Alma Mater Studiorum · Università di Bologna

Campus di Cesena
Scuola di Ingegneria e Architettura
CORSO DI LAUREA MAGISTRALE IN INGEGNERIA E SCIENZE INFORMATICHE

NEW MARKOV CHAIN BASED METHODS FOR SINGLE AND CROSS-DOMAIN SENTIMENT CLASSIFICATION

Tesi in: Data Mining

Relatore:
Prof. GIANLUCA MORO
Correlatori:
DOTT. ING. GIACOMO DOMENICONI
DOTT. ING. ROBERTO PASOLINI

Presentata da:
ANDREA PAGLIARANI

ANNO ACCADEMICO 2013-2014
III SESSIONE

ABSTRACT

Nowadays communication is switching from a centralized scenario, where communication media like newspapers, radio, TV programs produce information and people are just consumers, to a completely different decentralized scenario, where everyone is potentially an information producer through the use of social networks, blogs, forums that allow a real-time worldwide information exchange. These new instruments, as a result of their widespread diffusion, have started playing an important socio-economic role. They are the most used communication media and, as a consequence, they constitute the main source of information enterprises, political parties and other organizations can rely on. Analyzing data stored in servers all over the world is feasible by means of Text Mining techniques like Sentiment Analysis, which aims to extract opinions from huge amount of unstructured texts. This could lead to determine, for instance, the user satisfaction degree about products, services, politicians and so on. In this context, this dissertation presents new Document Sentiment Classification methods based on the mathematical theory of Markov Chains. All these approaches bank on a Markov Chain based model, which is language independent and whose killing features are simplicity and generality, which make it interesting with respect to previous sophisticated techniques. Every discussed technique has been tested in both Single-Domain and Cross-Domain Sentiment Classification areas, comparing performance with those of other two previous works. The performed analysis shows that some of the examined algorithms produce results comparable with the best methods in literature, with reference to both single-domain and cross-domain tasks, in 2-classes (i.e. positive and negative) Document Sentiment Classification. However, there is still room for improvement, because this work also shows the way to walk in order to enhance performance, that is, a good novel feature selection process would be enough to outperform the state of the art. Furthermore, since some of the proposed approaches show promising results in 2-classes Single-Domain Sentiment Classification, another future work will regard validating these results also in tasks with more than 2 classes.

ABSTRACT

Oggigiorno la comunicazione si sta evolvendo da uno scenario centralizzato, in cui mezzi di comunicazione come giornali, radio, programmi televisivi producono informazione e le persone fungono solamente da consumatori, ad uno scenario completamente differente, decentralizzato, in cui chiunque è potenzialmente un produttore di informazioni grazie all'uso di social network, blog, forum che permettono uno scambio di informazioni capillare in tempo reale. Questi nuovi strumenti, grazie alla loro diffusione mondiale, stanno iniziando sempre più a giocare un importante ruolo socio-economico. Essendo i mezzi di comunicazione oggi più usati, costituiscono di conseguenza la fonte di informazioni primaria per aziende, partiti politici ed altre organizzazioni. L'analisi dei dati memorizzati nei server sparsi per il mondo è possibile grazie a tecniche di Text Mining come la Sentiment Analisys, il cui obiettivo è l'estrazione di opinioni da enormi quantità di testo non strutturato. Tali analisi possono permettere di determinare ad esempio il grado di soddisfazione dell'utente riguardo a prodotti, servizi, personaggi politici, ecc. All'interno di questo contesto, questo lavoro presenta nuovi metodi di Document Sentiment Classification basati sulla teoria matematica delle Markov Chain. Tutti gli approcci proposti hanno come loro modello base una Markov Chain e non fanno assunzioni sulla lingua in cui i testi da analizzare sono scritti. Le carte vincenti del modello proposto, che lo rendono interessante rispetto alle precedenti tecniche spesso molto sofisticate, sono la sua semplicità e la sua generalità. Tutte le tecniche discusse sono state testate sia in Single-Domain che in Cross-Domain Sentiment Classification, confrontando le performance con quelle di altri due lavori precedenti. Le analisi condotte mostrano che alcuni degli algoritmi esaminati producono risultati in linea con i metodi della letteratura con riferimento alla Document Sentiment Classification a 2 classi (i.e. positiva e negativa), sia in problemi single-domain che in cross-domain. Tuttavia il margine di miglioramento è ancora ampio, in quanto questo lavoro indica la strada da percorrere al fine di incrementare le performance, ovvero la necessità di un nuovo metodo di feature selection, che dovrebbe essere sufficiente a superare lo stato dell'arte. Inoltre, dal momento che alcuni degli approcci presentati si sono dimostrati promettenti nella Single-Domain Sentiment Classification a 2 classi, un ulteriore sviluppo futuro riguarderà la validazione di questi risultati estendendoli al caso di Single-Domain Sentiment Classification con più classi.

I dedicate my dissertation to my girlfriend Francesca, my family and my friends. Francesca has always given her support, saying me to believe in what I was doing and encouraging all my choices.
I also dedicate my work to my parents and my grandparents. They always made sure I had everything I needed. I will never forget what they have done, giving me the opportunity to carry on with my studies.
A special thought is directed to my uncle Orio, who has gone prematurely. He always said to me that there is not a career better than another, what really matters is doing anything you like, because this is the only way to make dreams come true. I will remember his words forever and I will follow his precious suggestion.
In addition, I give a special thanks to my cousin Elisa, who gave me many valuable advices regarding both my dissertation and my future.
Finally, I dedicate this dissertation to my friends, especially Berna and Mangia, who have constantly asked me about working progress.

ACKNOWLEDGEMENTS

This dissertation would not have been possible unless the support of my supervisor, Professor Gianluca Moro, who gave me the opportunity of going abroad to carry it out. He trusted and supported me not only in Italy, but also during my stay in Spain, giving many precious suggestions and conveying interest to research.
I am also grateful to Professor Rubén Cuevas Rumín and to the whole Telematic Engineering Department at Charles III University of Madrid. They kindly hosted me from October to February, also providing two important datasets, whose analysis helped improving this work.
In addition, I would like to thank my co-supervisors, Dr. Eng. Giacomo Domeniconi and Dr. Eng. Roberto Pasolini, who constantly gave me food for thought. Without their valuable aid in several circumstances, results would not be as good as they are now.
Finally, I want to express my gratitude to my girlfriend and my family, who always morally support me and help me believing in myself.

DECLARATION

I declare that I have developed and written the enclosed Master Thesis completely by myself, and have not used sources or means without declaration in the text. Any thoughts from others or literal quotations are clearly marked. The Master Thesis was not used in the same or in a similar version to achieve an academic grading or is being published elsewhere.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

RMM Regular Markov Model

RMC Regular Markov Chain

HMM Hidden Markov Model

HMC Hidden Markov Chain

MC Markov Chain

DTMC Discrete-Time Markov Chain

TM Text Mining

TC Text Categorization

SA Sentiment Analysis

OM Opinion Mining

IR Information Retrieval

CD Cross-Domain

DSC Document Sentiment Classification

OW Opinion Words

FS Feature Selection

TW Term Weighting

MCSD Markov Chain Stationary Distribution

API Application Programming Interface

IDE Integrated Development Environment

INTRODUCTION

Communication is a purposeful activity of exchanging information and meaning across space and time using various technical or natural means, whichever is available or preferred. Communication requires a sender, a message, a medium and a recipient, although the receiver does not have to be present or aware of the sender's intent to communicate at the time of communication; thus communication can occur across vast distances in time and space. Communication requires that the communicating parties share an area of communicative commonality. The communication process is complete once the receiver understands the sender's message. [1]

Centralized and professional media have been the main senders in the communication process for a long time. News were used to be published on newspapers, magazines, or to be broadcasted through newscasts and other TV or radio programs. Although these media are still used nowadays, communication has been evolving towards different information media, such as social networks, blogs, forums, which allow an interactive and worldwide information exchange. This quick evolution is having a strong social impact, because people can also produce information rather than just being consumers. Consequently, information exchange and the problems of forming and expressing opinions become simpler, whereas at the same time news are spread more rapidly than in the classical media. Recent example is the crisis in Middle East, facilitated by social networks, which have enabled both people organization and people awareness about social and economical conditions in other countries.

Currently, along with their social value, social networks have started playing an important role from the economical point of view, since users exchange and share content about any topic, like for example people, products, services, events, TV programs, and so on. Therefore, information extraction from data could aid decision support for a large variety of organizations, which go from public and private enterprises to the financial sector, the political parties, etc. Moreover, relying on recent studies conducted by international companies as Gartner Group [2] and Butler Group [3] [52], the business value of textual information has overcome that of structured data (i.e. data commonly stored in databases). In fact, they estimate that about 80% of company's information is unstructured (i.e. textual) data.

