning in dynamic environments in text classification scenarios.

Chapter 7

Adaptive Learning in Dynamic Environments

In this chapter we focus on adaptive learning strategies to learn in dynamic environments with the use of ensembles. We start by presenting the Drift Adaptive Retain Knowledge (DARK) framework to effectively learn in dynamic environments in text classification scenarios. DARK uses an ensemble of classifiers with dynamic weighting schemes and variable training window sizes for model adaptation in incremental learning. We present two experiments. In the first experiment, we propose different strategies, using performance metrics, to combine models in an ensemble, whose aim is to learn in dynamic environments and, in the second experiment, we evaluate the relevance of specific examples in regard to others in the same environments. Finally, a comparative study of DARK with state-of-the-art solutions, namely the Learn++.NSE (NSE stands for Non Stationary Environments) algorithm is presented.

7.1 Introduction

Nowadays, most learning problems demand adaptive solutions, which can cope with new circumstances as they emerge. Paradigmatic to this setting are social networks' scenarios, as in our presented case study: Twitter, where new information appears all the time. Various efforts have been pursued in machine learning settings to learn in such environments, specially because of their non-trivial nature, since changes occur between the distribution data used to define the model and the current environment.

The learning task has specific training needs, because unlike other commonly used approaches, not all instances contribute equally to the final concept (Tsymbal *et al.*, 2008). Effective learning demands a learning algorithm with the ability to detect context changes without being explicitly informed about them, to quickly recover from the context change and to adjust its hypothesis to the new context. It should also make use of previous experienced situations when old contexts and corresponding concepts reappear (Widmer & Kubat, 1996).

In Elwell & Polikar (2011) the algorithm Learn++.NSE is proposed as an algorithm to deal with drift. It learns from consecutive batches of data without making any assumptions on the nature of drift. It learns from environments that experience constant or variable frequency drift, addition or deletion of concept classes, as well as reoccurring drift. We will discuss this algorithm further in chapter.

To deal with scenarios of imbalanced data, the authors in Ditzler & Polikar (2013) introduce the Learn++.NIE (Nonstationary and Imbalanced Environments) and the Learn++.CDS (Concept Drift and SMOTE) as two new members of the Learn++ family of incremental learning algorithms that explicitly and simultaneously address the aforementioned phenomena. Learn++.CDS is a combination of the Learn++.NSE algorithm with the SMOTE (Synthetic Minority Over-sampling TechniquE) algorithm proposed by Chawla *et al.* (2002). A different ensemble method called DWM-WIN (Dynamic weighted majority-Winnow) was recently proposed in Mejri *et al.* (2013), to overcome the known limits of Kolter & Maloof (2003), like not considering neither when the classifiers were defined nor the past correct classifications.

In Alippi *et al.* (2013) the authors present an adaptive classifier that exploits both supervised and unsupervised data to monitor the process' stationariness. The classifier follows the just-in-time approach and relies on two different change-detection tests to reveal changes in the environment and reconfigure the classifier accordingly. In Žliobaitė *et al.* (2014) a theoretically supported framework is presented for active learning from drifting data streams. In this case three active learning strategies are developed for streaming data that explicitly handle concept drift. They are based on uncertainty, dynamic allocation of labelling efforts over time, and randomization of the search space. Finally, in Dyer *et al.* (2014) the authors introduce COMPOSE (COMPacted Object Sample Extraction), a computational geometry-based framework to learn from non-

stationary streaming data, where labels are unavailable (or presented very sporadically) after initialization.

Ensembles of classifiers are a distinct approach to pursue the same goal. They integrate multiple classifiers to classify each example with the aim of improving classification performance, as described in Section 2.4.2. There are many approaches for ensemble of classifiers, such as *boosting* (Freund & Schapire, 1997), *bagging* (Breiman, 1996), or *random forests* (Breiman, 2001), but their original form is usually applied in static environments. However, ensembles are specially adequate to tackle dynamic evolving settings, given their modular nature, and different studies and approaches have been pursued (Ditzler & Polikar, 2013; Bagul & Phulpagar, 2016).

7.2 DARK Framework

The goal of the Drift Adaptive Retain Knowledge (DARK) framework is to build ensembles of classifiers with dynamic weighting schemes and variable training window sizes for model adaptation in incremental learning, and thus effectively learn in dynamic environments in text classification scenarios.

As already stated, one of the major challenges regarding dynamic environments with a substantial amount of data, like text data streams, is the infeasibility to store all previously seen data, even though it may carry substantial information for future use. The rationale of DARK is to use ensembles of classifiers to integrate multiple experts with different characteristics and thus benefit from their multitude, specially as they are created in different moments. DARK has a modular structure, which enables temporal adaptation to new incoming examples on the basis of the data sampling real distribution over time. There is a built-in memory mechanism that is inherited from the (recent) past, and thus an improvement in classification performance is achieved that would otherwise be dependent on more examples and computational burden.

Figure 7.1 depicts the DARK framework. It is divided in three parts, from top to bottom: (i) models' construction; (ii) learning process and (iii) models' combination.

The construction of the models (i) is carried out by defining time-windows and learning models for each time-window. In order to perceive the importance of past examples in the classification process we use the batch learning strategy presented in Section 6.5.1, that retains previously seen examples during a prefixed period. So, we

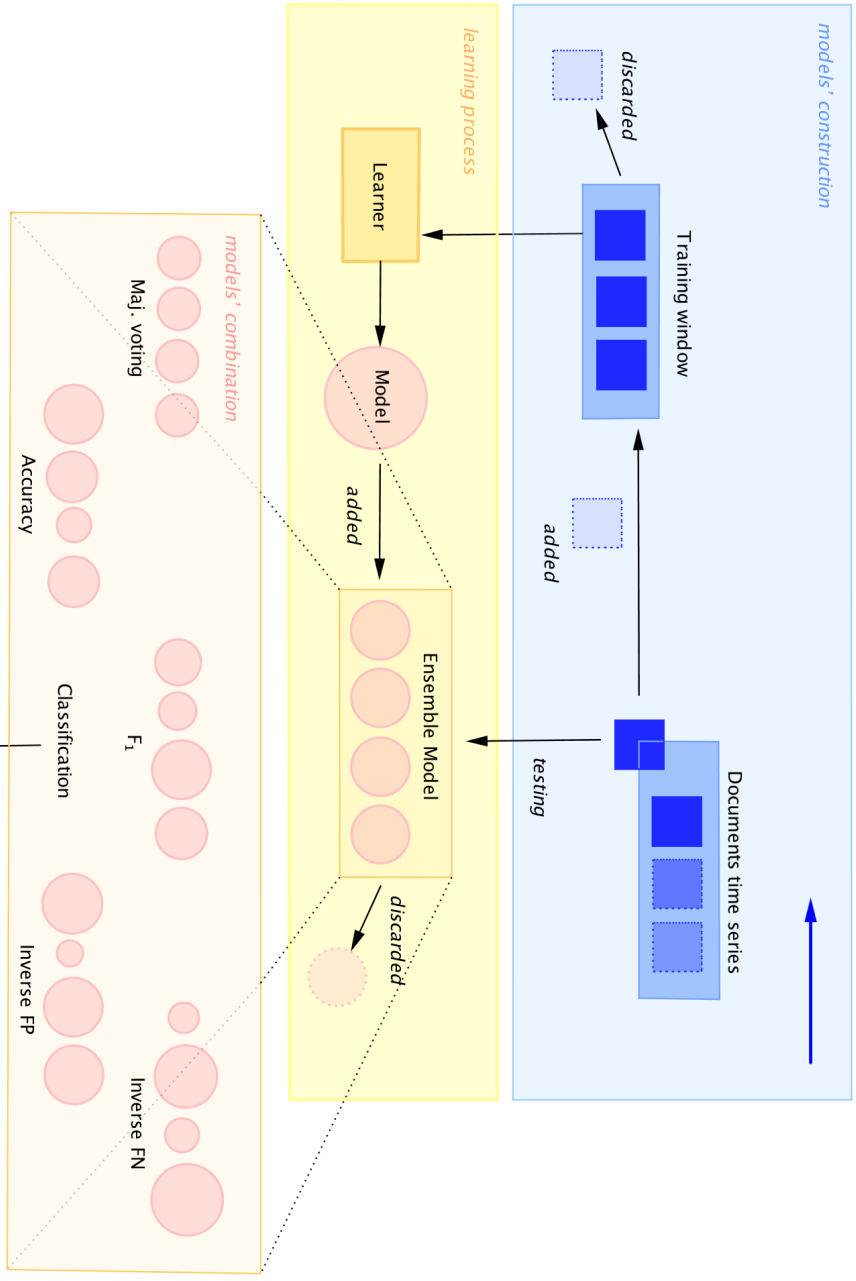

Figure 7.1: DARK framework (*training window size*=3 and *size of ensemble*=4)

aim to evaluate for how long it is relevant to keep information according to the different types of drift. This is the first memory mechanism that is present in our framework.

Different settings can also be constructed, i.e., the examples that are considered in each time-window may depend on the specific approach, like the one previously used, the timestamp of the example, or others such as the relevance, or the informativeness, among others possibilities.

The learning process (ii) focuses on the definition of the k baseline classifiers according to this algorithm. As base classifier we intend to represent a generic classifier, that might be chosen considering the classification problem we are trying to tackle, like a Naïve Bayes classifier, or a Support Vector Machine (SVM). Notice that, in dynamic environments, the ensemble must adapt to deal with changes usually dependent on hidden contexts, and it is usually infeasible to store all previously seen data, although it may carry substantial information for future use. Hence, not all previously constructed models are kept in the ensemble. Furthermore, the learning process determines which should be kept (or added) and which should be discarded Costa *et al.* (2014, 2015b).

Algorithm 7.1 presents the DARK framework learning process. Besides the creation of the base classifier model described in Algorithm 6.4, we also handle the combination of models in the ensemble. Firstly, for each time-window t we add the base classifier $\mathcal{C}^t$ to the ensemble $\mathcal{E}^t$. Secondly, we also consider that models can be outdated (in the same way we do with examples), so the ensemble $\mathcal{E}^t$ is pruned so that all the classifiers $\mathcal{C}^{t-k}$ that match the condition $t - k > 0$ are discarded and thus not included in the ensemble's final decision. Finally, we use the ensemble $\mathcal{E}^t$ to classify the document collection $\mathcal{T}^{t+1}$. The purpose of the classification is to define the unknown mapping function $e^t : \mathcal{X} \rightarrow \mathcal{Y}$, that predicts the class label y_i, according to x_i, the document message. In a timeline perspective, e^t uses the historical data $\{x_1, \ldots, x_t\}$ to predict x_{t+1} by combining the unknown prediction function $h^t : \mathcal{X} \rightarrow \mathcal{Y}$ provided by each base classifier model that composes the ensemble. The prediction function e^t can use different combining strategies for the output h^t of each classifier in the model's combining phase (iii), e.g. a majority voting strategy, where $e^t = \frac{\sum_t h^t(\mathcal{T}^{t+1})}{|\sum_t h^t(\mathcal{T}^{t+1})|}$. Other combining schemes might use performance metrics as we will mention further, which account for each individual contribution of the classifier in previous time-windows. In the next section we will study the impact of using different combining schemes of the ensemble on its classification performance.

Algorithm 7.1: DARK Framework learning process

Input:
For each collection of documents $\mathcal{T}$ in a time-window t, $\mathcal{T}^t = \{x_1, \ldots, x_{|T^t|}\}$ with labels $\{y_1, \ldots, y_{|T^t|}\} \rightarrow \{-1, 1\} \quad t = 1, 2, \ldots$

Training window size j

Ensemble size k

for *t=1,2,...* T **do**
  for *i=1,2,...* j **do**
    $\mathcal{D}^t \leftarrow \mathcal{D}^t \cup \mathcal{T}^{t-i}$, if $\mathcal{T}^{t-i}$ exists, otherwise $\mathcal{D}^t \leftarrow \mathcal{T}^t$
    $i++$
  end
  Classifier $\mathcal{C}^t$: Learn $(\mathcal{D}^t)$, obtain: h^t: $\mathcal{X} \rightarrow \mathcal{Y}$
  Ensemble $\mathcal{E}^t \leftarrow \mathcal{C}^t$
  if $t - k > 0$ **then**
    $\mathcal{E}^t \leftarrow \mathcal{E}^t \setminus \mathcal{C}^{t-k}$
  end
  Ensemble $\mathcal{E}^t$: Classify $(\mathcal{T}^{t+1})$ using: e^t: $\mathcal{X} \rightarrow \mathcal{Y}$
end

7.3 Combining Ensembles' Models

In this section we evaluate the effect of using different combining schemes in the performance obtained by the Drift Adaptive Retain Knowledge (DARK) framework in our case study, the Twitter classification problem. Besides a majority voting strategy, four performance metrics set forth in Section 2.3.4 were used to combine the ensemble model: accuracy, F_1 measure, the inverse of false positives ($\frac{1}{FP}$), and the inverse of false negatives ($\frac{1}{FN}$). The majority voting strategy is used as baseline, as all models contribute equally to the final decision of the ensemble, despite their previous performance.

Table 7.1 summarises the performance results obtained by classifying the dataset presented in 6.4, considering the micro-averaged F_1 measure.

Analysing the table we can observe that using different metrics to combine the ensemble can lead to different performance results, considering the Twitter classification problem. Globally we can achieve a 6% increase in the F_1 measure, when comparing the use of a majority voting strategy with micro-averaged F_1 score of 78.27% versus a performance-based strategy like $\frac{1}{FN}$ with an F_1 score of 84.39%.

It is particularly important to note that this performance increase is observed in all different types of drifts, despite their nature. We have also observed that the obtained results were achieved because the number of false negatives was reduced when using metrics based on the performance. Though this might be problem dependent, it is also relevant to pinpoint.

Table 7.2 summarises the performance results obtained by classifying the dataset, considering the micro-averaged Recall measure. Recall is highly dependent on the true positives, and the increase show us that using different metrics can reduce false negatives and consequently increase the true positives. The reduction of false negatives is, consequently, responsible for the increase of the F_1 results presented in Table 7.2, as precision is not significantly affected when using different combining strategies.

The results revealed the usefulness of using different combining strategies, as using different performance-based metrics led to improvement of the performance. This result was obtained with the micro-averaged F_1 when comparing to the baseline approach, a majority voting strategy, where all models contribute equally to the final decision of the ensemble, despite their previous performance.

	Learn++.NSE.SVM			**Learn++.NSE.CART**			**DARK.4.InverseFN**			**DARK.All.InverseFN**		
Drift	**Precision**	**Recall**	F_1	**Precision**	**Recall**	F_1	**Precision**	**Recall**	F_1	**Precision**	**Recall**	F_1
Sudden #1	70.20%	87.50%	77.90%	90.57%	80.65%	85.32%	99.78%	78.80%	88.06%	99.78%	78.80%	88.06%
Sudden #2	65.34%	87.86%	74.94%	92.79%	86.81%	89.70%	99.83%	83.75%	91.09%	99.83%	83.75%	91.09%
Gradual #1	84.91%	63.75%	72.82%	72.64%	61.63%	66.68%	99.90%	40.08%	57.21%	99.90%	40.08%	57.21%
Gradual #2	86.36%	74.92%	80.23%	92.99%	72.46%	81.45%	99.87%	62.88%	77.17%	99.87%	63.17%	77.39%
Incremental #1	83.44%	84.71%	84.07%	94.12%	87.66%	90.77%	99.29%	76.88%	86.66%	99.14%	79.03%	87.95%
Incremental #2	67.69%	80.29%	73.46%	60.57%	84.92%	70.70%	97.03%	49.09%	65.20%	97.27%	54.30%	69.69%
Reoccurring	75.02%	59.87%	66.59%	73.50%	52.33%	61.14%	99.29%	46.73%	63.55%	99.29%	46.73%	63.55%
Normal #1	97.93%	57.25%	72.26%	95.52%	68.08%	79.50%	100.00%	16.69%	28.60%	100.00%	16.69%	28.60%
Normal #2	89.84%	88.75%	89.29%	95.88%	87.45%	91.47%	98.50%	71.66%	82.96%	98.50%	71.72%	83.00%
Normal #3	83.04%	64.81%	72.80%	51.37%	74.31%	60.75%	99.20%	40.11%	57.12%	99.20%	40.11%	57.12%
Micro-averaged F_1	**77.94%**	**77.94%**	**77.94%**	**79.01%**	**79.01%**	**79.01%**	**99.14%**	**60.13%**	**74.86%**	**99.12%**	**61.13%**	**75.62%**

Table 7.1: Micro-averaged F_1 for different combining metrics

	Performance metrics for model fusion				
Drift	Maj. Voting	Accuracy	F_1	$\frac{1}{FP}$	$\frac{1}{FN}$
Sudden #1	59.78%	66.25%	78.53%	80.48%	80.52%
Sudden #2	78.36%	80.50%	86.61%	87.42%	87.39%
Gradual #1	35.67%	37.79%	49.00%	51.92%	51.92%
Gradual #2	45.54%	48.79%	62.67%	65.54%	65.54%
Incremental #1	79.87%	80.40%	81.40%	85.01%	85.24%
Incremental #2	63.67%	63.79%	64.83%	67.92%	67.68%
Reoccurring	21.47%	22.53%	42.67%	47.13%	47.73%
Normal #1	54.95%	55.03%	55.39%	57.53%	57.54%
Normal #2	82.93%	82.97%	83.03%	83.56%	83.68%
Normal #3	68.91%	69.07%	69.56%	69.80%	69.77%
Micro-averaged Recall	**64.55%**	66.08%	71.18%	73.25%	**73.29%**

Table 7.2: Micro-averaged Recall for different combining metrics

We may also conclude that the results obtained and the improvement observed are independent from the drift pattern the class represents, and thus can be applied in different dynamic scenarios. A more suited metric can better weight the models, as less performer models can cease their contribution.

So far we have only considered that all examples are equally important when they belong to the same time-window. Moreover, as the relevance of an example is based on the time-window, this means that time is the only factor of relevance. However, as we already enlightened in Section 7.2 it might be relevant to consider in each time-window examples depending on specific approaches, like the informativeness, among other possibilities.

In the next section we will try to perceive if there are examples more relevant for classification purposes than others, and thus choose them instead of others to improve the classification performance.

7.4 Ensemble Model Enrichment

We have already studied the impact of longstanding examples in future classification time-windows. The rationale was to store previously seen examples for a period of time regardless of the effect they might have as a solo example. The most relevant action

was to keep examples for a period of time instead of discarding them for future use. In that experiment, presented in Section 6.5, we were also not dealing with ensembles but single classifiers. Differently from that approach, we are now proposing to choose examples based on the effect they might have individually.

A baseline model is created for comparison purposes using the DARK framework. Considering Algorithm 7.1, that details the DARK framework learning process, the baseline model is characterized by having training window size $j = 4$ and ensemble size $k = 4$. The rationale of these values is based on the experiments presented in Section 6.5, where training window sizes greater than 4 present a decrease in the cost benefit relation. The prediction function of the ensemble in the baseline model is a combined function of all the outputs provided by the classifiers. A voting strategy is then put forward and thus each model participates equally in the final decision of the ensemble.

We then propose an ensemble learning model with an enrichment strategy, which we will designate as enriched model. The main difference is that we define a collection of documents $\mathcal{R}$ containing all the classification errors that occur in the time-windows prior to t. The classification errors are considered based on the ensemble classification and not in each model classification output. For each time-window t, a classifier $\mathcal{C}^t$ is trained with the collection of documents $\mathcal{T}$ plus the collection of documents $\mathcal{R}$, and stored. When a new collection of documents in the subsequent time-window occur, all the previously trained classifiers are loaded, and will classify the newly seen examples participating equally in the final decision of the ensemble.

The collection of documents $\mathcal{R}$ retains the misclassified examples indefinitely or will be pruned in a time-based approach. If pruned, the lifetime of an example in $\mathcal{R}$ is dependent on a pre-defined error window size g, which means that, in time-window t, $\mathcal{R}$ contains the misclassified examples that occur in all time-windows that satisfy the condition $t\text{-}g > 0$.

Figure 7.2 depicts the proposed models. It is important to understand that time is represented in two different directions because we are working with a time series and the last seen scenario is the input for the new one. The major difference between both models lies in that the enriched model uses the outcome of the classification as a new input in the subsequent training phase; the baseline model does not.

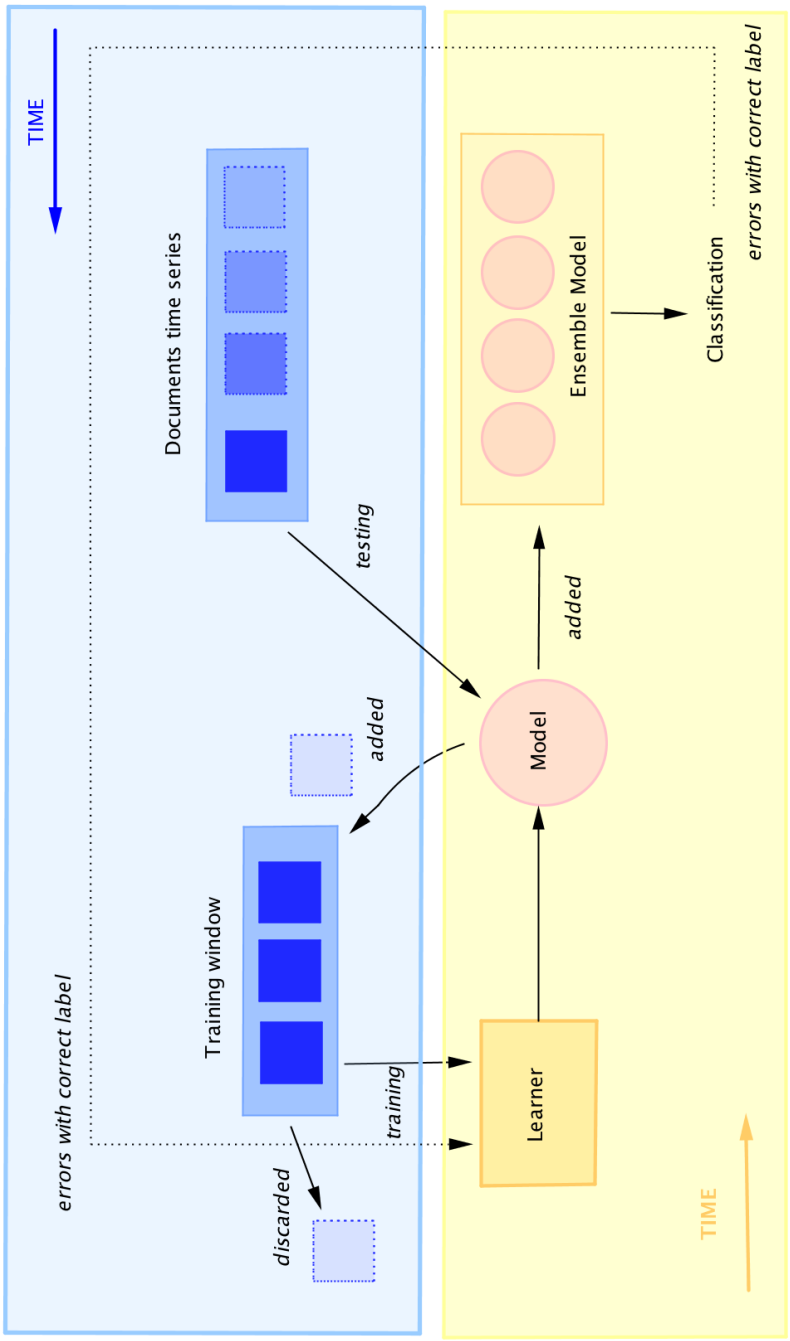

Figure 7.2: Ensemble model enrichment: the use of misclassified examples

We evaluated the performance by classifying the dataset presented in Section 6.4 using the two approaches described, namely the *baseline model approach* and the *enriched model approach*. In the enriched model we have used a two-fold approach: storing misclassified examples during 4 time-windows, represented as "$Enriched_4$", and storing examples *ad eternum*, named "$Enriched_\infty$". The rationale of storing during 4 time-windows is also based on the results presented in Section 6.5. Table 7.3 summarises the performance results considering the micro-averaged F_1 measure.

Drift	Baseline	$Enriched_4$	$Enriched_\infty$
Sudden #1	74.80%	75.45%	72.37%
Sudden #2	87.80%	88.06%	86.55%
Gradual #1	52.55%	54.82%	89.03%
Gradual #2	62.21%	63.43%	89.21%
Incremental #1	88.58%	88.99%	94.86%
Incremental #2	77.21%	79.26%	74.28%
Reoccurring	35.33%	36.63%	72.42%
Normal #1	70.89%	71.86%	91.89%
Normal #2	90.49%	91.01%	94.69%
Normal #3	81.52%	83.74%	73.60%
Average of micro-averaged F_1	78.27%	79.33%	83.75%

Table 7.3: Comparative results considering micro-averaged F_1

Analysing the table we can observe that globally, and considering the average of the micro-averaged F_1, the storage of the previously misclassified examples improves the overall classification. This is normal and expected as the learning models are trained with more informative examples and this leads to a better performance.

It is particularly important to note that model "$Enriched_4$", which stores relevant examples for 4 time-windows, presents an improvement in the classification performance of all classes, regardless the type of drift they represent. This demonstrates the importance of the misclassified examples to improve the classification performance of the subsequent time-windows.

Nevertheless, when considering storing those examples for longer periods, specially *ad eternum*, one must notice that such improvement is not straightforward. Most classes benefit from storing examples, and we have a significant improvement in the average of the micro-averaged F_1, that increases from 78.27% to 83.27%, but some

classes, namely *Sudden#1*, *Sudden#2*, *Incremental#2* and *Normal#3* have a worse classification performance.

Firstly, both classes that represent a sudden drift have a performance decrease, *Sudden#1* from 74.80% to 72.37% and *Sudden#2* from 87.80% to 86.55%. We are confident that such decrease might be explained by the nature of the drift pattern, as a sudden drift is characterized by an abrupt increase in the frequency of a given class that occurs during a period of time, followed by its disappearance. Storing examples that were misclassified, specially the positive ones that appeared firstly and remained misclassified until the classifier identified them as positive, will delude future classifiers, when the drift pattern is no longer represented. This is in-line with what we have already presented in Section 6.5.2. Secondly, we have identified a performance decrease in the classification of the class that represents *Incremental#2* drift. Similarly to what is happening with the sudden drift, the positive misclassified examples might be contributing to this decline, as the frequency of examples of this class is vanishing in time. Finally, we have the *Normal#3* drift. Differently from the mentioned above, the frequency of this drift is not diminishing in the time series. There is nothing in the drift pattern that allows us to infer what is happening, as it is exclusive to the *Normal#3* drift, and does not appear in the *Normal#1* or *Normal#2*, with the same nature. Although this is a supposition, that must be validated in future work, we do believe that it might be related to the class, that is the hashtag we have chosen to represent it.