Analyzing the huge amount of data that is stored daily in servers,

1 http://en.wikipedia.org/wiki/Communication

2 http://www.gartner.com/

3 http://ovumlive.com/

it is possible to find out useful information. For example, we could determine the users satisfaction degree about products or services, we could study market demand in order to foresee which new products or services might be successful, and so on. These analyses could be performed thanks to Text Mining [53, 54], a discipline that studies methods, techniques and algorithms aiming to extract knowledge from huge quantity of texts and to produce models capable of describing and/or predicting social, economical, scientific phenomena. Text Mining approaches have been widely used for security, biomedical, marketing purposes, and so on, but the focus of this work is on Sentiment Analysis (SA), also known as Opinion Mining (OM), a discipline framed within the area of Natural Language Processing (NLP) that can be viewed as the computational treatment of opinions, feelings and subjectivity in texts [1]. Over the years, lots of techniques have been developed in this area and in particular in the context of Document Sentiment Classification, which aims to find out document polarity (i.e. positive, negative, neutral opinions). However, many proposed models have a strong limitation, that is, they are able to classify polarity just in documents discussing the same topic they were built over (i.e. Single Sentiment Classification), while they show poor performance if applied to unrelated domains. To overcome this weakness, researchers have started working on Cross-Domain Sentiment Classification, whose goal is building general models over a particular domain, called source domain, and applying them to a different domain, namely target domain. It is not hard to understand that cross-domain problems are demanding, just think that analyses are carried out focusing on words and that people make use of diverse terms while describing things related to dissimilar topics. Literature is plenty of promising methods; however, due to the aforementioned complexity of the problem, the majority of them are complicated and require strong effort in parameter calibration.

This dissertation proposes new general and relatively simple methods for Single-Domain and Cross-Domain Sentiment Classification, based on the mathematical theory of Markov Chains. Markov models have shown being promising in various areas, such as Statistics, Speech Recognition, Chemistry, Physics, Finance, Games, Music, and they have been included in several Text Mining techniques that have proved their usefulness. Therefore, the idea behind this work is exploiting the soundness of a mathematical theory to build and validate a model for automated Document Sentiment Classification. Markov Chains have been basically employed in order to model co-occurrences of words in documents relying on a bag-of-words document representation. The performed tests have revealed results comparable with the state of the art in both Single and Cross-Domain Sentiment Classification.

The remainder of the dissertation is organized as follows:

- *Chapter 1* introduces Markov models, describing their basic characteristics and their main properties, with reference to both Regular Markov Models and Hidden Markov Models. Then, the chapter throws light on some algorithms, namely Forward-Backward algorithm, Viterbi algorithm and Baum-Welch algorithm, widely used in the Hidden Markov Models context.

- *Chapter 2* exposes and analyzes some related works which made use of Markov models in Text Mining branches as Information Retrieval (IR), Text Categorization (TC), Sentiment Analysis (SA). Furthermore, some papers about Cross-Domain Text Categorization and Document Sentiment Classification are described as well.

- *Chapter 3* presents the novel Markov Chain based methods for Single and Cross-Domain Sentiment Classification, emphasizing their straightforwardness and generality. In particular, a basic algorithm relying on a Markov Chain model is introduced at first. Then, other variants of this algorithm are discussed and analyzed.

- *Chapter 4* acquaints with the framework built for the sake of implementing the proposed methods and performing analysis. Further, some examples are shown about how the techniques illustrated in the previous chapter can be realized in the proposed framework.

- *Chapter 5* first of all describes the datasets used to perform experiments. Then, it illustrates tests carried out in both Single-Domain and Cross-Domain Sentiment Classification, and shows results, comparing them with other two methods, namely Spectral Feature Alignment (SFA), advanced by Pan et al. [55], and Joint sentiment-topic (JST) with polarity-bearing topics (PBT), introduced by He et al. [56].

- The dissertation ends with a conclusion, where the work is summed up and some possible future improvements are proposed.

1

MARKOV MODELS: AN INTRODUCTION

This chapter will give readers some basic notions about Markov models. Firstly, I will introduce Markov models and their fundamental characteristics. Then, I will talk about two different Markov models, namely Regular Markov Models (RMMs) and Hidden Markov Models (HMMs), widely used in many different areas. Finally, I will analyze in detail some algorithms, such as Forward-Backward algorithm, Viterbi algorithm and Baum-Welch algorithm, which have been proved useful in solving some general problems related to HMMs.

1.1 WHAT IS A MARKOV MODEL?

Markov models derive their name from Andrej Markov (1856-1922), a Russian mathematician who focused his researches on probabilistic models, namely stochastic processes dealing with uncertainty.
A Markov process, better known as Markov Chain (MC), is a mathematical system that, starting from an initial state, is subjected to transition from one state to another on a space state. The term "chain" is due to the sequence of steps (i.e. transitions) the process moves through. Basically, a MC is a discrete-time stochastic process having the special characteristic to be memoryless. A stochastic process is a collection of random variables (or aleatory variables) dealing with some probability distribution. Stochastic processes are particularly useful in modeling non-deterministic scenarios, where we cannot know in advance the exact evolution of the system, because there is more than one available future state starting from the current state. Stochastic processes are particularly suitable for modeling Complex Systems, which usually are dynamical systems composed of many elements having non-linear interactions and whose topology can be represented as a network. These systems behaviour is intrinsically governed by non-determinism and for this reason stochastic processes are appropriate models. Apart from being a stochastic process, the characteristic distinguishing a Markov process is, as I said before, memorylessness, which is guaranteed by the so called Markov property. Essentially, this property states that the future state in the system evolution depends only on present state, regardless all the past states that have been crossed.
Formally, let T be an aleatory variable with negative exponential dis-

tribution. Its probability density function (pdf) is:

$$f(x;\lambda) = \begin{cases} \lambda e^{-\lambda x}, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

and it satisfies:

$$P(T > s + t | T > s) = P(T > t), \qquad \forall t, s > 0 \tag{1}$$

The equation 1 is known as **Markov property**. It is important to note the assumption specifying that T has a negative exponential distribution, because Markov property does not hold for any probability distribution. As a consequence, not every process is suitable to be modeled as a Markov process. For example, walk on graph without visiting two times the same node is not a Markov process. The same is true also for reacting in some way if two sequential events happen, or for sampling without replacement in order to draw a particular object in a container. All the aforementioned tasks are not fitted to be modeled as Markov processes because they need for history trace, whereas as I pointed out Markov processes are based on the assumption of independence from past states. On the other hand, examples of problems naturally representable by Markov processes are goals scored in a soccer match, phone calls arrival, requests for a certain document on a web server, and many others.

1.2 REGULAR MARKOV MODELS

This section is about Regular Markov Models, according to the definition given in [85] to distinguish them from Hidden Markov Models (see Section 1.3). Since in Section 1.1 we have introduced some basic concepts, now we can give a formal definition of what a Discrete-Time Markov Chain (DTMC) is. A Discrete-Time Markov Chain, or simply a Markov Chain (MC), is a sequence of random variables having the Markov property.

Definition Let $X_1, X_2, ..., X_n$ be random variables satisfying the Markov property. Then the following relation

$$\begin{aligned} P(X_{n+1} = x | X_1 = x_1, X_2 = x_2, ..., X_n = x_n) = \\ = P(X_{n+1} = x | X_n = x_n) \end{aligned} \tag{2}$$

holds if both conditional probabilities are well defined, that is, if

$$\begin{aligned} P(X_1 = x_1, X_2 = x_2, ..., X_n = x_n) > 0 \\ P(X_n = x_n) > 0 \end{aligned}$$

All X_i values form the *state space* of the Markov Chain.

Being composed of a set of states and characterized by transition probabilities of moving from one state to another, a Markov Chain can be equivalently described in two main ways, that is, by means of:

- a sequence of directed graphs
- a sequence of transition matrices

However, Markov Chains are commonly assumed to be time-homogeneous, namely transition probabilities between states do not change over time (a better explanation of what a time-homogeneous Markov Chain is can be found later in this chapter). In this case, for modeling purposes it is enough a single directed graph, or equivalently a single transition matrix. So we are going to explain better how these two representations work.