One of the possible problems that might arise from our approach is storing examples that are not representative of the class, as we cannot guarantee that a message is well-classified by its hashtag; in fact, we might be propagating errors by storing examples that were misclassified, but, unlike our expectations of storing the most informative ones, we might be propagating those that do not represent the class at all. This is a problem that might arise in a dataset like ours, because it is hard to validate that the Twitter's user who wrote the tweet is using the hashtag correctly.

The results seem promising and validate the usefulness of our strategy: we achieved an improvement of 5% in comparison with the baseline approach, taking into account micro-averaged F_1 mean values.

It is also important to state that we have shown that retaining informative examples during the right amount of time can improve the learners' ability to identify a given

class, regardless of the drift pattern the class is presenting. We do believe that the improvements are problem dependent, even though it is an important insight in dynamic models, as they are particularly difficult learning scenarios. Special attention must be given to classes that tend to disappear. In this particular case, retaining examples for long periods can lead to severe bias due to misclassification.

In the next section we will present a comparative study between our Drift Adaptive Retain Knowledge (DARK) framework and a state-of-the-art solution, namely the Learn++.NSE algorithm.

7.5 Comparative Study with Learn++.NSE

We described the Drift Adaptive Retain Knowledge (DARK) framework in Section 7.2 to tackle adaptive learning in dynamic environments based both on recent and on retained knowledge. DARK handles an ensemble of multiple Support Vector Machine (SVM) models that are dynamically weighted and have distinct training window sizes.

In this section we present a comparative study with benchmark solutions in the field, namely the Learn++.NSE algorithm. We firstly introduce the Learn++.NSE algorithm and then present the experimental results obtained, along with the analysis of those results. Finally, we present a statistical evaluation to validate the obtained results.

7.5.1 Learn++.NSE

Learn++.NSE (NSE stands for Non Stationary Environments) was presented by Ryan Elwell and Robin Polykar in Elwell & Polikar (2011). It is one of the most relevant passive solutions for learning in the presence of drift. According to the authors, the framework is not only able to learn in the presence of drift without explicit knowledge, but can also accommodate a wide variety of drift settings, regardless of the drift nature, which is not common in most learning algorithms that try to deal with dynamic environments.

Learn++.NSE is thus a popular batch-based algorithm that trains one new classifier for each batch of data it receives, and combines these classifiers using a dynamically weighted majority voting. According to its authors, the novelty of the approach is based on determining the voting weights, for each classifier's time-adjusted accuracy

on current and past environments. It aims to act in respect to the changes in underlying data distributions, as well as to the possible reoccurrence of an earlier distribution. In Learn++.NSE all classifiers are maintained and reweighted on the most recent training data, and thus the learning time increases linearly.

Learn++.NSE software implementation for *Matlab*, with a classification and regression tree (CART) base classifier, was made available by authors at the website `http://github.com/gditzler/IncrementalLearning`.

7.5.2 Experimental Results and Analysis

In this section we evaluate the performance obtained with the Twitter dataset presented in Section 6.4 using the DARK framework in comparison with the Learn++.NSE benchmark algorithms.

We have tested both DARK framework and Learn++.NSE with different settings. Firstly, we tested the classifier used by Learn++.NSE authors as base classifier, a classification and regression tree (CART), namely Learn++.CART. We also implemented the SVM as an alternative base classifier, since DARK uses SVM and we could have similar base classifiers in both. As the implementation provided by Learn++.NSE is in *Matlab* code, and the base classifier must report a single output, we have used a multi-class SVM strategy provided by Matlab called *ClassificationECOC*, which is an error-correcting output codes (ECOC) classifier for multi-class learning. This procedure reduces a multi-class problem to multiple, binary classifiers like SVM. As the use of the ECOC classifier introduces some variability, due to random construction of the coding matrix, we have run multiple experiments and have chosen the one that presents the best performance. We have named this approach as Learn++.NSE.SVM. A one-against-all strategy is used both in Learn++.NSE.SVM and in DARK. Secondly, we have set up DARK with two different configurations: an ensemble composed of 4 models, named DARK.4.InverseFN and a solution that retains all models previously created, named DARK.All.InverseFN. The performance metric used to combine the ensemble in DARK was the Inverse False Negative ($\frac{1}{FN}$). The idea of using the Inverse False Negative ($\frac{1}{FN}$) is based on the experiments detailed in Section 7.3, where combining the ensembles' models using this metric resulted in higher classification performance of the ensemble.

All SVM models processed in our experiments have linear kernels, and DARK implements the SVM Light application provided by Joachims (2002) and available at http://svmlight.joachims.org/ with cost-factor equal to 2.

We have tested both algorithms with two distinct setups: using a training window of size 1 and a training window of size 4. The rationale of using training window size equal to 4 is based on what we have presented in Section 6.5 where we have studied the importance of past examples in the classification by presenting a batch learning model that retained previously seen examples during a defined period. By retaining examples during different periods we aimed to evaluate for how long it is relevant to keep information according to the different types of drift. In this way, best tailoring of the memory mechanism needed for classification purposes is attained. A training window size of 4 revealed the best cost benefit tradeoff between performance and computational burden.

Table 7.4 summarises the performance results obtained by classifying the dataset and considering a training window of size 1. Three performance metrics of each model are presented, namely, precision, recall, and F_1. We have highlighted (in bold) the micro-averaged values.

A training window of size 1 means that the models are trained using exclusively the documents arising in the previous time-window. All documents that have been seen prior to that are discarded.

The results presented in Table 7.4 show that Learn++.NSE outperforms DARK, with SVM and CART as base classifiers. In both cases we considered the micro-averaged F_1. DARK scores 74.86% and 75.62%, with an ensemble size of 4 and without pruning, respectively, while Learn++.NSE scores 77.94% and 79.01% with SVM as base classifier and CART, respectively. Considering Learn++.NSE, using CART as the base classifiers outperforms the use of SVM. Even though the average performance of DARK is worse than the average performance of Learn++.NSE in this scenario, one must notice that DARK precision is far superior, 77.94% and 79.01% against 99.14% and 99.12%. It means that even though we miss to classify more positive examples than Learn++.NSE, we had fewer false positives. It is also important to observe that the implementation of Learn++.NSE is set out to achieve the micro-averaged break-even point between precision and recall[1].

[1] As well-known in this case micro-averaged precision is equal to micro-averaged recall.

	Learn++.NSE.SVM			Learn++.NSE.CART			DARK.4.InverseFN			DARK.All.InverseFN		
Drift	**Precision**	**Recall**	F_1	**Precision**	**Recall**	F_1	**Precision**	**Recall**	F_1	**Precision**	**Recall**	F_1
Sudden #1	70.20%	87.50%	77.90%	90.57%	80.65%	85.32%	99.78%	78.80%	88.06%	99.78%	78.80%	88.06%
Sudden #2	65.34%	87.86%	74.94%	92.79%	86.81%	89.70%	99.83%	83.75%	91.09%	99.83%	83.75%	91.09%
Gradual #1	84.91%	63.75%	72.82%	72.64%	61.63%	66.68%	99.90%	40.08%	57.21%	99.90%	40.08%	57.21%
Gradual #2	86.36%	74.92%	80.23%	92.99%	72.46%	81.45%	99.87%	62.88%	77.17%	99.87%	63.17%	77.39%
Incremental #1	83.44%	84.71%	84.07%	94.12%	87.66%	90.77%	99.29%	76.88%	86.66%	99.14%	79.03%	87.95%
Incremental #2	67.69%	80.29%	73.46%	60.57%	84.92%	70.70%	97.03%	49.09%	65.20%	97.27%	54.30%	69.69%
Reoccurring	75.02%	59.87%	66.59%	73.50%	52.33%	61.14%	99.29%	46.73%	63.55%	99.29%	46.73%	63.55%
Normal #1	97.93%	57.25%	72.26%	95.52%	68.08%	79.50%	100.00%	16.69%	28.60%	100.00%	16.69%	28.60%
Normal #2	89.84%	88.75%	89.29%	95.88%	87.45%	91.47%	98.50%	71.66%	82.96%	98.50%	71.72%	83.00%
Normal #3	83.04%	64.81%	72.80%	51.37%	74.31%	60.75%	99.20%	40.11%	57.12%	99.20%	40.11%	57.12%
Micro-averaged F_1	**77.94%**	**77.94%**	**77.94%**	**79.01%**	**79.01%**	**79.01%**	**99.14%**	**60.13%**	**74.86%**	**99.12%**	**61.13%**	**75.62%**

Table 7.4: Performance results for training window size = 1

	Learn++.NSE.SVM			Learn++.NSE.CART			DARK.4.InverseFN			DARK.All.InverseFN		
Drift	**Precision**	**Recall**	F_1	**Precision**	**Recall**	F_1	**Precision**	**Recall**	F_1	**Precision**	**Recall**	F_1
Sudden #1	83.68%	85.78%	84.72%	90.87%	84.10%	87.35%	99.82%	81.73%	89.87%	99.82%	81.73%	89.87%
Sudden #2	76.15%	91.28%	83.03%	97.14%	85.75%	91.09%	99.81%	87.83%	93.44%	99.81%	87.86%	93.46%
Gradual #1	91.43%	71.13%	80.01%	80.19%	70.33%	74.94%	99.70%	54.79%	70.72%	99.70%	55.29%	71.13%
Gradual #2	94.36%	83.71%	88.72%	97.41%	83.00%	89.63%	99.21%	67.71%	80.49%	99.16%	68.83%	81.26%
Incremental #1	93.67%	85.94%	89.64%	97.39%	89.35%	93.20%	99.47%	87.35%	93.02%	99.38%	87.59%	93.11%
Incremental #2	75.99%	92.05%	83.25%	72.04%	89.47%	79.82%	97.86%	71.78%	82.82%	97.89%	74.99%	84.93%
Reoccurring	88.24%	53.00%	66.22%	75.94%	53.67%	62.89%	97.95%	54.07%	69.67%	97.95%	54.07%	69.67%
Normal #1	97.88%	83.96%	90.39%	92.47%	78.87%	85.13%	99.86%	59.27%	74.39%	99.86%	59.47%	74.55%
Normal #2	86.94%	94.38%	90.51%	96.07%	91.76%	93.86%	99.48%	85.70%	92.08%	99.09%	86.06%	92.12%
Normal #3	88.79%	87.57%	88.17%	61.59%	89.58%	73.00%	99.23%	76.08%	86.13%	99.23%	76.43%	86.35%
Micro-averaged F_1	**85.89%**	**85.89%**	**85.89%**	**84.62%**	**84.62%**	**84.62%**	**99.30%**	**76.04%**	**86.13%**	**99.21%**	**76.72%**	**86.53%**

Table 7.5: results for training window size = 4

In the second setting we have used a training window of size 4. Table 7.5 summarises the performance results obtained by classifying the dataset, considering the micro-averaged measures in this new setting. The results revealed that DARK performs 86.13% and 86.53%, micro-averaged F_1, with ensemble size of 4 and without pruning respectively. Both settings outperform Learn++.NSE, with 85.89% of F_1 using an SVM as base classifier, and 84.62% of F_1 using CART as base classifier. It is important to notice that the training dataset is exactly the same for all the algorithms, i.e., the dataset contains all examples from the 4 previously occurring time-windows. The major difference seems to be that DARK makes a better use of the extra information provided when more examples are available, specially longstanding examples. It is also important to note that DARK.4.InverseFN is a lightweight solution, as it discards all the information more than 4 time-windows old. It discards not only the examples but also the models that constitute the ensemble. It is also worth mentioning that in this experiment the insights we have described in Section 7.4 were not applied. In fact, therein we concluded that by retaining misclassified examples could lead to an improvement in the classification performance, which could boost the performance obtained with the DARK framework.

7.5.3 Statistical Evaluation of the Comparative Study

In the previous section we have presented the experimental results of the comparative study of DARK with benchmark solutions in the field, namely Learn++.NSE. In this section we statistically compare both algorithms. When dealing with dynamic environments, one of the major challenges is sample's dependency that undermines the use of validation techniques, like cross-validation. The sample's dependency is due to the classification task at hand, that is time-dependent, as each sample is timestamped and the classification has a temporal context, as it evolves over time. To clarify the dependency of samples, suppose we could alter the order of each example in the time series. It would be easier for classification purposes that the examples of each class appeared consecutively. In this case, the classification would be simpler, as we could just put efforts in the identification of the change, which would be abrupt. However, as easily perceived in the real scenario considered here, this is not possible, as we can not alter the order in which each example is presented to the classifier. Moreover, the examples, that occur prior to the occurrence of that example, influence it, not only

because they might have been sufficiently informative to train a classifier to identify the newly incoming samples, but also because in Twitter, a tweet can trigger similar tweets and retweets. In this case, the research design falls in the category of *matched pairs*, and the samples are not independent.

The most used statistical tests in machine learning aim at assessing a classification algorithm's performance and there are several examples extensively used, e.g., Binomial Test, Approximate Normal Test, and the famous t-Test (Alpaydin, 2010). However, it has been shown in Dietterich (1998) that such tests are not adequate for determining whether a certain learning algorithm outperforms, or not, another one, on a particular learning task.