1.2.1 *Directed Graph Representation*

As we have just declared, one way to represent a time-homogeneous Markov Chain is through a directed graph. People unfamiliar with Graph Theory could find a great introduction in [86]. Anyway, I would like to remind that a graph is an ordered pair $G = (V, A)$, where V is a set of vertices and A is a set of arcs. Each arc is related with two vertices. If the relationship between the two vertices and the edge is represented as an unordered pair, the graph is called undirected (in this case the set A is indicated with the letter E). On the other hand, if the relationship is an ordered pair, namely the edge goes from a vertex $x \in V$ to a vertex $y \in V$, the graph is named directed and instead of edge we can use expressions like arc and directed edge.
The directed graph representation is suitable for Markov Chains, because nodes (i.e. vertices) embody MC states and edges embody MC transition probabilities between states. Of course, for each state, the sum of all outgoing arcs values is equal to 1, since they represent probabilities. More formally, let $V = \{x_1, ..., x_n\}$ be the set of MC states and let $A = \{(x_i, x_j)\}$ be the set of couples (x_i, x_j) representing directed edges. Further, let each directed edge (x_i, x_j) embody $P(x_j|x_i)$, namely the conditional probability that, starting from a state x_i, a state x_j is reached in one step. Then, the following properties hold:

$$P(x_i) \geq 0, \qquad \forall i = 1, ..., n \tag{3}$$

$$P(x_j|x_i) \geq 0, \qquad \forall i, j = 1, ..., n \tag{4}$$

$$\sum_{j=1}^{n} P(x_j|x_i) = 1, \qquad \forall i = 1, ..., n \tag{5}$$

Equation 3 refers to the non-negativity property of probability. Further, being arcs probability values, they must be non-negative as well,

according to equation 4. Finally, equation 5 recalls the unitarity property of probability. Careful readers could have noticed that equation 4 is general and it can be applied as is to fully connected graphs. However, we are not required dealing with fully connected graphs. Therefore, when building the directed graph representation of the Markov Chain, we can just draw arcs having a probability value greater than zero, overlooking the others.
Before talking about the alternative mode to represent a Markov Chain, let us look at a simple example just to clarify what we have just introduced. Let us imagine we would like to model weather. Let us assume for example that weather should be *sunny*, *cloudy* or *rainy*. Then, let us suppose we know that the day after a *sunny* day will be *cloudy* with probability 0.6, *sunny* again with probability 0.3, whereas *rainy* with probability 0.1. Likewise, after a *cloudy* day will be *sunny* day with probability 0.4, a *rainy* day with probability 0.4 and a *cloudy* day again with probability 0.2. Finally, after a *rainy* day weather will be *sunny* with probability 0.65 and *cloudy* with probability 0.35, while it is not possible another *rainy* day. In this case, a Markov Chain could be used to model this system, whose states will be *sunny*, *cloudy* and *rainy* and whose transition probabilities will be the aforementioned ones. Figures 1 and 2 will show two equivalent ways to represent Markov Chain as directed graph. The second one, where the directed graph is not fully connected if unnecessary, is obviously preferable.

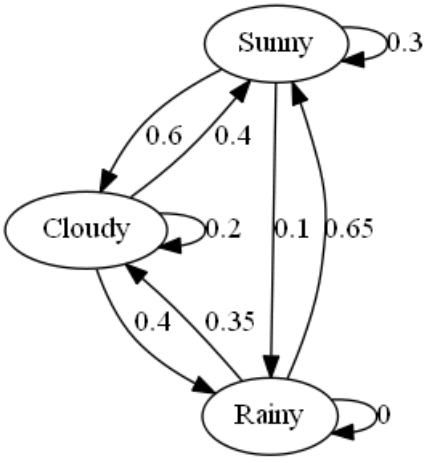

Fig. 1: *The figure shows a directed graph representation of a Markov Chain. Nodes represent the states in the system being modeled, whereas arcs the transition probabilities between them. Notice that this is a fully connected graph, because each state is connected with all others, but there is no need for this since* $P(rainy_{n+1}|rainy_n) = 0$.

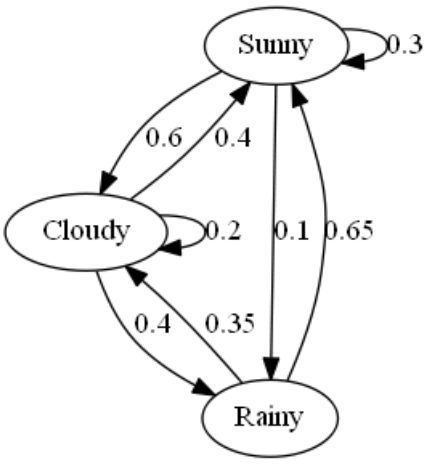

Fig. 2: *The figure represents the same model as in figure 1. However, we can notice that the arc expressing that* $P(rainy_{n+1}|rainy_n) = 0$ *has been removed, since probabilities equal to* 0 *do not add useful information. Thus, this representation is preferable to the one in figure 1.*

1.2.2 *Transition Matrix Representation*

We have just shown that a time-homogeneous Markov Chain (or time-homogeneous DTMC) can be portrayed as a directed graph. Besides this, there is an alternative way to provide its description, namely the Transition Matrix representation. A transition matrix T is simply a squared matrix where rows contain current states (i.e. states at time n) and columns contain future states (i.e. states at time $n+1$). So both on rows and on columns the same set of states is represented. Instead what is really discriminant is the content of each cell, which is the probability that a future state is reached while being in a particular current state. I would like to remind that this alternative illustration is totally equivalent to the one based on directed graph. In fact, each transition from state x_i to state x_j, occurring with probability $P(x_j|x_i)$, is mapped from directed graph to transition matrix as shown in table 1. Therefore, equations 3, 4 and 5 must hold as well. In particular, all matrix rows sum-up to 1, as claimed by 5.

According to what I have just affirmed, we can find in table 2 how the aforementioned example about weather results in transition matrix representation.

1.2.3 *Beyond Discrete-Time Markov Chains*

So far I have presented Discrete-Time Markov Chains (DTMC). Here, I will talk about some possible variations of this basic model, such as

Element	**Directed Graph Representation**	**Transition Matrix Representation**
Initial state x_i	Vertex x_i where the arc starts from	State x_i belonging to rows
Final state x_j	Vertex x_j where the arc ends	State x_j belonging to columns
Transition between states $P(x_j\|x_i)$	Directed edge (x_i, x_j) linking the two vertices	Matrix entry $T[i, j]$

Table 1: The table represents the different but equivalent way of expressing Markov Chain elements in directed graph and transition matrix.

		FS		
		Sunny	**Cloudy**	**Rainy**
CS	**Sunny**	0.3	0.6	0.1
	Cloudy	0.4	0.2	0.4
	Rainy	0.65	0.35	0

Table 2: This table shows the Transition Matrix representation of a Markov Chain. CS stands for current states, whereas FS stands for future states. Notice that all rows sum-up to 1.

Time-Homogeneous Markov Chains (or Stationary Markov Chains) and Markov Chains of order k. Further, I would like to point out that these are not the only known variants. For example, another largely employed model is Continuous-Time Markov Chains (CTMC), which is substantially the continuous-time version of DTMC. I will not discuss this and other models for time reason. Readers who like to know more about these models can find more information in [77], [78], [84] and [83].

1.2.3.1 *Time-Homogeneous Markov Chains*

We are already aware of the fact that DTMC can be represented in the form of either a sequence of directed graphs or a sequence of transition matrices. This lets Markov Chains evolve, in accordance with the fact that transition probabilities should change over time. Nevertheless, a widely used assumption is the time-homogeneity, which deals with situations where the transition probabilities do not change over time. Regarding the weather forecast example, this means that we could model it in a given time range in which transition probabilities remain the same. In fact, this is exactly what we did in the aforementioned example where, as pointed out, the time-homogeneity property held. On the other side, we could not relate weather forecast in two not overlapping time ranges characterized by diverse transition probabilities (e.g. considering two years, the probability weather will be *sunny*, *cloudy* or *rainy* should change from the first to the second year).
Formally, Time-Homogeneous Markov Chains (also known as Stationary Markov Chains) are processes where

$$P(X_{n+1} = x | X_n = y) = P(X_n = x | X_{n-1} = y), \qquad \forall n \in \mathbb{N}^+ \tag{6}$$

that is, transition probabilities do not change over time. As a consequence, an unique directed graph (or equivalently an unique transition matrix) is sufficient to represent the Markov Chain.

1.2.3.2 *Markov Chains of order k*

Another variant of standard DTMC is DTMC of order k, namely a process where

$$\begin{aligned} &P(X_n = x_n | X_{n-1} = x_{n-1}, X_{n-2} = x_{n-2}, ..., X_1 = x_1) = \\ &\quad = (X_n = x_n | X_{n-1} = x_{n-1}, X_{n-2} = x_{n-2}, ..., X_{n-k} = x_{n-k}), \\ &\quad \forall n > k \end{aligned} \tag{7}$$

This is an extension of the basic memoryless version of Markov Chains. In fact, one of the characteristics of standard Markov Chains is the independence from past states. Instead, introducing Markov Chains of order k, the future state depends on the previous k states. According to this, it is possible to partially keep history trace: incrementing k the model will rely on a longer timespan and vice versa.

1.2.4 *Properties*

Now we are going to analyze some Markov Chain properties. To describe each property, let $S = \{s_1, s_2, ..., s_n\}$ be the set of Markov Chain states and consider $s_i = i$ and $s_j = j$ for simplicity of notation. Moreover, henceforth I will use the terms Markov Chains, MC, Regular Markov Chains, RMC, Regular Markov Models, RMM interchangeably to indicate a Time-Homogeneous Discrete-Time Markov Chain.

1.2.4.1 *Reducibility*

In order to describe the reducibility property, we need introducing some preliminary auxiliary definitions.

A state j is defined as *accessible* from a state i, say $i \to j$, if, starting from i, the probability of transitioning into state j at some point is greater than zero. In a formal fashion, a state j is accessible from a state i if there exists $k_{ij} \in \mathbb{Z}$ such that

$$P(X_{k_{ij}} = j | X_0 = i) = p_{ij}^{(k_{ij})} > 0, \qquad i, j \in S \tag{8}$$

where k_{ij} represents one particular iteration in the evolution of the system (i.e. the iteration in which $p_{ij} > 0$).

A state i is said to *communicate* with a state j, say $i \leftrightarrow j$, if i and j are accessible one from each other.

A set of states U is a *communicating class* if every pair of states in U communicates with each other and no state in U communicates with another state not in U.

Consequently, the reducibility property is as follows:

Definition A Markov Chain MC is said to be *irreducible* if its state space is a communicating class (i.e. every pair of MC states communicates with each other). In other words, according to the definition of the communication property, it is irreducible is any state is accessible from any other state.

1.2.4.2 *Periodicity*

A state i is said to have *period* k if it is possible to reach this state only in multiples of k time steps. The formal definition of period relies on the greatest common divisor (gcd) function and it is as follows:

$$k = gcd\{n : P(X_n = i | X_0 = i) > 0\} \tag{9}$$

According to this, it is also possible that a periodic state having period k will not be reached every k time steps. On the other hand, if a periodic state having period k has been reached, the current time step is surely a multiple of k.

Instead, a state i is defined as *aperiodic* if $k = 1$, because in this case state i can be reached with no specified periodicity. More formally:

Definition A state i is said to be aperiodic if there exists n such that

$$P(X_{n'} = i | X_0 = i) > 0, \qquad \forall n^{'} \geq n, \quad i \in S \tag{10}$$

Definition A Markov Chain is called *aperiodic* if and only if each state is aperiodic.

Theorem 1.2.1. *Let MC be an irreducible Markov Chain. Then, if one state is aperiodic this imply the entire Markov Chain is aperiodic.*

1.2.4.3 *Transience*

A state i is said to be *transient* if the probability that, starting from i, we will never return to it, is different from 0. More precisely:

Definition Let the random variable T_i be the so called hitting-time, that is the first time we return to state $i \in S$:

$$T_i = inf\{n > 1 : X_n = i | X_0 = i\} \tag{11}$$

The probability we come back again to state i after n steps is

$$h_{ii}^{(n)} = P(T_i = n) \tag{12}$$

Consequently, state i is *transient* if

$$P(T_i < \infty) = \sum_{n=1}^{\infty} h_{ii}^{(n)} < 1 \tag{13}$$

A state which is not transient it is called *recurrent*.

Lemma 1.2.2. *Recurrent states have always a finite hitting-time.*

Definition The *mean recurrence time* at state $i \in S$ is the expected return time M_i:

$$M_i = E[T_i] = \sum_{n=1}^{\infty} n \cdot h_{ii}^{(n)} \tag{14}$$

If M_i is finite, state i is defined *positive recurrent*; otherwise, it is defined *null recurrent*.

1.2.4.4 *Absorbing States*

Definition A state $i \in S$ is said to be *absorbing* if

$$p_{ij} = \begin{cases} 1 & : i = j \\ 0 & : i \neq j \end{cases} \tag{15}$$

In other words, a state is absorbing if, once entered it, it is impossible to leave.

Definition A Markov Chain is called an *absorbing Markov Chain* if any Markov Chain state can reach an absorbing state.

1.2.4.5 *Ergodicity*

Definition A state $i \in S$ is *ergodic* if it is aperiodic and positive recurrent.

Definition An irreducible Markov Chain is said to be *ergodic* if all Markov Chain states are ergodic.

Definition A model has the ergodic property if there exists $N \in \mathbb{N}^+$ such that any state can be reached from any other state in exactly N steps.

Theorem 1.2.3. *Let a finite state Markov Chain MC be irreducible. If MC has an aperiodic state, then it is ergodic.*

1.2.4.6 *Stationary Distribution and Limiting Distribution*

Definition Let MC be a Time-Homogeneous Markov Chain. Then the vector π is defined as *stationary distribution* (or *steady state*) of the Markov Chain if $\forall j \in S$ the following relations hold:

$$\begin{cases} 0 \leq \pi_j \leq 1 \\ \sum\limits_{j \in S} \pi_j = 1 \\ \pi_j = \sum\limits_{i \in S} \pi_i p_{ij} \end{cases} \tag{16}$$

where p_{ij} is the transition probability from state $i \in S$ to state $j \in S$.

Theorem 1.2.4. *Let MC be an irreducible Markov Chain. Then MC has stationary distribution π if and only if all its states are positive recurrent.*

Lemma 1.2.5. *Let MC be an irreducible Markov Chain having stationary distribution π. Then π is unique and it is related to the expected return time as follows:*

$$\pi_j = \frac{c}{M_j} \tag{17}$$

where c is a constant value.

Definition Let MC be a positive recurrent Markov Chain both irreducible and aperiodic. Then MC has the so called *limiting distribution* regardless its initial distribution, namely

$$\lim_{n \to \infty} p_{ij}^{(n)} = \frac{c}{M_j}, \qquad \forall i, j \in S \tag{18}$$

1.2.5 *Applications*

Regular Markov Models are essentially a set of states linked by transition probabilities. These models are particular useful to accomplish three general tasks:

- Simulation
- Estimation
- Prediction

Simulation consists in generating a random sequence based on the state transition probabilities, starting from a specified state in the Markov Chain. Simulation is a widely used task in many scientific contexts, mainly suitable for showing and/or understanding the behaviour of a system. Regarding the aforementioned example, we could for instance simulate the weather trend starting from a sunny day and stopping the simulation after 20 days.
Another important achievable task is estimation, which is dual of simulation. Here, starting from a sequence of observations we can fit the model estimating transition probabilities between the states that appeared in the observation itself. For example, if we knew what weather was in a given time range, say 20 days, we could estimate the probability that a sunny day is followed by a cloudy day and so on. A widespread estimator is Maximum Likelihood Estimation (MLE), which is

$$p_{ij}^{MLE} = \frac{n_{ij}}{\sum_{u=1}^{k} n_{iu}} \tag{19}$$

where n_{ij} is the number of sequences $(X_t = s_i, X_{t+1} = s_j)$ found in the input sample. Apart from this, other estimators could be used, like for instance MLE with Laplace smoothing and Bootstrap estimator, just to mention some.
A third common task solvable by RMMs is prediction, which involves computing conditional probability distribution of X_{t+1} given $X_t = s_j$, written as $P(X_{t+1} = s_i | X_t = s_j)$, being s_j the last realization of DTMC. In accordance with the same example, prediction by means of Markov Chains allows to forecast the next day's weather given the current meteorological condition.

1.3 HIDDEN MARKOV MODELS

In this section I will give some basic concepts about Hidden Markov Models, a more general and complex model than RMM. More complete explanations can be found in [85], [83] and [80].
A Hidden Markov Model (HMM) is a statistical model in which the system is modeled as a Markov process where

- state y_i is not directly visible (hidden)
- output (or observation) x_i, depending on states, is visible

As we can see in figure 3, since HMM states are linked by arcs representing conditional dependencies, they satisfy the Markov property,

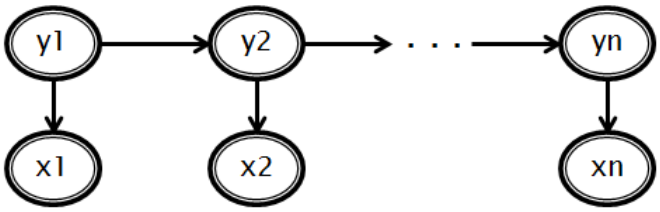

Fig. 3: *The figure shows a HMM. The nodes $y_1, y_2, ..., y_n$ represent hidden states. The nodes $x_1, x_2, ..., x_n$ represent observations. Notice that each state is related to the previous one according to the Markov property. Furthermore, state y_i is also related to the observation x_i.*

namely state y_{i+1} is conditionally independent from state y_{i-1} given state y_i. Therefore, the set of states form a Markov Chain as in Regular Markov Models. Similarly, also the output x_i depends only on the state y_i. The main difference between a HMM and a RMM is that in the latter states are completely visible. We exactly know all the states being modeled in the system we are describing and, consequently, the only parameters to be estimated are state transition probabilities. Instead, in the former states are hidden and, as we will see, there are many parameters to be calibrated.
Now that I have given readers an idea of what a Hidden Markov Model is, let me also present a formal definition, which is as follows:

Definition Let $S = \{s_1, ..., s_N\}$ be a set of states and let $O = \{o_1, ..., o_M\}$ be a set of possible observations. A Hidden Markov Model is a statistical model characterized by a triple of parameters $\Lambda = \{A, B, \Pi\}$, where

- A is the set of state transition probabilities

$$A = \{a_{ij}\}, \qquad a_{ij} = P(y_{t+1} = s_j | y_t = s_i)$$

- B is the set of observation probabilities

$$B = \{b_i(k)\}, \qquad b_i(k) = P(x_t = o_k | y_t = s_i)$$

- Π is the initial state distribution

$$\Pi = \{\pi_i\}, \qquad \pi_i = P(y_0 = s_i)$$

Given a HMM as just defined, there are three basic problems that can be faced:

- **The Evaluation Problem** Given a HMM Λ and a sequence of observations $O = \{o_1, ..., o_M\}$, what is the probability $P(O|\Lambda)$ that the observations have been generated by the model?