To this end, the type I error should be taken into account, i.e., their probability of incorrectly detecting a difference between two algorithms when no difference exists. These widely used statistical tests have been shown to have high probability of type I error in certain situations and should not be used, namely (i) a test for difference of two proportions; (ii) a paired-differences t test based on taking several random train-test splits; and (iii) a paired-differences t test based on 10-fold cross-validation (Dietterich, 1998).

As such, these tests become less appropriate when the goal is to compare two classification algorithms, i.e., to test for difference of classification proportion. In this setting less known approaches as McNemar's Test McNemar (1947) that has been used in different scenarios, like Omer *et al.* (2015), van Essen *et al.* (2014) and Simonson *et al.* (2007) are available. We will give further detail in the following subsection.

McNemar's Test

The McNemar's test is used to test for binary matched-pairs data. It is a nonparametric test based on standardized normal test statistic calculated from error matrices of two algorithms, and it is applied to 2×2 contingency tables with a dichotomous trait to determine if there is marginal homogeneity between error frequencies. To apply the McNemar's test and to evaluate the performance of two different algorithms in the same sample, the available data sample must be partitioned into training set and test set, and both algorithms must share not only the same training set but also the same testing set.

After the classification task, the following contingency table should be constructed considering the algorithms 1 and 2 that one aims to evaluate:

number of examples misclassified by both algorithms (a)	number of examples only misclassified by 1 (c)
number of examples only misclassified by 2 (b)	number of examples correctly classified by both algorithms (d)

Under the null hypothesis, both algorithms should have the same error rate, i.e., $b = c$.

The McNemar's statistic test can be defined as:

$$Z^2 = \frac{(b-c)^2}{b+c}, \tag{7.1}$$

and, for a large number of discordances $(c+b) > 20$, it can be approximate to the chi-squared χ^2 distribution with 1 degree of freedom under the null hypothesis. Z^2 can then be expressed as χ^2.

In Edwards (1948), a continuity correction is proposed, as due to the studies of Yates (1934), when we are dealing with discrete frequencies and if the data are to be evaluated in terms of continuous distributions, either the normal distribution or the chi square distribution, a continuity correction is in order (Edwards, 1948).

The McNemar's statistic test with continuity correction is then:

$$Z^2 = \frac{(|b-c|-1)^2}{b+c}, \tag{7.2}$$

Considering the χ^2 distribution, the difference in performance is statistically significant ($p \leq 0.05$) if χ^2 is greater than 3.841.

Statistical Results and Analysis

To evaluate the results obtained with the comparative study and to assert the classification differences between both algorithms, Learn++ and DARK, we have performed the McNemar's test. We have compared Learn++.NSE, with both base classifiers SVM and CART, with the two DARK configurations: DARK.4.InverseFN and DARK.All.InverseFN.

Table 7.6 summarises the classification differences obtained by classifying the dataset with Learn++.NSE.SVM and with DARK, considering a training window of size 1. The values defined in the contingency table used by the McNemar's test are presented, namely the examples misclassified by both algorithms (M. by both), the examples only misclassified by DARK (M. by DARK), the examples only misclassified by Learn++.NSE.SVM (M. by Learn++.NSE.SVM), and finally the examples that were not misclassified by any of them (Not M.). We have presented the results considering the classification of each drift type, but we also counted the correct and incorrect classification of each example in each class to evaluate the performance of the micro-averaged results as shown in Table 7.6. We highlighted (in bold) the McNemar χ^2 values.

The results show that there are significant differences in the classification of each class by both algorithms, with the exception of the drift type Gradual #2, that scores 0.06 in the McNemar's χ^2 against DARK.4.InverseFN and 0.00 against DARK.All.InverseFN. Considering the overall results, and the classification of each example, the McNemar's χ^2 scores 106.88 and 172.41. According to the McNemar's test, if the null hypothesis is correct, i.e., there is marginal homogeneity between samples and both algorithms have the same error rate, then the probability of the McNemar's test value to be greater than $\chi^2 = 3.841$ would be less than 0.05.

Table 7.7 summarises the classification differences obtained by classifying the dataset again with Learn++.NSE.SVM and DARK considering a training window of size 4. Unlike in the previously shown results, the drift type where there is not significant differences is Normal #3. In all of the other drift types including the micro-averaged results, we reject the null hypothesis of the McNemar's test, and attest that there are significant differences in the classification provided by both algorithms at the level of significance of 5%.

Identical statistical settings where performed with Learn++.NSE with CART as base classifier, Learn++.NSE.CART. Table 7.8 summarises the McNemar's test to evaluate the relative performance between Learn++.NSE.CART and DARK, considering now a training window of size 1.

The results show that the classification of drift Gradual #1 is the only one where it is not possible to ensure a significant difference between the performance of both algorithms, as McNemar's χ^2 is 1.24, and thus less than 3.841. In all of the other drifts and, particularly, in the micro-averaged results that embody the overall performance

	DARK.4.InverseFN					DARK.All.InverseFN				
	M. by both	**M. by DARK**	**M. by Learn++.NSE.SVM**	**Not M.**	**McNemmar χ^2**	**M. by both**	**M. by DARK**	**M. by Learn++.NSE.SVM**	**Not M.**	**McNemmar χ^2**
Sudden #1	507	348	1479	31716	**698.91**	507	348	1479	31716	**698.91**
Sudden #2	291	299	1824	31636	**1094.01**	291	299	1824	31636	**1094.01**
Gradual #1	817	622	325	32286	**92.52**	817	622	325	32286	**92.52**
Gradual #2	571	322	315	32842	**0.06**	565	321	321	32843	**0.00**
Incremental #1	593	483	866	32108	**108.17**	560	424	899	32167	**169.82**
Incremental #2	833	1502	1752	29963	**19.05**	786	1318	1799	30147	**73.92**
Reoccurring	560	244	341	32905	**15.75**	560	244	341	32905	**15.75**
Normal #1	1025	1037	63	31925	**860.66**	1025	1037	63	31925	**860.66**
Normal #2	524	933	530	32063	**110.46**	524	930	530	32066	**109.04**
Normal #2	1180	1063	624	31183	**113.72**	1180	1063	624	31183	**113.72**
Σ	6901	6853	8119	318627	**106.88**	6815	6606	8205	318874	**172.41**

Table 7.6: McNemar's test for the results of Learn++.NSE.SVM against DARK strategies for training window size = 1

	DARK.4.InverseFN					DARK.All.InverseFN				
	M. by both	**M. by DARK**	**M. by Learn++.NSE.SVM**	**Not M.**	**McNemmar χ^2**	**M. by both**	**M. by DARK**	**M. by Learn++.NSE.SVM**	**Not M.**	**McNemmar χ^2**
Sudden #1	530	207	708	32605	**273.22**	530	207	708	32605	**273.22**
Sudden #2	271	173	1072	32534	**647.71**	271	172	1072	32535	**649.68**
Gradual #1	643	446	210	32751	**84.18**	636	441	217	32756	**75.58**
Gradual #2	388	400	123	33139	**145.65**	383	379	128	33160	**123.27**
Incremental #1	364	232	539	32915	**121.45**	367	222	536	32925	**129.25**
Incremental #2	381	946	1269	31454	**46.81**	370	816	1280	31584	**102.28**
Reoccurring	595	111	216	33128	**33.08**	595	111	216	33128	**33.08**
Normal #1	376	634	66	32974	**459.27**	376	629	66	32979	**454.45**
Normal #2	242	488	738	32582	**50.57**	241	488	739	32582	**50.94**
Normal #3	415	498	460	32677	**1.43**	416	484	459	32691	**0.61**
Σ	4205	4135	5401	326759	**167.81**	4185	3949	5421	326945	**230.93**

Table 7.7: McNemar's test for the results of Learn++.NSE.SVM against DARK strategies for training window size = 4

of the algorithms regarding all classified examples, despite which drift pattern they represent, we reject the null hypothesis of the McNemar's test, and verify that there are significant differences in the classification provided by both algorithms.

Finally, Table 7.9 presents the results of the McNemar's test concerning the relative performance between Learn++.NSE.CART and DARK with a training window of size 4. In the drifts Gradual #1 and Incremental #1, there are no significant classification differences between both algorithms, with McNemar's χ^2 of 1.69 and 0.01, respectively, when comparing Learn++.NSE.CART and DARK.4.InverseFN at a significance level of 5%. Similarly, the McNemar's χ^2 is, respectively, 2.89 and 0.02, when comparing Learn++.NSE.CART and DARK.All.InverseFN at the same level of significance. Still, the McNemar's χ^2 regarding the micro-averaged results allows to reject the null hypothesis, attesting significant differences in the performance of both algorithms at 5% of significance level.

7.6 Conclusion

In this chapter we focused on adaptive learning strategies to learn in dynamic environments. We presented the Drift Adaptive Retain Knowledge (DARK) framework, that sets out to effectively learn in dynamic environments in text classification scenarios. We have also presented two experiments, one that proposes different combining strategies for combining ensembles, whose aim is to learn in dynamic environments, and another that evaluates the relevance of some examples in regard to others in the same environments. Finally, a comparative study of DARK with a state-of-the-art solution, namely the Learn++.NSE algorithm, was presented.

We achieved four contributions in this chapter: (i) to evaluate the effect of using different combining strategies for combining ensembles in text-based dynamic environments; (ii) to infer about the influence, for the overall learning and classification performances, of having recent examples retained during different time lengths, that is, to analyse the impact of using longstanding examples by the classifiers models and how they can contribute positively to the ensemble classification; (iii) to validate the DARK framework with text classification based on the Twitter social network public stream and (iv) to present a comparative study with a benchmark solution in the field, namely the Learn++.NSE algorithm.

	DARK.4.InverseFN					DARK.All.InverseFN				
	M. by both	**M. by DARK**	**M. by Learn++.NSE.CART**	**Not M.**	**McNemmar χ^2**	**M. by both**	**M. by DARK**	**M. by Learn++.NSE.CART**	**Not M.**	**McNemmar χ^2**
Sudden #1	667	188	443	32752	**102.24**	667	188	443	32752	**102.24**
Sudden #2	405	185	313	33147	**32.39**	405	185	313	33147	**32.39**
Gradual #1	874	565	604	32007	**1.24**	874	565	604	32007	**1.24**
Gradual #2	637	256	155	33002	**24.33**	631	255	161	33003	**20.79**
Incremental #1	488	588	322	32652	**77.17**	473	511	337	32729	**35.29**
Incremental #2	701	1634	2434	29281	**156.93**	667	1437	2468	29478	**271.68**
Reoccurring	631	173	367	32879	**68.98**	631	173	367	32879	**68.98**
Normal #1	765	1297	104	31884	**1014.18**	765	1297	104	31884	**1014.18**
Normal #2	586	871	221	32372	**385.72**	586	868	221	32375	**383.21**
Normal #3	862	1381	2715	29092	**433.81**	862	1381	2715	29092	**433.81**
Σ	6616	7138	7678	319068	**19.61**	6561	6860	7733	319346	**52.11**

Table 7.8: McNemar's test for the results of Learn++.NSE.CART against DARK strategies for training window size = 1

	DARK.4.InverseFN					DARK.All.InverseFN				
	M. by both	**M. by DARK**	**M. by Learn++.NSE.CART**	**Not M.**	**McNemmar χ^2**	**M. by both**	**M. by DARK**	**M. by Learn++.NSE.CART**	**Not M.**	**McNemmar χ^2**
Sudden #1	586	151	388	32925	**103.33**	586	151	388	32925	**103.33**
Sudden #2	382	62	222	33384	**89.02**	383	60	221	33386	**91.10**
Gradual #1	658	431	471	32490	**1.69**	653	424	476	32497	**2.89**
Gradual #2	400	388	61	33201	**236.69**	395	367	66	33222	**207.85**
Incremental #1	319	277	274	33180	**0.01**	319	270	274	33187	**0.02**
Incremental #2	470	857	1546	31177	**196.98**	457	729	1559	31305	**300.37**
Reoccurring	612	94	338	33006	**136.69**	612	94	338	33006	**136.69**
Normal #1	480	530	202	32838	**146.08**	480	525	202	32843	**142.62**
Normal #2	358	372	236	33084	**29.98**	355	374	239	33082	**29.29**
Normal #3	313	600	2156	30981	**877.37**	313	587	2156	30994	**896.33**
Σ	4578	3762	5894	326266	**470.29**	4553	3581	5919	326447	**574.90**

Table 7.9: McNemar's test for the results of Learn++.NSE.CART against DARK strategies for training window size = 4

The evaluation of the DARK framework demands a deep investment on ensemble frameworks' deployment for learning. It was useful to include flexible and adjustable training window sizes to retain recent knowledge. We were thus able to contribute through DARK, as we may test datasets processing with different training window sizes of recent examples in order to infer about its impact on ensemble classification.

Regarding experimental results obtained with a state-of-the-art algorithm, namely the Learn++.NSE, we have processed the same dataset with DARK framework and Learn++.NSE algorithm, being the latter processed with both SVM and CART as base classifiers. For both processings we have used two distinct window sizes and the results calculated for micro-averaged F_1 revealed that as we increased the window size, the DARK framework became more precise and accurate, which demonstrates its potential as a learning strategy in dynamic environments. The statistical evaluation we have performed with the use of the McNemar's test validates the obtained results, attesting the significant difference in the performance of both algorithms with 95% of confidence.