- **The Decoding Problem** Given a HMM Λ and a sequence of observations $O = \{o_1, ..., o_M\}$, what is the most likely state sequence in the model that can have produced the observations?
- **The Learning Problem** Given a HMM Λ and a sequence of observations $O = \{o_1, ..., o_M\}$, how should we adjust the model parameters $\{A, B, \Pi\}$ in order to maximize $P(O|\Lambda)$?

Below I will discuss three algorithms to deal with these general problems.

1.3.1 *Forward-Backward Algorithm*

Remind that the Evaluation Problem consists in finding the probability $P(O|\Lambda)$ that the observations $O = \{o_1, ..., o_M\}$ have been generated by a HMM Λ. That requires summation over all possible state sequences. In fact, the probability of observing an output sequence $X_1^T = o_1, ..., o_T$ of length T is given by

$$P(X_1^T|\Lambda) = \sum_{Y_1^T} P(X_1^T|Y_1^T, \Lambda)P(Y_1^T|\Lambda), \qquad \forall Y_1^T = y_1, ..., y_T \quad (20)$$

where Y_1^T is a possible state sequence. This problem complexity grows exponentially with the length of observation sequence. Therefore, a standard algorithm could not to find its optimal solution in a finite time. Luckily, over the years a lot of techniques have been developed to deal with these computational issues, so that the Evaluation Problem can be efficiently handled by using the *forward-backward algorithm*. The forward-backward algorithm needs two kinds of variables to be defined, say α and β.
Let

$$\alpha_t(i) = P(x_1 = o_1, ..., x_t = o_t, y_t = s_i|\Lambda) \quad (21)$$

be the forward (α) values and let

$$\beta_t(i) = P(x_{t+1} = o_{t+1}, ..., x_T = o_T|y_t = s_i, \Lambda) \quad (22)$$

be the backward (β) values. Since forward values allow solving the problem through marginalizing, the probability of observing an output sequence of length T becomes

$$P(X_1^T|\Lambda) = \sum_{i=1}^{N} P(o_1, ..., o_T, y_T = s_i|\Lambda) = \sum_{i=1}^{N} \alpha_T(i) \quad (23)$$

The forward values $\alpha_T(i), 1 \leq i \leq N$ can be computed by means of dynamic programming through the following recursion:

$$\alpha_1(j) = \pi_j b_j(o_1), \qquad 1 \leq j \leq N$$

$$\alpha_{t+1}(j) = b_j(o_{t+1}) \sum_{i=1}^{N} \alpha_t(i) a_{ij}, \qquad 1 \leq j \leq N, \quad 1 \leq t \leq T-1 \quad (24)$$

The probability $P(O|\Lambda)$ required by the Evaluation Problem is

$$P(O|\Lambda) = \sum_{i=1}^{N} \alpha_T(i) \quad (25)$$

Similarly, there exists another dynamic programming recursion to compute $\beta_t(i)$:

$$\beta_T(i) = 1, \qquad 1 \leq i \leq N$$

$$\sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \qquad 1 \leq i \leq N, \quad 1 \leq t \leq T-1 \quad (26)$$

Since the following relation holds

$$\alpha_t(i)\beta_t(i) = P(O, s_t = i|\Lambda), \qquad 1 \leq i \leq N, \quad 1 \leq t \leq T \quad (27)$$

we found another way to compute $P(O|\Lambda)$, which makes use of both forward and backward values:

$$P(O|\Lambda) = \sum_{i=1}^{N} P(O, s_t = i|\Lambda) = \sum_{i=1}^{N} \alpha_t(i)\beta_t(i) \quad (28)$$

The complexity of the forward-backward algorithm is $O(T \times N^2)$, where T is the length of the observed sequence and N is the number of states.

1.3.2 *Viterbi Algorithm*

Another relevant problem as I pointed out is the Decoding Problem, that is finding the most likely state sequence in a HMM Λ that could have produced the observations $O = \{o_1, ..., o_M\}$. That problem can be efficiently solved thanks to the *Viterbi algorithm*.
Let $V_t(j)$ be the probability of the most likely state sequence responsible for the first t observations that ends in state s_j. This probability can be calculated through the following recursion:

$$V_1(j) = b_j(o_1)\pi_j, \qquad 1 \leq j \leq N$$

$$V_t(j) = b_i(o_t) \max_i (V_{t-1}(i) a_{ij}), \qquad 1 \leq j \leq N, \quad 1 \leq t \leq T-1 \quad (29)$$

The Viterbi path, namely the most likely sequence that could have produced the observed output, can be found by saving back pointers in order to remember which state $y_t = s_j$ was used to compute

$V_t(j)$. In order to save back pointers, let $Pointer(y_t, s_i)$ be the function that calculates the y_{t-1} values. This function is employed for the computation of $V_t(i)$ in the following way:

$$y_T = \arg\max_{s_i \in S} V_T(i) \tag{30}$$

$$y_{t-1} = Pointer(y_t, s_i) \tag{31}$$

Once y_T is found thanks to relation 30, all previous states can be retrieved thanks to the function *Pointer* (equation 31). For each time instant t, $1 \leq t \leq T$, the Viterbi algorithm can be seen as a search in a graph whose vertices are the HMM states.
The complexity of the Viterbi algorithm is $O(T \times N^2)$, where T is the length of the observed sequence and N is the number of states.

1.3.3 *Baum-Welch Algorithm*

The last well-known problem related to HMMs I mentioned before is the Learning Problem, namely the task of adjusting the parameters of a HMM Λ in order to maximize $P(O|\Lambda)$, where $O = \{o_1, ..., o_M\}$ is a sequence of observed outputs. To solve this problem, a powerful algorithm called *Baum-Welch algorithm* [87] is used, which utilizes the forward-backward algorithm.

Baum-Welch algorithm makes use of the variables α and β defined in 21 and 22 respectively. Furthermore, it needs other two variables to be defined, say ξ and γ, which can be expressed in terms of the above mentioned ones.
Let

$$\xi_t(i,j) = P(s_t = i, s_{t+1} = j|O, \Lambda) \tag{32}$$

or, equivalently,

$$\xi_t(i,j) = \frac{P(s_t = i, s_{t+1} = j, O|\Lambda)}{P(O|\Lambda)} \tag{33}$$

be the probability of being in state i and j at times t and $t+1$ respectively.
Further, let

$$\gamma_t(i) = P(s_t = i|O, \Lambda) \tag{34}$$

be the probability of being in state i at time t.
The ξ variables can be expressed in terms of α and β as follows:

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} \beta_{t+1}(j) b_j(o_{t+1})}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} \beta_{t+1}(j) b_j(o_{t+1})} \tag{35}$$

Similarly, the γ variables can be expressed in terms of α and β as follows:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \tag{36}$$

Comparing the equations 35 and 36 we can find that the relationship between them is given by

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j), \qquad 1 \leq i \leq N, \quad 1 \leq t \leq M \tag{37}$$

Once having defined these variables, let me describe the Baum-Welch algorithm:

1. Set the parameters $\Lambda = (A, B, \Pi)$ with random initial conditions.
2. Compute the forward (i.e. α) and the backward (i.e. β) variables by means of 21 and 22 formulas.
3. Compute ξ and γ variables by means of 35 and 36 formulas.
4. Update the HMM parameters according to the following equations:

$$\pi_i^* = \gamma_1(i), \qquad 1 \leq i \leq N \tag{38}$$

$$a_{ij}^* = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}, \qquad 1 \leq i \leq N, \quad 1 \leq j \leq N \tag{39}$$

$$b_j^*(k) = \frac{\sum_{\substack{t=1 \\ s_t = v_k}}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}, \qquad 1 \leq j \leq N, \quad 1 \leq k \leq M \tag{40}$$

These steps are repeated iteratively until reaching the desired level of convergence.
Notice that the algorithm converges at a local maximum. Global maximum is not guaranteed.
The complexity of the Baum-Welch algorithm is $Z \times O(M \times T \times N^2)$, where Z is the number of iterations, M is the number of observed sequences, T is the length of a sequence, N is the number of states.

2

THE STATE OF THE ART

This chapter will analyze some works which made use of Markov models in Text Mining branches as Information Retrieval (IR), Text Categorization (TC), Sentiment Analysis (SA). Subsequently, being this dissertation about Cross-Domain Sentiment Classification, papers about cross-domain techniques and Document Sentiment Classification will be described as well.

2.1 INFORMATION RETRIEVAL

The sudden explosion in the amount of unstructured data both inside documents and on the Internet has led to the need for effective and efficient automated methods that allow to recover information. Information Retrieval (IR) fulfills this purpose, being the discipline that deals with retrieval of unstructured data, especially textual documents, in response to a query which may itself be unstructured, like a natural language sentence, or structured, as a boolean expression. [18]
The ultimate goal of IR is satisfying the user information need. Unfortunately this ambition is not easily achievable, because it is not trivial to define what is relevant for a given query. There are plenty of factors that can affect relevance, like for example the time when query is being formulated, the particular user who needs information, and so on. Over the years, many researchers have developed methods and strategies to handle the Information Retrieval complexity.
Now we are going to take a look at some Markov Model based approaches used for this purpose.

2.1.1 *Markov Model based methods in Information Retrieval*

It has been a long time since Markov Chain was viewed as a good model to support Information Retrieval. Consider that when people are looking for some information, they execute a search hoping to immediately obtain the long result. Unfortunately, not always the first search attempt succeeds. In this unlucky case, it is necessary to refine the first submitted query in order to satisfy the information need. What happens is that a search goes through a sequence of states, start-

ing from the initial query and ending when either users find what they were looking for or they give up. An important achievable task was studied by Qiu [25], who used Markov Chains to discover the presence of patterns when humans search for information in hypertexts. Hypertexts can be naturally modeled as graphs, where pages or paragraphs represent states and links represent transitions between states. So this problem is strictly related to the one described above. It can be characterized by Markov Chain, which is a reasonable model because both it allows graph modeling and it permits prediction over the built graph. Qiu found the configuration that fits best this task is the second-order Markov model by using the log-linear model test. This means that search future state depends not only on the present state, but also on the immediately past state, while all previous states are assumed to not affect the search.
A different goal was pursued by Mittendorf and Schäuble [24], who encouraged making use of Hidden Markov Models (HMMs) in order to retrieve documents. In particular, HMMs have been employed to model two stochastic processes: the first one regards retrieving text fragments relevant with respect to a particular query, whereas the second one regards retrieving general text fragments (i.e. independent from any query). Their approach has several advantages. In fact, the position of occurrences of indexing features can be utilized for indexing, since they have conserved information about position of terms in documents. Further, from training data they automatically derived optimal weights for arbitrary features by using the Baum-Welch algorithm. Finally, building a framework of probabilistic retrieval, each document can be split into parts, either relevant or not depending on a specific query. However, relevance has been evaluated by means of scores, assigned by the Viterbi algorithm. So focus is on ranking rather than on relevant passages extraction. Explaining better, they did not clarify how effective their method is in extraction of passages that are not only query-dependent but also mutually coherent.
Miller et al. [22] focused on retrieving relevant documents given a query as well. They developed a Hidden Markov Model based method that outperformed standard *tf-idf* ranking. When a user submits a query, before presenting results, an automatic query expansion is performed to augment that initial query. In particular, after the first search, words belonging to some of the returned documents are used for the query expansion. The resulting augmented query will be used in order to perform a second search. Their Hidden Markov Model based method can also handle bigram productions, simply adding new document-dependent states to be considered when computing probabilities. Bigrams deal with the fact that words are not independent one from each other: differently, they can have different meanings depending on the context. On the other hand, there are aspects excluded from their model, such as for instance explicit syn-

onym modeling and concept modeling.
Unlike the previous works, Cao et al. [30] proposed a different approach to query/document expansion. First of all, they expanded both queries and documents rather than doing just one of these two tasks. Furthermore, the expansion process they performed took into account not only the directly related terms, but also indirect word relationships, thanks to the use of multi-stage Markov Chains. The idea behind this model is that after one step, we can only model relationships between directly related terms, while after more steps we can also take into account indirect relationships. Remarkably, when the number of steps approaches to infinite, the transition probability distribution is known as stationary distribution (or steady state) of the Markov Chain. Cao et al. showed the usefulness of their method, even if there are differences between query expansion and document expansion, because the former task achieved much better performance than the latter one, due to the document expansion task complexity. Despite of the interesting idea, there were also some issues in their approach, all related to the calculation of stationary distribution. In fact, it is not only difficult to establish if the process has converged, but there is also no proof that the algorithm produces just one single stationary distribution. Finally, the algorithm complexity is not negligible. All these limitations, as explained by Hoenkamp et al. in [19], can be circumvented if Markov Chain transition matrix is ergodic. In fact, ergodic chains have the important property that they reach the steady state (i.e. stationary distribution) after some steps, regardless the initial state. Consequently, it is simple to see if the process has converged. Moreover, the steady state is always unique and convergence is very fast. In order to be ergodic, chains have to be both aperiodic and irriducible. This is always true when creating Markov Chain transition matrix from a corpus of documents, using words as matrix states.
Like Cao et al. [30] did, Pan et al. [29] also thought about expanding both queries and documents. They worked in the context of Spoken Document Retrieval (SDR), employing a Markov decision process to model allowable state transitions. Their idea was that both system and users are essential in the retrieval process. In this approach, apart from the results, given the users query, the system also returns a list of ranked key terms, possibly selectable by the users in order to expand the query itself. These ranked key terms embody allowable state transitions and hence can be naturally modeled as a Markov process, whose initial state is obviously the users query. At each iteration, users can either terminate the session, because they are satisfied or give up, or iteratively choose one word in the key-list, according to which the query will be refined. Notice that this is feasible because every decision (i.e. taken by either the system or a user) affects the same Markov process.

A related work is that of Benoit [48], who described information seeking (IS) as a Markov process which handles user uncertainty. In fact, as previously said, it is quite natural considering that each user interaction starts, goes through a series of states dynamically influencing the system, and then terminates. Therefore, he suggested the integration of Markov Chains into dynamic information retrieval systems where both the system itself and the user are involved in the retrieval process. Markov Chains are useful to predict the process convergence, whereas user choices have the effect of modifying model transition probabilities. It is important to notice that thanks to these kind of information systems, the more users interact with the system the more the system will be effective in satisfying information needs.
Apart from document/query expansion, Markov models can also be used to accomplish other tasks. For example, in [47], Ghose et al. investigated the applicability of Markov Chains in the analysis of information derived from large amounts of data. They used a Markov process as main model, considering both entropy and conditional entropy functions to measure the information content. These measures are proved useful as transition probabilities of a Markov Chain, which instead represents the whole information stream. Therefore, each transition between two states corresponds to the information gain we obtain by moving from the former state to the latter one.
In [28], Xu and Weischedel utilized a Hidden Markov Model in the area of cross-lingual information retrieval. That is, starting from a query written in a language, also documents in another language could have been retrieved. Notice that it is unlikely to find an algorithm capable of finding exact translations, because same words can have many different meanings depending on context. Since HMM is a model based on probability, it is suitable to handle this kind of uncertainty. In particular, to accomplish this task their HMM based system estimates the probability that a query in one language could be generated from a document in another language. They chose performing query translation rather than documents translation because of flexibility, so that there was no need to change the index structure. Therefore, all their experiments were based on a HMM monolingual indexing.
One of the most known Information Retrieval application is the PageRank by Brin and Page [26], which aims to define a ranking of Web pages just dealing with their hypertext structure, regardless information about page content. The way they walked is modeling Web pages graph as a Markov Chain, whose states represent Web pages and whose edges transition probabilities between Web pages. The PageRank definition deals with random walks on graph, since it corresponds to the standing probability distribution of a random walk on the Web graph. This definition is recursive, that is, the PageRank of a Web page depends on PageRanks of other pages. In particular,

the more a page is referred by other pages the more its PageRank tend to be high. Brin and Page also dealt with the problem of trying to avoid loops in a small set of pages, because while a user is surfing the Internet it is unlikely he/she persists in looping always in the same small set of pages. This is the reason why they introduced an additional factor in the model, that is, a probability distribution embodying the fact the user, periodically, could get bored and randomly jump to another completely different Web page.
The Brin and Page idea was pursued by Neumann et al. [43], who introduced Markov Logic Sets (MLS) method, which relies on Markov random walks and logical features, to compute set completions. This method personalizes on the query the PageRank algorithm. As they explained, Markov Logic Sets method basically takes a query consisting of a small set of information and returns additional information. Particularly, it computes the relevance score by comparing the posterior (i.e. personalized) PageRank of an item given the query to the prior (i.e. uniform) PageRank of that item. They proved MLS performs effectively in retrieving relational set completions. However, currently their approach has the drawback, as they pointed out, to not deal with the problem of setting the size of completion (i.e. they can get any amount of additional information).
In [23], Sarukkai performed probabilistic World Wide Web link prediction and path analysis using Markov Chains. He used Markov Chain states to model URLs and Markov Chain edges to model transition probabilities between URLs, which have been estimated based on navigation history. As a matter of fact, Markov Chains allow the system to dynamically model the URL access patterns that are observed in navigation logs based on the previous state. Therefore, simply adding information regarding the navigation history for a particular client, web servers can predict the probabilities of HTTP requests. Notice that, differently from PageRank, this model relies on user dependent information in order to define the page relevance. Another important application, achievable if we know history of visits for a given user, concerns in suggesting him/her which other sites might be of interest, namely what is also known as Recommendation. Moreover, Markov Chains can be used in a generative mode to automatically obtain navigation tours (i.e. a sequence of URLs accessed by a user). Finally, Markov Chain transition matrix can be analysed by using eigenvector decomposition in order to identify possible hubs and authorities (i.e. important sites where to extract information from). Besides this, Sarukkai pointed out some limitations of his approach and more generally of Markov Chains, which need a huge amount of training data and are not very scalable due to the dimensionality ($N * N$, if N is the number of states) of the Markov Chain transition matrix. However, the latter problem can be softened just by considering that Markov Chain transition matrix is usually sparse. Hence,

sparse matrix representations can address the curse of dimensionality.