Part IV

Conclusions and Future Work

Chapter 8

Conclusion

8.1 Introduction

In the previous chapters we have presented a number of novel techniques to tackle the problem of effectively learning in dynamic environments, mainly through the use of crowds. The extraordinary and relevant amount of data made available today by social networks, a real-world dynamic environment, may be used towards the resolution of challenges faced by individuals and companies. However, adaptive strategies must be proposed and put forward, as static models do not have the ability to constantly learn.

The contributions presented in this thesis integrated knowledge from multiple fields, like information extraction, particularly text classification, crowdsourcing, social networks, active learning, and adaptive models, particularly ensembles.

We have studied the impact of using crowdsourcing as data source, using a humour classification dataset, and the effect regarding the use of active learning to improve classification in such highly subjective task.

Concerning learning in dynamic environments, social networks were used as case study, particularly Twitter, and specific preprocessing strategies were presented, in order to avoid the bias associated with the multitude of users with different social and educational contexts.

The Drift Oriented Tool System (DOTS) was also presented, allowing for the definition and generation of text-based datasets. Another important contribution is the Drift Adaptive Retain Knowledge (DARK) framework, that aims to effectively learn in dynamic environments in text classification scenarios.

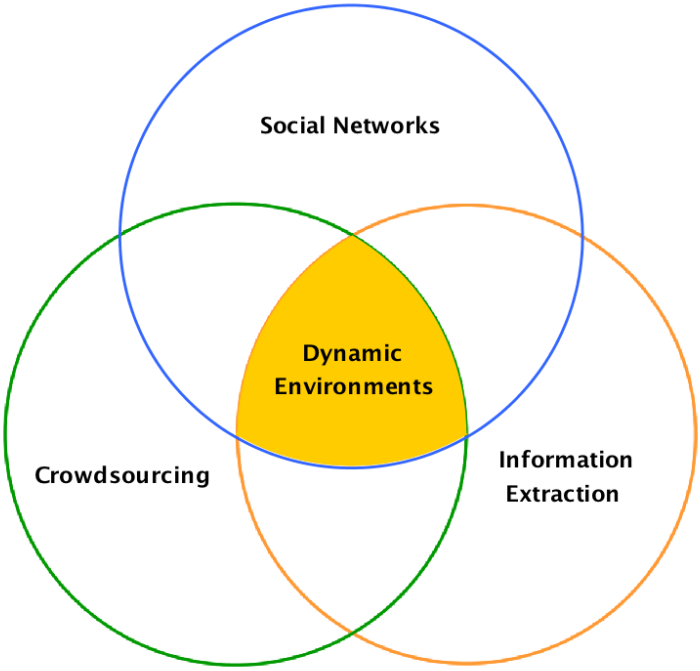

Figure 8.1: Context and framing of the thesis

In the next section, we will discuss the contributions made and present some conclusions regarding this work. Finally, future work will be addressed by defining future research directions that can be explored in further investigations regarding adaptive machine learning strategies.

8.2 Discussion and Conclusions

In this thesis we have presented innovative and relevant contributions regarding information extraction from crowds in dynamic environments. We have used social networks as our main case study, due to their intrinsic potential and the fact that they constitute text based scenarios ripe to explore.

Figure 8.1 frames the thesis context. It represents the broader areas we have engaged, and the overlap between them. Dynamic environments are the core of those broader areas, as information extraction from crowds in social networks needs to deal with the learning process in an environment with constant changes.

Considering the work done in the above mentioned areas, the main contributions of this thesis research work can be summarized as follows:

- **Crowdsourcing as data source**.

  There are real-world problems where it is infeasible to have an assertive supervisor. Considering the potential of crowdsourcing as data source, we have experienced an active learning strategy to compare the use of an assertive supervisor versus the use of crowdsourcing. A humour classification problem was used as a case study, as it is an extremely subjective task, particularly difficult to a machine learning algorithm to cope with. The experiments revealed that, though an assertive supervisor still outperforms a crowd, as it never fails, crowdsourcing can be used as a solution, as it improves the classification performance considering the baseline approach, where crowdsourcing was not used.

- **Active learning for customization purposes**.

  Model customization is an important challenge in machine learning. It is specially relevant in recommendation applications where users' preferences are dependent on social, cultural, and emotional background. Crowdsourcing can be suited for model customization, if we can define the best suited crowd for performing a given task. Our contribution in this work was to use active learning for customization purposes. Active learning allowed us to integrate the user feedback into the learning process, and thus defining the customized crowd that will be used to model customization. The results revealed the usefulness of using crowds in the adjustment of user preferences.

- **Preprocessing in social networks**.

  Social network posts are characterized by being small and freely introduced by users. They are dependent on age, gender, location, and social upbringing, and thus become easily biased. However, and considering the importance of extracting information from such media, specific preprocessing strategies must be put forward to tackle this specificity. We have proposed the definition of semantic meta-hashtags that cluster similar messages, and thus are able to deal with the aforementioned bias, and improve classification performance. This improvement sustains the use of meta-hashtags and makes it possible to infer that we avoided

the classification problems raised by having different classes representing the same concept.

- **The impact of longstanding messages in classification**.

  We have studied the impact of longstanding messages in classification using Twitter as a case study. The proposed approach used different training window sizes to understand the possibility of achieving the best balance between the classifier performance and the computational effort needed in the training phase. The usefulness of this contribution is to understand how informative can past events be to current learning models, and for how long it is relevant to store previously seen information, which is particularly important in dynamic environments.

- **Learning in dynamic environments**.

  Modern challenges in machine learning include non-stationary environments. Due to their dynamic nature, learning in these environments must take into account model adaptation, as there are significant changes between data distributions used to define models and those responsible for generating new data. Three different models to learn in dynamic environments were proposed: a time-window model, an ensemble-based model, and an incremental model. The time-window model is characterized by taking into account recent information in a given a time-window. The ensemble model is based on the idea that the use of a committee of classifiers can provide better results than the best of the single classifiers, if correctly combined. Finally, the incremental model is characterized by retaining in a single classifier all the information gathered over time. We have also presented a benchmark dataset regarding drift in Twitter, where real tweets were artificially timestamped in order to represent different drift patterns.

- **Drift Oriented Tool System (DOTS)**.

  The research efforts concerning dynamic environments are mostly in proposing, implementing, and testing innovative solutions to cope with the drifting scenario. However, building drift aware datasets with blended temporal distributions is also a challenging task. Finding acceptable benchmarks for these environments is not straightforward, and so we have contributed with the Drift Oriented Tool System (DOTS), a framework that allows for the definition and generation of text-based

datasets, and can also be used to simulate a set of different drift patterns with a temporal basis. The datasets obtained with DOTS can be used to evaluate and validate learning strategies used in dynamic environments.

- **Drift Adaptive Retain Knowledge (DARK)**.

  Concerning the relevance of learning in dynamic environments, such as allowing information extraction in scenarios, like social networks, we have presented DARK, the Drift Adaptive Retain Knowledge framework. DARK uses an ensemble of classifiers with dynamic weighting schemes to integrate multiple experts with different characteristics, and thus benefit from their multitude. Another important feature is allowing variable training window sizes for model adaptation in incremental learning. We have also evaluated the effect of using different combining schemes in the performance obtained by DARK framework in the Twitter classification problem, such as using performance metrics, like accuracy, F_1 measure, the inverse of false positives ($\frac{1}{FP}$), and the inverse of false negatives ($\frac{1}{FN}$), and have compared it with a benchmark solution in the field, namely the Learn++.NSE algorithm. Experimental results revealed that DARK outperforms Learn++.NSE.

8.3 Future Work

Our work reveals that novel approaches can be developed, and there are further directions that be worth exploring in the near future. We will describe those that we consider the most relevant:

- **To integrate ontology-related information in social networks**.

  We have explored the concept of meta-hashtags, by clustering similar hashtags in order to avoid the bias introduced by different user contexts, and thus improve classification performance. However, in the presented work we have heuristically defined the meta-hashtags. In future work it would be worth exploring the integration of ontology-related information, based on, for instance, the wordnet, so relations between hashtags could be more easily perceived. Along with improving the definition of meta-hashtags, it would also be interesting to study specific preprocessing methods for social network documents. Social network documents,

due to their specificity, like being small and freely introduced without any major rules, could benefit from adapted preprocessing strategies. There are some proposals regarding the preprocessing to classification of SMS documents that could be adapted and used in this new context.

- **Recommendation in Twitter**.

  Considering the impact of social networks in the daily routine of Internet users, and specially the importance they have gained as an information source, recommendation in Twitter is being considered an interesting research path, as it can potentiate service customization to user profiles. We have used a humour classification dataset as a case study regarding the use of crowdsourcing, and have presented some experiments regarding joke recommendation, but Twitter recommendation was not explored. The path that we propose is to merge sentiment analysis techniques in Twitter with our adaptive learning strategy, so model customization in Twitter could also deal with the intrinsic dynamic nature of Twitter, and thus better be up to date in the provided recommendations. Regarding drift in Twitter, it would also be important to extend the study of drift patterns in Twitter, so a dataset with real data and identified drift patterns would be possible to create in a near future.

- **Adaptive time-windows in DARK**.

  DARK outperformed the benchmark algorithm Learn++.NSE in the real-world dataset we have proposed, the dataset for drift in Twitter. DARK allows variable training window sizes for model adaptation. Nevertheless, the variable training window size is not adaptive, i.e., we can define which is the size of the training window, but during all the learning process it remains the same and independent from the drift pattern. One of the improvements we would like to implement in the future is an adaptive training window size. The adaptive training window size might not increase the classification performance of the framework, but might optimize the ratio between the computational burden of storing examples for training purposes, and the model performance. It would also be interesting to study the effect of the adaptive training window size in each drift, as due to their intrinsic nature they react differently to the enlargement of the training window

size, as we already concluded by the experiments we have done regarding the effect of time in dynamic environments.

- **Advanced analysis in ensemble modelling in social networks**.

  Regarding the use of ensembles to learn in dynamic environments, particularly in Twitter, there are different paths that are worth exploring. We have contributed by studying the impact of combining individual ensemble classifiers by weighting their future contribution to the ensemble based on their previous performance. On the other hand, we have actively selected examples based on their ability to effectively contribute to the performance in classifying drifting concepts. Both strategies were not combined so far. It might be worth exploring the use of different performance metrics to determine each classifier's in the ensemble, and each example's importance, and hence their lifetime, in the learning process. More complex performance metrics could also be explored, as could a pruning strategy to the examples that are retained. It is important to evaluate if in a given moment an example loses its ability to positively contribute to boost the classification performance of the ensemble, and thus should be discarded.

Part V

Appendix

Appendix A

Published Materials

A.1 Journal Papers

J1. Costa, J. and Silva, C. and Antunes, M. and Ribeiro, B. , "Adaptive Learning for Dynamic Environments: a Comparative Approach", Engineering Applications of Artificial Intelligence, Elsevier, vol. 65, pp. 336-345, 2017 (https://doi.org/10.1016/j.engappai.2017.08.004) (IF=2.894)

J2. Costa, J. and Silva, C. and Antunes, M. and Ribeiro, B. , "Customized Crowds and Active Learning to Improve Classification", Expert Systems With Applications, Elsevier, vol. 40, pp. 7212-7219, 2013 (https://doi.org/10.1016/j.eswa.2013.06.072) (IF=2.240)

A.2 Conference Papers

C1. Oliveira, N. and Costa, J. and Silva, C. and Ribeiro, B. , "On the importance of Users' Features in Retweeting", in 23rd Portuguese Conference on Pattern Recognition (RecPad), Lisboa, Portugal, October 2017

C2. Costa, J. and Silva, C. and Antunes, M. and Ribeiro, B. , "Performance Metrics for Model Fusion in Twitter Data Drifts", in 8th Iberian Conference on Pattern Recognition and Image Analysis (IbPria), pp. 13-21, Faro, Portugal, June 2017. Springer (https://doi.org/10.1007/978-3-319-58838-4_2)

C3. Costa, J. and Silva, C. and Antunes, M. and Ribeiro, B. , "Twitter classification: are some examples better than others?", in 22nd Portuguese Conference on Pattern Recognition (RecPad), Aveiro, Portugal, October 2016 (*Best Paper Award*)

C4. Costa, J. and Silva, C. and Antunes, M. and Ribeiro, B. , "Choice of Best Samples for Building Ensembles in Dynamic Environments", in 17th International Conference on Engineering Applications of Neural Networks (EANN), pp. 35-47, Aberdeen, UK, September 2016. Springer (https://doi.org/10.1007/978-3-319-44188-7_3)

C5. Costa, J. and Silva, C. and Antunes, M. and Ribeiro, B. , "DOTS: Drift Oriented Tool System", in 22nd International Conference on Neural Information Processing (ICONIP), pp. 615-623, Istambul, Turkey, November 2015. Springer (https://doi.org/10.1007/978-3-319-26561-2_72)

C6. Costa, J. and Silva, C. and Antunes, M. and Ribeiro, B. , "Generating datasets with drift", in 21th Portuguese Conference on Pattern Recognition (RecPad), Faro, Portugal, October 2015

C7. Silva, C. and Antunes, M. and Costa, J. and Ribeiro, B. , "Active Manifold Learning with Twitter Big Data", in International Neural Network Society Conference on Big Data (INNS), pp. 208-215, San Francisco, CA, USA, August 2015. Elsevier (https://doi.org/10.1016/j.procs.2015.07.296)

C8. Costa, J. and Silva, C. and Antunes, M. and Ribeiro, B. , "The Impact of Long-standing Messages in Micro-Blogging Classification", in IEEE International Joint Conference on Neural Networks (IJCNN), pp. 1-8, Killarney, Ireland, July 2015. IEEE (https://doi.org/10.1109/IJCNN.2015.7280731)