2.2 TEXT CATEGORIZATION

Nowadays on the Internet we can find almost any kind of data. There are texts talking about economy, politics, sports, books, movies, and so on. If we were interested in news about sports for example, we would like to retrieve documents regarding every sports. IR techniques could help us in finding most relevant documents according to the query *sport* as we have already seen, possibly expanding this query. Nevertheless, once users have chosen a Web page, again they could find heterogeneous information because that Web page discusses not only about sports but also about different topics or maybe because it contains disordered information about many sports. Therefore, it would be important to categorize information so that we could split texts according to their topic. Beyond general arguments, inside a text could be argued even more specific topics. Consequently, we might hierarchically organize information.
Now imagine we have a corpus of documents and a set of categories representing topics, either hierarchically structured or unstructured. Each document belongs to a certain subset of categories. When a new document is added to the collection, it would be valuable to automatically assign categories to that document, according with its content. Text Categorization (or Text Classification) is the task fulfilling this purpose.

2.2.1 *Markov Model based methods in Text Categorization*

One of the first attempts of using Markov based models in the context of Text Categorization is the result of Yi and Beheshti's work [39], who investigated the applicability of HMMs in categorizing digital documents with the aid of library classification and subject headings for training purposes. Yi and Beheshti made use of a HMM for each different category. They proposed a HMM architecture with one state for each source of information used in order to model the particular category. In this model the transition probabilities between states are seen as the information quantity each source of information has, with respect to a particular category, and are calculated by means of *tf-idf*. The idea behind information sources is that they model different aspects and, as a consequence, they allow increasing the information power. Moreover, if it was useful adding other sources of information by need, it would be feasible thanks to the flexibility of their HMM architecture.
In a following work [32], Yi and Beheshti proposed a HMM method to classify medical documents. They used Medical Subject Heading

(MeSH), a hierarchically organized thesaurus of medical subject headings [81], as prior knowledge in their model in order to improve HMM method performance. As in their preceding work [39], their Hidden Markov Model is represented as a fully connected graph having one state for each source of information (e.g. MeSH is a source of information). Moreover, every state contains all preprocessed words derived from the particular source of information. Since it would be meaningless assigning probability values while passing from one source of information to another, all the transition probabilities between states are equal, in spite of what is usually done with HMMs and in spite of what they did in [39]. The main advantage of this approach is always flexibility, because the model could be extended just by adding a new state for each new available source of information. Notice that, due to the previously mentioned assumption about state transition probabilities, there is no need of a training set in order to calibrate them.

Vieira et al. [36] developed a HMM based classifier, following the same strategy applied by Yi and Beheshti [32] in order to perform classification, that is, using a HMM as a document generator. The idea is, for each given document, to evaluate the probability of being generated by each HMM, finally assigning to that document the class corresponding to the HMM maximizing this probability. Differently from Yi and Beheshti [32], whose model introduced one state for each source of information, Vieira et al. used Markov Chain states to encode words relevance. As they explained, the first state has the most relevant words in the corpus as possible outputs and so on regarding the following states, which encode observations of decreasing relevance. This approach reaches better results than Naive Bayes and comparable performance with SVM. Despite modeling differences in words relevance, Vieira's method does not establish how much a word is more relevant than another.

As in [36], Li et al. [38] thought about creating a different HMM for each category. Each HMM has an initial state, a state transitions probability distribution and an observation probability distribution. These parameters require a huge amount of training data in order to be estimated. After having built the model, a new document is labeled as usual by evaluating the probability of being generated by each HMM, finally assigning to it the class corresponding to the HMM maximizing this probability.

Xu et al. [31] combined HMMs with Text Categorization techniques in order to improve medical evidence retrieval in Randomized Clinical Trials (RCT) papers. A common problem in this area is that abstracts are often unstructured, while it would be simpler extracting the most relevant information if they were structured. To accomplish this task, they created a Hidden Markov Model having one state for each considered category (i.e. Background, Objective, Methods, Re-

sults, Conclusion), transforming the sentence labeling problem into a HMM sequence alignment problem. The transition probabilities between these 5 states have been estimated on the training set. In particular, they have been assigned according to the number of times a sentence belonging to a section is followed by another sentence belonging to a different section. Further, state probabilities have been computed through the output probability distribution produced by a standard classifier (e.g. Naive Bayes). Then, the final classification has been performed on the HMM by means of the Viterbi algorithm, which allows computing the most likely sequence of states for all sentences. They showed their approach reaches high performance (precision = 94% and recall = 93% on average).

Zhou and Li [34] extended the basic idea of random walk on a graph by introducing the concept of reward. They assumed distance between vectors is based on Euclidean norm and considered the angle between vectors as a correlation factor, which is directly employed as reward in their model. It is important to note that the graph model can be viewed as a Markov Chain, whose transition matrix has been used to perform classification. Zhou and Li showed the results are already good with this simple model, but even better if accumulating rewards walk by walk. In fact, while the first memoryless version is affected by noise, this advanced version is more robust.

In [37], Frasconi et al. proposed a HMM based model for Text Categorization to rely on correlation between documents to be classified, with reference to the case of multi-page documents. The idea is modeling each page as a different bag-of-words and designing a HMM which could link concepts in different pages. This approach aims to exploit augmented contextual information, which deals not only with the single page bag-of-words but also with relationships between pages. They proved it is possible to improve classification accuracy by taking between pages relationships into account. On the other hand, they assigned a category to every page, assuming the coincidence between category boundaries and page boundaries, which is not always true because a single page may contain more than one concept and consequently it should be mapped to more than just one category.

Li et al. [40] presented a completely different Markov based model for classifying texts. Rather than dealing with a bag-of-words representation of documents, they chose another model able to express sequential information. Inter-words dependencies have been modeled by means of Markov Chains with diverse order values (e.g. a k order Markov Chain can model dependency of a certain word with its previous k terms in the same document), which could also be combined together to improve performance. Based on this choice, they invented a novel adaptive procedure to classify documents, which allows dynamical switching between different order Markov Chains.

This feature makes this method able to model different length word dependencies. Another benefit related to this is that all preprocessing tasks are not required.
Beyond directly using HMMs to perform Text Classification, Li and Dong [41] also utilized them to model inter-cluster associations. More precisely, they clustered words in documents by means of *tf-idf* for dimensionality reduction purposes, mapping each cluster to a Markov Chain state. Instead, document classification is executed as usual by considering n HMMs (i.e. one for each class) and computing the probability that the observed sequence is produced by each HMM. Of course, a document will be assigned to the most likely category. It is important to note that this method and, in general, all previously mentioned ones where a different HMM is required for each class have a linear complexity, which grows with the number of categories.
Wren et al. [42] relied on a n-gram Markov model in order to classify DNA sequences. A DNA sequence is a succession of tags (i.e. characters) that constitute a word. Each word has a different length depending on if it is a nucleic acid, a peptide abbreviation or a symbolic peptide string. The approach they followed is the classical one, that is, constructing a HMM with one state for each possible tag, estimating all state transition probability between tags. Then, when classifying a word they assigned the label according to the most likely sequence of tags that could have generated that particular word.

2.3 OPINION MINING AND SENTIMENT ANALYSIS

Everyday it happens that someone needs suggestions or guidelines about something. For example, if we wanted to buy some products, knowing in advance which ones have been proved to be the most reliable or the best working would surely be a valuable information. It is essentially for this reason that product reviews are written. They usually describe in a detailed way product characteristics and give readers suggestions comparing either goods or their individual parts with other alike artefacts, possibly made by different brands. Similar considerations could be done with reference to movie reviews, book reviews and so on. However, reviews are just one precious source of opinions. In fact nowadays we can find sentiment everywhere thanks to the widespread use of the Internet, since people daily write comments about everything happens in the world. So today the Internet has probably become the main source of information.
Both medium-sized enterprises and large companies are very interested in knowledge extraction in order to improve their business, but analysis cannot be done simply reading each article, comment or review, due to the huge quantity of data stored in servers all around the world. This is basically the reason why we need automated systems capable of performing large-scale analysis.