C9. Costa, J. and Silva, C. and Antunes, M. and Ribeiro, B. , "Concept Drift Awareness in Twitter Streams", in IEEE International Conference on Machine Learning and Applications (ICMLA), pp.294-299, Detroit, MI, USA, December 2014 (https://doi.org/10.1109/ICMLA.2014.53). IEEE

C10. Costa, J. and Silva, C. and Antunes, M. and Ribeiro, B. , "Learning with Drift in Twitter", in 20th Portuguese Conference on Pattern Recognition (RecPad), Covilha, Portugal, October 2014

C11. Costa, J. and Silva, C. and Ribeiro, B. and Antunes, M. , "CrowdTargeting: Making Crowds More Personal", in 8th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP), pp. 21-26, Bayonne, France, December 2013. IEEE (https://doi.org/10.1109/SMAP.2013.20)

C12. Costa, J. and Silva, C. and Antunes, M. and Ribeiro, B. , "Defining Semantic Meta-Hashtags for Twitter Classification", in 11th International Conference on Adaptive and Natural Computing Algorithms (ICANNGA), pp. 226–235, Lausanne, Switzerland, April 2013. Springer (https://doi.org/10.1007/978-3-642-37213-1_24)

C13. Costa, J. and Silva, C. and Antunes, M. and Ribeiro, B., "On Using Crowdsourcing and Active Learning to Improve Classification Performance", in 11th International Conference on Intelligent Systems Design and Applications (ISDA), pp. 469-474, Cordoba, Spain, November 2011. IEEE (https://doi.org/10.1109/ISDA.2011.6121700)

C14. Costa, J. and Silva, C. and Antunes, M. and Ribeiro, B. , "Get Your Jokes Right: Ask The Crowd", in 1st International Conference on Model and Data Engineering (MEDI), pp. 178-185, Obidos, Portugal, September 2011. Springer (https://doi.org/10.1007/978-3-642-24443-8_20)

C15. Costa, J. and Silva, C. and Antunes, M. and Ribeiro, B. , "The Importance of Precision in humour Classification", in 12th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL), pp.271-278, Norwich, UK, September 2011. Springer (https://doi.org/10.1007/978-3-642-23878-9_33)

A.3 Software

S1. DOTS - Drift Oriented Tool System
Open-source software that allows users to create multiple text classification

datasets with drift patterns. Text preprocessing methods can also be applied and export file types include SVMLight, ARFF (Weka) and CSV.
(http://dotspt.sourceforge.net)

References

Adamic, Lada A., Zhang, Jun, Bakshy, Eytan, & Ackerman, Mark S. 2008. Knowledge Sharing and Yahoo Answers: Everyone Knows Something. *Pages 665–674 of: Proceedings of the 17th International Conference on World Wide Web.*

Agarwal, Apoorv, Xie, Boyi, Vovsha, Ilia, Rambow, Owen, & Passonneau, Rebecca. 2011. Sentiment analysis of Twitter data. *Pages 30–38 of: Proceedings of the Workshop on Languages in Social Media.*

Ahsan, MN Istiaq, Nahian, Tamzid, Kafi, Abdullah All, Hossain, Md Ismail, & Shah, Faisal Muhammad. 2016. Review spam detection using active learning. *Pages 1–7 of: Proceedings of the 7th Annual Conference on Information Technology, Electronics and Mobile Communication.*

Alippi, Cesare, & Roveri, Manuel. 2008. Just-in-time adaptive classifiers - Part II: Designing the classifier. *IEEE Transactions on Neural Networks*, **19**(12), 2053–2064.

Alippi, Cesare, Boracchi, Giacomo, & Roveri, Manuel. 2009. Just in time classifiers: managing the slow drift case. *Pages 114–120 of: Proceedings of the 2009 International Joint Conference on Neural Networks.*

Alippi, Cesare, Boracchi, Giacomo, & Roveri, Manuel. 2012. Just-in-time ensemble of classifiers. *Pages 1–8 of: Proceedings of the 2002 International Joint Conference on Neural Networks (IJCNN).*

Alippi, Cesare, Boracchi, Giacomo, & Roveri, Manuel. 2013. Just-in-time classifiers for recurrent concepts. *IEEE Transactions on Neural Networks and Learning Systems*, **24**(4), 620–634.

REFERENCES

Alpaydin, Ethem. 2010. *Introduction to Machine Learning.* The MIT Press Cambridge, Massachusetts London, England.

Alsaedi, Nasser, Burnap, Pete, & Rana, Omer. 2017. Can We Predict a Riot? Disruptive Event Detection Using Twitter. *ACM Transactions on Internet Technology*, **17**(2), 18:1–18:26.

Antunes, M., Silva, C., Ribeiro, B., & Correia, M. 2011. A Hybrid AIS-SVM Ensemble Approach for Text Classification. *Pages 342–352 of: Proceedings of the 10th International Conference on Adaptive and Natural Computing Algorithms.*

Armitage, P. 1960. *Sequential Medical Trials.* Blackwell Scientific Publications.

Atefeh, Farzindar, & Khreich, Wael. 2015. A Survey of Techniques for Event Detection in Twitter. *Computational Intelligence*, **31**(1), 132–164.

Attea, Bara'a A., & Khoder, Haidar S. 2016. A new multi-objective evolutionary framework for community mining in dynamic social networks. *Swarm and Evolutionary Computation*, **31**, 90 – 109.

Ayeldeen, H., Hassanien, A.E., & Fahmy, A.A. 2013. Fuzzy clustering and categorization of text documents. *In: Proceedings of the 13th International Conference on Hybrid Intelligent Systems (HIS).*

Bagul, Rajni David, & Phulpagar, B. D. 2016. Survey on Approaches, Problems and Applications of Ensemble of Classifiers. *International Journal of Emerging Trends & Technology in Computer Science*, **5**(1), 28–30.

Baram, Yoram, El-Yaniv, Ran, & Luz, Kobi. 2003. Online Choice of Active Learning Algorithms. *Pages 19–26 of: Proceedings of the 20th International Conference on Machine Learning.*

Barr, J, & Cabrera, L F. 2006. AI gets a brain. *Queue*, **4**(4), 24–29.

Bifet, Albert, & Frank, Eibe. 2010. Sentiment Knowledge Discovery in Twitter Streaming Data. *Pages 1–15 of: Proceedings of the 13th International Conference on Discovery Science.*

Bifet, Albert, & Gavalda, Ricard. 2007. Learning from Time-Changing Data with Adaptive Windowing. *In: Proceedings of the 2007 SIAM International Conference on Data Mining.*

Binsted, Kim, & Ritchie, Graeme. 1994. *An implemented model of punning riddles.* Tech. rept. Department of Artificial Intelligence, University of Edinburgh.

Brabham, Daren C. 2008. Crowdsourcing as a Model for Problem Solving: An Introduction and Cases. *Convergence: The International Journal of Research into New Media Technologies*, **14**(1), 75–90.

Brabham, Daren C. 2013. *Crowdsourcing.* The MIT Press.

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

Breiman, L. 2001. Random forests. *Machine Learning*, **45**(1), 5–32.

Brew, A., Greene, D., & Cunnigham, P. 2010. The interaction between supervised learning and crowdsourcing. *In: Proceedings of the NIPS Workshop on Computational Social Science and the Wisdom of Crowds.*

Cambria, Erik. 2016. Affective computing and sentiment analysis. *IEEE Intelligent Systems*, **31**(2), 102–107.

Carreras, Xavier, & Màrquez, Lluís. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. *Pages 152–164 of: Proceedings of the 9th Conference on Computational Natural Language Learning.*

Chang, Hsia-Ching. 2010. A new perspective on Twitter hashtag use: diffusion of innovation theory. *Pages 85:1–85:4 of: Proceedings of the 73rd Annual Meeting on Navigating Streams in an Information Ecosystem.*

Chawla, Nitesh V., Bowyer, Kevin W., Hall, Lawrence O., & Kegelmeyer, W. Philip. 2002. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, **16**(1), 321–357.

Chen, Y., Hsu, W., & Liao, H. 2011. Learning facial Attributes by Crowdsourcing in Social Media. *Pages 25–26 of: Proceedings of the 20th International Conference companion on World Wide Web.*

Cohen, L., Avrahami, G., Last, M., & Kandel, A. 2008a. Info-fuzzy algorithms for mining dynamic data streams. *Applied Soft Computing*, **8**(4), 1283–1294.

Cohen, Lior, Avrahami-Bakish, Gil, Last, Mark, Kandel, Abraham, & Kipersztok, Oscar. 2008b. Real-time data mining of non-stationary data streams from sensor networks. *Information Fusion*, **9**(3), 344–353.

Cohen, W., & Singer, Y. 1999. Context-Sensitive Learning Methods for Text Categorization. *ACM Transactions on Information Systems*, **17**(2), 141–173.

Collobert, Ronan, & Weston, Jason. 2008. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. *Pages 160–167 of: Proceedings of the 25th International Conference on Machine Learning*.

Cordeiro, Mário, & Gama, João. 2016. *Online Social Networks Event Detection: A Survey*. Springer International Publishing. Pages 1–41.

Costa, Joana, Silva, Catarina, Antunes, Mário, & Ribeiro, Bernardete. 2011a. Get your jokes right: ask the crowd. *Pages 178–185 of: Proceedings of the 1st International Conference on Model and Data Engineering*. Springer.

Costa, Joana, Silva, Catarina, Antunes, Mário, & Ribeiro, Bernardete. 2011b. The importance of precision in humour classification. *Pages 271–278 of: Proceedings of the 12th International Conference on Intelligent Data Engineering and Automated Learning*. Springer.

Costa, Joana, Silva, Catarina, Antunes, Mário, & Ribeiro, Bernardete. 2011c. On using crowdsourcing and active learning to improve classification performance. *Pages 469–474 of: Proceedings of the 11th International Conference on Intelligent Systems Design and Applications*. IEEE.

Costa, Joana, Silva, Catarina, Ribeiro, Bernardete, & Antunes, Mário. 2013a. CrowdTargeting: making crowds more personal. *Pages 21–26 of: Proceedings of the 8th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP)*. IEEE.

Costa, Joana, Silva, Catarina, Antunes, Mário, & Ribeiro, Bernardete. 2013b. Customized crowds and active learning to improve classification. *Expert Systems with Applications*, **40**(18), 7212 – 7219.

Costa, Joana, Silva, Catarina, Antunes, Mário, & Ribeiro, Bernardete. 2013c. Defining semantic meta-hashtags for Twitter classification. *Pages 226–235 of: Proceedings of the 11th International Conference on Adaptive and Natural Computing Algorithms*, vol. 7824. Springer.

Costa, Joana, Silva, Catarina, Antunes, Mário, & Ribeiro, Bernardete. 2014. Concept drift awareness in Twitter streams. *Pages 294–299 of: Proceedings of the 13th International Conference on Machine Learning and Applications*. IEEE.

Costa, Joana, Silva, Catarina, Antunes, Mário, & Ribeiro, Bernardete. 2015a. DOTS: Drift Oriented Tool System. *Pages 615–623 of: Proceedings of the 22nd International Conference on Neural Information Processing (ICONIP)*. Springer.

Costa, Joana, Silva, Catarina, Antunes, Mário, & Ribeiro, Bernardete. 2015b. The impact of longstanding messages in micro-blogging classification. *Pages 1–8 of: Proceedings of the International Joint Conference on Neural Networks (IJCNN)*. IEEE.

Costa, Joana, Silva, Catarina, Antunes, Mário, & Ribeiro, Bernardete. 2016. Choice of best samples for building ensembles in dynamic environments. *Pages 35–47 of: Proceedings of the 17th International Conference on Engineering Applications of Neural Networks (EANN)*. Springer.

Costa, Joana, Silva, Catarina, Antunes, Mário, & Ribeiro, Bernardete. 2017a. Adaptive learning for dynamic environments: A comparative approach. *Engineering Applications of Artificial Intelligence*, **65**, 336 – 345.

Costa, Joana, Silva, Catarina, Antunes, Mário, & Ribeiro, Bernardete. 2017b. Performance Metrics for Model Fusion in Twitter Data Drifts. *Pages 13–21 of: Proceedings of the 8th Iberian Conference on Pattern Recognition and Image Analysis*. Springer.

REFERENCES

Cowie, Jim, & Lehnert, Wendy. 1996. Information extraction. *Communications of ACM*, **39**(1), 80–91.

Cox, L.P. 2011. Truth in Crowdsourcing. *Security & Privacy, IEEE*, **9**(5), 74–76.

Cucchiara, Rita, Grana, Costantino, Piccardi, Massimo, & Prati, Andrea. 2003. Detecting moving objects, ghosts, and shadows in video streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **25**(10), 1337–1342.

Dan, Shen. 2004. *Multi-Criteria-Based Active Learning for Named Entity Recognition.* M.Phil. thesis, National University of Singapore.

Das, Sufal, & Kalita, Hemanta Kumar. 2017. Sentiment Analysis for Web-based Big Data: A Survey. *International Journal of Advanced Research in Computer Science*, **8**(5).

Datar, Mayur, & Motwani, Rajeev. 2016. The Sliding-Window Computation Model and Results. *Pages 149–165 of: Data Stream Management.* Springer.

Davis, Joseph G. 2011. From Crowdsourcing to Crowdservicing. *IEEE Internet Computing*, **15**(3), 92–94.

Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., & Harshman, R.A. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science 41*, 391–407.

Dietterich, Thomas G. 1998. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Comput.*, **10**(7).

Ditzler, Gregory, & Polikar, Robi. 2013. Incremental Learning of Concept Drift from Streaming Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, **25**(10), 2283–2301.

Ditzler, Gregory, Roveri, Manuel, Alippi, Cesare, & Polikar, Robi. 2015. Learning in nonstationary environments: a survey. *IEEE Computational Intelligence Magazine*, **10**(4), 12–25.

Doan, Anhai, Ramakrishnan, Raghu, & Halevy, Alon Y. 2011. Crowdsourcing systems on the World-Wide Web. *Communications of the ACM*, **54**(4), 86.

Doerr, Benjamin, Fouz, Mahmoud, & Friedrich, Tobias. 2012. Why rumors spread so quickly in social networks. *Communications of ACM*, **55**(6), 70–75.

Domingos, P., & Hulton, G. 2000. Mining high-speed data streams. *Pages 71–80 of: Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery Data Mining.*

Doulamis, Nikolaos D, Doulamis, Anastasios D, Kokkinos, Panagiotis, & Varvarigos, Emmanouel Manos. 2016. Event detection in twitter microblogging. *IEEE transactions on cybernetics*, **46**(12), 2810–2824.

Duan, Yajuan, Wei, Furu, Zhou, Ming, & Shum, Heung-Yeung. 2012. Graph-based Collective Classification for Tweets. *Pages 2323–2326 of: Proceedings of the 21st International Conference on Information and Knowledge Management.*

Dyer, K.B., Capo, R., & Polikar, R. 2014. COMPOSE: a semisupervised learning framework for initially labeled nonstationary streaming data. *IEEE Transactions on Neural Networks and Learning Systems*, **25**(1), 12–26.

Edwards, Allen L. 1948. Note on the "correction for continuity" in testing the significance of the difference between correlated proportions. *Psychometrika*, **13**(3), 185–187.

Elwell, Ryan, & Polikar, Robi. 2011. Incremental learning of concept drift in nonstationary environments. *IEEE Transactions on Neural Networks*, **22**(10), 1517–1531.

Estellés-Arolas, Enrique, & González-Ladrón-De-Guevara, Fernando. 2012. Towards an Integrated Crowdsourcing Definition. *Journal of Information Science*, **38**(2), 189–200.

Finin, Tim, Murnane, Will, Karandikar, Anand, Keller, Nicholas, Martineau, Justin, & Dredze, Mark. 2010. Annotating Named Entities in Twitter Data with Crowdsourcing. *Pages 80–88 of: Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk.*

Fisk, Raymond P, Patrício, Lia, Ordanini, Andrea, Miceli, Lucia, Pizzetti, Marta, & Parasuraman, A. 2011. Crowd-funding: transforming customers into investors through innovative service platforms. *Journal of service management*, **22**(4), 443–470.

Franklin, M, Kossman, D, & Kraska, T. 2011. CrowdDB: Answering queries with crowdsourcing. *Pages 61–72 of: Proceedings of the 2011 International Conference on Management of data*.

Freund, Y., & Schapire, R. E. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal Computer and Systems Science*, **55**(1), 119–139.

Gama, João, Žliobaitė, Indrė, Bifet, Albert, Pechenizkiy, Mykola, & Bouchachia, Abdelhamid. 2014. A survey on concept drift adaptation. *ACM Computing Surveys*, **46**(4), 44.

Giachanou, Anastasia, & Crestani, Fabio. 2016. Like It or Not: A Survey of Twitter Sentiment Analysis Methods. *ACM Computing Surveys*, **49**(2), 28:1–28:41.

Greengard, Samuel. 2011. Following the crowd. *Communications of the ACM*, **54**(2), 20.

Grossberg, Stephen. 1988. Nonlinear neural networks: Principles, mechanisms, and architectures. *Neural networks*, **1**(1), 17–61.

Haque, Ahsanul, Khan, Latifur, & Baron, Michael. 2015. Semi supervised adaptive framework for classifying evolving data stream. *Pages 383–394 of: Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*.

Haque, Md Enamul, & Alkharobi, Talal M. 2015. Adaptive Hybrid Model for Network Intrusion Detection and Comparison among Machine Learning Algorithms. *International Journal of Machine Learning and Computing*, **5**(1), 17.

Harel, Maayan, Mannor, Shie, El-Yaniv, Ran, & Crammer, Koby. 2014. Concept Drift Detection Through Resampling. *Pages 1009–1017 of: Proceedings of the 31st International Conference on Machine Learning*.

Howe, Jeff. 2006. The Rise of Crowdsourcing. *Wired*, June.

Huang, Jeff, Thornton, Katherine M., & Efthimiadis, Efthimis N. 2010. Conversational tagging in Twitter. *Pages 173–178 of: Proceedings of the 21st ACM Conference on Hypertext and Hypermedia.*

Hull, D. A. 1994. Improving Text Retrieval for the Routing Problem using Latent Semantic Indexing. *Pages 282–289 of: Proceedings of the International Conference on Research and Development in Information Retrieval.*

Hulten, G., Spencer, L., & Domingos, P. 2001. Mining time-changing data streams. *Pages 97–106 of: Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

Joachims, Thorsten. 1997. A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. *Pages 143–151 of: Proceedings of the 14th International Conference on Machine Learning.*

Joachims, Thorsten. 1998. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. *Pages 137–142 of: Proceedings of the European Conference on Machine Learning.*

Joachims, Thorsten. 2002. *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms.* Kluwer Academic Publisher.

John, George H., Kohavi, Ron, & Pfleger, Karl. 1994. Irrelevant Features and the Subset Selection Problem. *Pages 121–129 of: Proceedings of the International Conference on Machine Learning (ICML).*

Johnson, S. 2009. How Twitter will change the way we live. *Time Magazine*, **173**, 23–32.

Jr., Moacir P. Ponti. 2011. Combining Classifiers: from the creation of ensembles to the decision fusion. *Pages 1–10 of: Proceedings of the 24th Conference on Graphics, Patterns and Images.*

Karataev, Evgeny, & Zadorozhny, Vladimir. 2016. Adaptive social learning based on crowdsourcing. *IEEE Transactions on Learning Technologies.*

Karimi, Fariba, Wagner, Claudia, Lemmerich, Florian, Jadidi, Mohsen, & Strohmaier, Markus. 2016. Inferring gender from names on the web: A comparative evaluation of gender detection methods. *Pages 53–54 of: Proceedings of the 25th International Conference Companion on World Wide Web.*

Karnick, Matthew, Muhlbaier, Michael D., & Polikar, Robi. 2008. Learning concept drift in nonstationary environments using an ensemble of classifiers based approach. *In: Proceedings of the International Joint Conference on Neural Networks (IJCNN).*

Katakis, Ioannis, Tsoumakas, Grigorios, & Vlahavas, Ioannis. 2010. Tracking recurring contexts using ensemble classifiers: an application to email filtering. *Knowledge and Information Systems*, **22**(3), 371–391.

Kim, J., Bentley, P.J., Aickelin, U., Greensmith, J., Tedesco, G., & Twycross, J. 2007a. Immune system approaches to intrusion detection - a review. *Natural Computing*, **6**(4), 413–466.

Kim, S., Oh, J. S., & Oh, S. 2007b. Best-Answer Selection Criteria in a Social Q&A site from the User-Oriented Relevance Perspective. *In: Proceedings of the 70th Annual Meeting of the American Society for Information Science and Technology (ASIST '07).*

Klinkenberg, Ralf. 2004. Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis*, **8**(3), 281–300.

Kolter, Jeremy Z., & Maloof, Marcus A. 2003. Dynamic weighted majority: a new ensemble method for tracking concept drift. *Pages 123–130 of: Proceedings of the 3rd IEEE International Conference on Data Mining.*

Koychev, Ivan. 2000. Gradual forgetting for adaptation to concept drift. *In: Proceedings of ECAI 2000 Workshop on Current Issues in Spatio-Temporal Reasoning.*

Krovetz, Robert. 1993. Viewing morphology as an inference process. *Pages 191–202 of: Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

Kruger, Uwe, Zhang, Junping, & Xie, Lei. 2008. *Developments and Applications of Nonlinear Principal Component Analysis – a Review.* Springer Berlin Heidelberg. Pages 1–43.

Kuncheva, Ludmila. 2002. A Theoretical Study on Six Classifier Fusion Strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **24**(2), 281–286.

Kwak, Haewoon, Lee, Changhyun, Park, Hosung, & Moon, Sue. 2010. What is Twitter, a social network or a news media? *Pages 591–600 of: Proceedings of the 19th International Conference on World Wide Web.*

Langley, Pat. 1994. Selection of Relevant Features in Machine Learning. *Pages 140–144 of: Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) Fall Symposium on Relevance.*

Larkey, L., & Croft, W. 1996. Combining Classifiers in Text Categorization. *Pages 289–297 of: Proceedings of the International Conference on Research and Development in Information Retrieval.*

LaToza, Thomas D, & Van Der Hoek, André. 2015. A vision of crowd development. *Pages 563–566 of: Proceedings of the 37th IEEE International Conference on Software Engineering (ICSE).*

Lee, Chei Sian, Anand, Vishwaraj, Han, Feng, Kong, Xiaoyu, & Goh, Dion Hoe-Lian. 2016. Investigating the Use of a Mobile Crowdsourcing Application for Public Engagement in a Smart City. *Pages 98–103 of: Digital Libraries: Knowledge, Information, and Data in an Open Access Society.* Springer.

Leimeister, JM. 2010. Collective Intelligence. *Business & Information Systems Engineering*, 1–4.

Lewis, David D., Schapire, Robert E., Callan, James P., & Papka, Ron. 1996. Training Algorithms for Linear Text Classifiers. *Pages 298–306 of: Proceedings of the International Conference on Research and Development in Information Retrieval.*

Linting, Marielle, Meulman, Jacqueline J, Groenen, Patrick JF, & van der Kooij, Anita J. 2007. Nonlinear principal components analysis: introduction and application. *Psychological methods*, **12**(3), 336.

Liu, J., Li, X., & Zhong, W. 2009. Ambiguous decision trees for mining concept-drifting data streams. *Pattern Recognition Letters*, **30**(15), 1347–1355.

Liu, Yandong, & Agichtein, Eugene. 2008. On the Evolution of the Yahoo! Answers QA Community. *Pages 737–738 of: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Liu, Yi-Hau, Lin, Shian-Hua, Lai, Chun-Ku, Huang, Chun-Che, & Lu, Cheng-Yu. 2016. Mining Crowdsourcing Photos for Recognizing Landmark Areas. *Pages 12–19 of: Proceedings of the 10th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*.

Lu, Z., Wen, Y., Zhang, W., Zheng, Q., & Cao, G. 2016. Towards Information Diffusion in Mobile Social Networks. *IEEE Transactions on Mobile Computing*, **15**(5), 1292–1304.

Lughofer, Edwin, & Mouchaweh, Moamar Sayed. 2015. Adaptive and on-line learning in non-stationary environments. *Evolving Systems*, **6**(2), 75–77.

Mao, Ke, Capra, Licia, Harman, Mark, & Jia, Yue. 2017. A survey of the use of crowdsourcing in software engineering. *Journal of Systems and Software*, **126**, 57–84.

McCallum, Andrew K., & Nigam, Kamal. 1998. Employing EM and Pool-Based Active Learning for Text Classification. *Pages 350–358 of: Proceedings of the 5th International Conference on Machine Learning (ICML)*.

McNemar, Quinn. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, **12**(2), 153–157.

McPherson, M, Smith-Lovin, L, & Cook, J M. 2001. Birds of a feather: homophily in social networks. *Annual review of sociology*, 415–444.

Mejri, Dhouha, Khanchel, Riadh, & Limam, Mohamed. 2013. An ensemble method for concept drift in nonstationary environment. *Journal of Statistical Computation and Simulation*, **83**(6), 1115–1128.

Merriam-Webster. 2004. *Merriam-Webster's collegiate dictionary.* Merriam-Webster.

Mihalcea, R, & Strapparava, C. 2005. Making computers laugh: investigations in automatic humor recognition. *Pages 531–538 of: Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing.*

Mihalcea, R, & Strapparava, C. 2006. Technologies that make you smile: adding humor to text-based applications. *Intelligent Systems, IEEE*, **21**(5), 33–39.

Mitchell, Tom M. 1997. *Machine learning.* McGraw-Hill.

Mladenić, Dunja. 1998. Feature subset selection in text-learning. *Pages 95–100 of:* Nédellec, Claire, & Rouveirol, Céline (eds), *Proceedings of the 10th European Conference on Machine Learning (ECML).*

Moens, Marie-Francine. 2006. *Information extraction: algorithms and prospects in a retrieval context (The Information Retrieval xseries).* Springer-Verlag New York, Inc.

Mollá, Diego, Santiago-Martínez, María Elena, Sarker, Abeed, & Paris, Cécile. 2016. A corpus for research in text processing for evidence based medicine. *Language Resources and Evaluation*, **50**(4), 705–727.

Moral, Cristian, de Antonio, Angélica, Imbert, Ricardo, & Ramírez, Jaime. 2014. A survey of stemming algorithms in information retrieval. *Information Research: An International Electronic Journal*, **19**(1), n1.

Muhlbaier, Michael D., & Polikar, Robi. 2007. An ensemble approach for incremental learning in nonstationary environments. *Pages 490–500 of: Proceedings of the 7th International Conference on Multiple Classifier Systems.*

Nekovee, Maziar, Moreno, Yamir, Bianconi, Ginestra, & Marsili, Matteo. 2007. Theory of rumour spreading in complex social networks. *Physica A: Statistical Mechanics and its Applications*, **374**(1), 457–470.

REFERENCES

Nguyen, Duc T., & Jung, Jai E. 2017. Real-time event detection for online behavioral analysis of big social data. *Future Generation Computer Systems*, **66**, 137 – 145.

Nishida, Kyosuke, & Yamauchi, Koichiro. 2007. Detecting concept drift using statistical testing. *Pages 264–269 of: Proceedings of the International Conference on Discovery Science*.

O Ozdikis, P Senkul, H Oguztuzun. 2012. Semantic expansion of hashtags for enhanced event detection in Twitter. *In: Proceedings of the 1st International Workshop on Online Social Systems*.

O'Connor, Brendan, Balasubramanyan, Ramnath, Routledge, Bryan R., & Smith, Noah A. 2010. From tweets to polls: linking text sentiment to public opinion time series. *In: Proceedings of the International Conference on Weblogs and Social Media*.

Omer, G., Mutanga, O., Abdel-Rahman, E. M., & Adam, E. 2015. Performance of Support Vector Machines and Artificial Neural Network for Mapping Endangered Tree Species Using WorldView-2 Data in Dukuduku Forest, South Africa. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **8**(10), 4825–4840.

Paola Merlo, James Henderson, Gerold Schneider, & Wehrli, Eric. 2013. Learning Document Similarity Using Natural Language Processing. *Linguistik Online*, **17**(5).

Patist, Jan Peter. 2007. Optimal window change detection. *Pages 557–562 of: Proceedings of the 7th IEEE International Conference on Data Mining Workshop*.

Peer, Eyal, Brandimarte, Laura, Samat, Sonam, & Acquisti, Alessandro. 2017. Beyond the Turk: Alternative platforms for crowdsourcing behavioral research. *Journal of Experimental Social Psychology*, **70**, 153 – 163.

Petrie, C. 2010. Plenty of room outside the firm . *IEEE Internet Computing*, **14**(1), 92–96.

Piskorski, Jakub, & Yangarber, Roman. 2013. *Information Extraction: Past, Present and Future*. Springer Berlin Heidelberg. Pages 23–49.

Polikar, R. 2006. Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, **6**(3), 21–45.

Polikar, R., Upda, L., Upda, S. S., & Honavar, V. 2001. Learn++: an incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 497–508.

Porter, M. F. 1980. An algorithm for suffix stripping. *Program*, **14**(3), 130–137.

Pozzi, Federico Alberto, Fersini, Elisabetta, Messina, Enza, & Liu, Bing. 2016. *Sentiment Analysis in Social Networks*. Morgan Kaufmann.

Raykar, VC, Yu, S, Zhao, LH, Valadez, GH, Florin, C, Bogoni, L, & Moy, L. 2010. Learning from crowds. *The Journal of Machine Learning Research*, **99**, 1297–1322.

Reichenbacher, Tumasch, De Sabbata, Stefano, Purves, Ross S, & Fabrikant, Sara I. 2016. Assessing geographic relevance for mobile search: A computational model and its validation via crowdsourcing. *Journal of the Association for Information Science and Technology*, **67**(11), 2620–2634.

Ren, Ye, Zhang, Le, & Suganthan, P. N. 2016. Ensemble Classification and Regression - Recent Developments, Applications and Future Directions. *IEEE Computational Intelligence Magazine*, **1**(1), 41–43.

Reyes, A, Potthast, M, Rosso, P, & Stein, B. 2010. Evaluating Humor Features on Web Comments. *In: Proceedings of the 7th Conference on International Language Resources and Evaluation*.

Rocchio, J. 1971. *Relevance Feedback in Information Retrieval, The SMART Retrieval System: Experiments in Automatic Document Processing*. G. Salton edn. Prentice Hall. Pages 313–323.

Rodrigues, Filipe, Pereira, Francisco, & Ribeiro, Bernardete. 2013. Learning from multiple annotators: Distinguishing good from random labelers. *Pattern Recognition Letters*, **34**(12), 1428 – 1436.

Ross, Gordon J, Tasoulis, Dimitris K, & Adams, Niall M. 2011. Nonparametric monitoring of data streams for changes in location and scale. *Technometrics*, **53**(4), 379–389.

Salton, Gerard, & Buckley, Christopher. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management*, **24**(5), 513–523.

Sánchez-Maroño, Noelia, Alonso-Betanzos, Amparo, & Tombilla-Sanromán, María. 2007. Filter Methods for Feature Selection – A Comparative Study. *Pages 178–187 of: Proceedings of the 8th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL)*.

Schapire, R., & Singer, Y. 2000. Boostexter: A Boosting-based System for Text Categorization. *Machine Learning*, **39**(2/3), 135–168.

Schapire, R., Singer, Y., & Singhal, A. 1998. Boosting and Rocchio Applied to Text Filtering. *Pages 215–223 of: Proceedings of the International Conference on Research and Development in Information Retrieval*.

Schenk, E., & Guittard, C. 2011. Towards a characterization of crowdsourcing practices. *Journal of Innovation Economics*.

Schlimmer, Jeffrey C., & Granger, Jr., Richard H. 1986. Incremental learning from noisy data. *Mach. Learn.*, **1**(3), 317–354.

Schohn, Greg, & Cohn, David. 2000. Less is More: Active Learning with Support Vector Machines. *Pages 839–846 of: Proceedings of the 7th International Conference on Machine Learning*.

Schütze, Hinrich, Hull, David A., & Pedersen, Jan O. 1995. A Comparison of Classifiers and Document Representations for the Routing Problem. *Pages 229–237 of: Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Sebastiani, Fabrizio. 2002. Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, **34**(1), 1–47.

Settles, Burr. 2010. *Active Learning Literature Survey.* CS Technical Report 1648, University of Wisconsin-Madison.

Shneiderman, Ben, Preece, Jennifer, & Pirolli, Peter. 2011. Realizing the value of social media requires innovative computing research. *Communications of the ACM*, **54**(9), 34–37.

Silva, C., & Ribeiro, B. 2007. On Text-based Mining with Active Learning and Background Knowledge using SVM. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, **11**(6), 519–530.

Silva, C., & Ribeiro, B. 2010. *Inductive Inference for Large Scale Text Classification.* Springer.

Simonson, Katherine M., Jr., Steven M. Drescher, & Tanner, Franklin R. 2007. A Statistics-Based Approach to Binary Image Registration with Uncertainty Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **29**(1), 112–125.

Stanoevska-Slabeva, Katarina, & Schmid, Beat F. 2001. A typology of online communities and community supporting platforms. *In: Proceedings of the 34th Annual Hawaii International Conference on System Sciences.*

Stavropoulos, Thanos G, Andreadis, Stelios, Riga, Marina, Kontopoulos, Efstratios, Mitzias, Panagiotis, & Kompatsiaris, Ioannis. 2016. A Framework for Measuring Semantic Drift in Ontologies. *In: Proceedings of the 12th International Conference on Semantic Systems (SEMANTiCS 2016).*

Stock, O, & Strapparava, C. 2003. Getting serious about the development of computational humor. *Pages 59–64 of: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI).*

Street, W Nick, & Kim, YongSeog. 2001. A streaming ensemble algorithm (SEA) for large-scale classification. *Pages 377–382 of: Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

Sun, Yong, Tan, Wenan, & Zhang, Quanquan. 2016. An efficient algorithm for crowdsourcing workflow tasks to social networks. *Pages 532–538 of: Proceedings of the 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD).*

Surowiecki, James. 2004. *The Wisdom of Crowds.* Doubleday.

Tabassum, Nazia, & Ahmed, Tanvir. 2016. A theoretical study on classifier ensemble methods and its applications. *Pages 67–78 of: Proceedings of the 3rd International Conference on Computing for Sustainable Global Development.*

Tantipathananandh, Chayant, & Berger-Wolf, Tanya Y. 2011. Finding communities in dynamic social networks. *Pages 1236–1241 of: Proceedings of the 11th International Conference on Data Mining.*

Tarasov, Alexey, Delany, Sarah Jane, & Namee, Brian Mac. 2014. Dynamic estimation of worker reliability in crowdsourcing for regression tasks: Making it work. *Expert Systems with Applications*, **41**(14), 6190 – 6210.

Tong, S., & Koller, D. 2002. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, **2**, 45–66.

Treiber, Martin, Schall, Daniel, Dustdar, Schahram, & Scherling, Christian. 2011. Tweetflows: flexible workflows with Twitter. *Pages 1–7 of: Proceedings of the 3rd International Workshop on Principles of Engineering Service-Oriented Systems.*

Tsur, Oren, & Rappoport, Ari. 2012. What's in a hashtag?: content based prediction of the spread of ideas in microblogging communities. *Pages 643–652 of: Proceedings of the 5th International Conference on Web Search and Data Mining.*

Tsymbal, Alexey. 2004. *The problem of concept drift: definitions and related work.* Tech. rept. Department of Computer Science, Trinity College Dublin.

Tsymbal, Alexey, Pechenizkiy, Mykola, Cunningham, Padraig, & Puuronen, Seppo. 2008. Dynamic integration of classifiers for handling concept drift. *Information Fusion*, **9**(1), 56–68.

Uğuz, Harun. 2011. A two-stage feature selection method for text categorization by using information gain, principal component analysis and genetic algorithm. *Knowledge-Based Systems*, **24**(7), 1024–1032.

van Essen, R. M., Milea, V., & Frasincar, F. 2014. A framework for Web news items analysis in relation to share prices. *Pages 197–202 of: Proceedings of the IEEE Conference on Computational Intelligence for Financial Engineering Economics.*

van Rijsbergen, C. 1979. *Information Retrieval.* Butterworths Ed.

Vapnik, Vladimir. 2000. *The Nature of Statistical Learning Theory.* Springer-Verlag New York.

Vitter, Jeffrey S. 1985. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, **11**(1), 37–57.

Wald, A. 1992. *Sequential Tests of Statistical Hypotheses.* Springer New York. Pages 256–298.

Wang, Xiangyang, Yang, Jie, Teng, Xiaolong, Xia, Weijun, & Jensen, Richard. 2007. Feature selection based on rough sets and particle swarm optimization. *Pattern recognition letters*, **28**(4), 459–471.

Weerkamp, W., Carter, S., & Tsagkias, E. 2011. *How People use Twitter in Different Languages.*

Welinder, P., & Perona, P. 2010. Online crowdsourcing: rating annotators and obtaining cost-effective labels. *Pages 25–32 of: Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition workshops.*

Widmer, Gerhard, & Kubat, Miroslav. 1993. Effective learning in dynamic environments by explicit context tracking. *Pages 227–243 of: Proceedings of the European Conference on Machine Learning.*

Widmer, Gerhard, & Kubat, Miroslav. 1996. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, **23**(1), 69–101.

Willett, Peter. 2006. The Porter stemming algorithm: then and now. *Program*, **40**(3), 219–223.

Wong, Christina, Amini, Reza, & De Koninck, Joseph. 2016. Automatic gender detection of dream reports: A promising approach. *Consciousness and cognition*, **44**, 20–28.

Xu, Zheng, Zhang, Hui, Hu, Chuanping, Mei, Lin, Xuan, Junyu, Choo, Kim-Kwang Raymond, Sugumaran, Vijayan, & Zhu, Yiwei. 2016. Building knowledge base of urban emergency events based on crowdsourcing of social media. *Concurrency and Computation: Practice and Experience.*

Yang, Lei, Sun, Tao, Zhang, Ming, & Mei, Qiaozhu. 2012. We know what @you #tag: does the dual role affect hashtag adoption? *Pages 261–270 of: Proceedings of the 21st International Conference on World Wide Web.*

Yates, Frank. 1934. Contingency tables involving small numbers and the χ 2 test. *Supplement to the Journal of the Royal Statistical Society*, **1**(2), 217–235.

Ye, Y., Squartini, S., & Piazza, F. 2013. Online sequential extreme learning machine in nonstationary environments. *Neurocomputing*, **116**.

Yuen, M. C., King, I., & Leung, K. S. 2011. A Survey of Crowdsourcing Systems. *Pages 766–773 of: Proceedings of the 3rd International Conference on Privacy, Security, Risk and Trust and 3rd International Conference on Social Computing.*

Zangerle, Eva, Gassler, Wolfgang, & Specht, Günther. 2011. Recommending #-tags in Twitter. *Pages 67–78 of: Proceedings of the 19th International Conference on User Modeling, Adaptation and Personalization.*

Zappavigna, Michele. 2011. Ambient affiliation: A linguistic perspective on Twitter. *New Media & Society*, **13**(5), 788–806.

Zareapoor, Masoumeh, & Seeja, KR. 2015. Feature Extraction or Feature Selection for Text Classification: A Case Study on Phishing Email Detection. *International Journal of Information Engineering and Electronic Business*, **7**(2), 60.

Žliobaitė, Indrė. 2010. *Learning under concept drift: an overview.* Tech. rept. Faculty of Mathematics and Informatic, Vilnius University, Latvia.

Žliobaitė, Indrė, Bifet, Albert, Pfahringer, Bernhard, & Holmes, Geoffrey. 2014. Active learning with drifting streaming data. *IEEE Transactions on Neural Networks and Learning Systems*, **25**(1), 27–39.