Opinion Mining (OM) or Sentiment Analysis (SA) is a discipline framed within the area of Natural Language Processing (NLP) that can be viewed as the computational treatment of opinions, feelings and subjectivity in texts [1]. An opinion is simply a positive or negative sentiment, emotion or appraisal about an entity or an aspect of the entity expressed by an opinion holder. Analyzing texts we could find several opinions and it is not uncommon to detect also comparative opinions, where a comparison is made either between some entities or between some specific characteristics of entities [2, 50].
One of the most studied topics in the literature [1] related to Sentiment Analysis is Document Sentiment Classification, or simply Sentiment Classification. The goal is to classify a document (e.g., a product review) as expressing a positive or negative opinion. The main assumption is that every document expresses opinions on a single entity and that it is written by a single opinion holder. This is typically true with regard to reviews, but not if we are considering forum or blog posting. In literature we can find plenty of supervised techniques, but unsupervised methods are used as well in Document Sentiment Classification. In the supervised version, Sentiment Classification can be formulated as learning problem with three classes: positive, negative and neutral. Dataset could be split into a training set and a test set and then analyzed applying a traditional classification algorithm (e.g., SVM, RandomForest, IBk, etc.). Although Document Sentiment Classification is useful in order to extract opinions from reviews, it is not simply applicable in other domains. In fact, on the one hand forum and blog postings typically evaluate more than one entity, whereas we pointed out that Sentiment Classification can only detect the general polarity inside a document. On the other hand, these traditional techniques are not always effective when we try to classify short texts like for example tweets, because of the lack of a large number of words. Therefore, more complex techniques need to be developed in order to deal with these critical issues that can negatively affect Sentiment Classification.

2.3.1 *Markov Model based methods in Opinion Mining and Sentiment Analysis*

In Sentiment Analysis, Markov Chains are typically used to model word dependencies and above all having as main goal to find out opinion words (i.e. words that express sentiment). For example, in [3] Li et al. made use of a Markov Chain for this purpose. Their method aims to establish words polarity even before classifying the whole document sentiment. So a Hidden Markov Model is used, where the observation layer is made of all words whose sentiment has to be extracted, whereas the sentiment layer is hidden (i.e. has to be discovered). According to what has been stated before, their

Markov Chain is capable of modeling word dependencies, because each word depends on its previous one due to the Markov property. Moreover, they relied on some knowledge bases to guess words polarity. However, every word that does not match with any entry will be assigned to a random label rather than using other information for the sake of guessing words polarity. Since they correctly said that words inside a sentence usually retain the same polarity while a negative conjunction is found, a probably better alternative could include these conjunctions in the model.

Another approach to Document Sentiment Classification is described by Jin et al. in [27], who integrated in a HMM framework both Part-Of-Speech (POS) tagging and lexicalization. The HMM they proposed is provided with a state for each couple {word, POS tag} and it basically outputs what they called hybrid tag, which is a more complex tag taking into account word relationships and positions. Furthermore, they also investigated the information propagation problem, using Microsoft Word's thesaurus to find synonyms, antonyms, related words and also considering bigrams. Finally, they used the Viterbi algorithm to estimate the most likely sequence of hybrid tags. This model not only deals with some linguistic features, but it has also the merit of identifying not frequently used expressions thanks to the utilization of Microsoft Word's thesaurus. On the other hand, the underlined method has some trouble in distinguishing opinions about different objects in the same document. This could be an issue because if we think about the common case of reviews, it is not rare that a reviewer compares the product he/she is reviewing with another one.

As pointed out, POS tagging is an outstanding helpful task in the context of Opinion Mining and Sentiment Analysis. Markov models have been widely used to accomplish this task. A standard way to cope with this problem has been used by Nasukawa and Yi [35], who performed POS tagging following the description in [82]. The idea is that a Markov model is used to represent the sequence of tags or, more precisely, each Markov Chain state corresponds to a tag. When a tagging for a sequence of words is demanded, the goal is to find the most likely sequence of tags for that sequence of words. So, since tags are mapped into Markov Chain states, this means computing the most likely sequence of Markov Chain states, which could be calculated via Maximum Likelihood Estimation by means of the Viterbi algorithm.

Mei et al. [33] with their Topic Sentiment Mixture (TSM) tried jointly modeling topics and sentiments by means of a Hidden Markov Model. They intended tagging each word with a sentiment label in the direction of extracting the polarity of an entire document. To fulfil this purpose, their HMM has one state for each topic, which is fully connected with all its subtopics, in turn connected to other two states

modeling polarity (i.e. positive and negative). All transition probabilities and output probabilities between states are estimated by using the Baum-Welch algorithm, while the most likely sequence of topics and related sentiments is computed through the Viterbi algorithm. Despite of attempting to model together topics and sentiments, they are represented by using two different language models.
Mei's approach was outperformed by Jo and Oh in [45], whose Aspect and Sentiment Unification Model (ASUM) integrated both topics and sentiments in the same language model. In this way it is possible to understand how much a word is correlated with particular topics and sentiments. ASUM is an extension of Sentence-LDA, which is in turn an extension of the classical LDA (Latent Dirichlet Allocation) [79]. Once again, beyond other interesting aspects, what involves us is that all the aforementioned are graphical models having Markov Chains as basic model. A big benefit of ASUM is the capability of extracting different sentiments regarding the same aspect. On the contrary, as the authors revealed, they assumed each sentence contains exactly one aspect, which is not always an acceptable hypothesis. Moreover, they did not make use of a POS tagger, which in other works has already shown being helpful and which could facilitate the identification of both attributes (often expressing sentiments) and conjunctions/negations (relevant in individuating either different aspects or changes in polarity inside a sentence).
So far we spoke about the utilization of HMMs with reference to English language. Anyway we could notice that HMMs are quite general and therefore they could be extended to other languages with little modifications. For example, in [44] Zhang et al. applied a Hierarchical Hidden Markov Model (HHMM) based technique to Chinese language. A HHMM is more complex than a HMM, because a single state in a higher level HMM forms another lower level independent HMM. Further, the main difference between Chinese and English is that in the former a sentence is written without whitespaces and thus a word segmentation process is needed. They proved a class based word segmentation process, in addition with n-shortest-path (NSP) algorithm, can solve word segmentation ambiguity. As a consequence, their HHMM based method has been shown effective in Chinese lexical analysis.
Slightly different areas in which Markov based method can be utilized are speech recognition [80] and gesture recognition [49], because of the effectiveness in modeling human communication dynamism. Morency et al. [46] exploited this idea using a HMM based approach in order to classify the polarity of a Youtube video clip. In their model, each Markov Chain state represents a spoken utterance in a Youtube video clip. Every utterance is modeled by using tri-modal features, i.e. including text, audio and video in the Hidden Markov Model. In spite of the not so good performance in video sentiment

classification, the authors showed the utility of tri-modal features in comparison with models based only on one of the three kind of features.
A not less interesting work was made by Choudhury et al. [51], who investigated Texting Language (TL), that is the compressed language often used in SMS, informal chats, and so on. They aspired to translate compressed language into its standard form, reaching this goal with more than 80% of accuracy. To accomplish this task, a HMM was built for each word that is used to be abbreviated in TL. Markov Chain states represent graphemes and phonemes, utilized to deal with any possible modification of the standard word. Both state transition probabilities and state observation probabilities have been estimated based on the training data, with some tricks for estimating values for unseen words as well. The translation from the compressed word to the standard one is obtained by applying a variant of the Viterbi algorithm. It is important to note that this work is orthogonal to Information Retrieval, Text Categorization and Sentiment Analysis tasks and hence it could be useful as base step in other methods.

2.4 CROSS-DOMAIN TEXT CATEGORIZATION

Before, we have already spoke about Text Categorization. Remind that it consists in organizing textual documents into a user-defined taxonomy of categories or classes. Categories usually correspond to the topics discussed in those texts, such as politics, sport, music, and so forth. As already pointed out, this task is helpful to put documents in order, organizing them hierarchically. As a consequence, Text Categorization can be the basic step of other methods, such as spam filtering, Sentiment Analysis and in particular Document Sentiment Classification. In a standard Text Categorization task, a training set of labeled documents is demanded in order to build a model, which can be subsequently used to classify another set of unlabeled documents, named test set. Formally, let $C = \{c_1, ..., c_M\}$ be the set of available categories. Then, let us imagine we have a set of documents $D_{train} = \{d_1^{train}, ..., d_{N_s}^{train}\}$, whose labels are available, and a set of documents $D_{test} = \{d_1^{test}, ..., d_{N_t}^{test}\}$, whose labels are unknown. The goal is assigning to each document $d_i^{test} \in D_{test}$ a class $c_j \in C$.
For the sake of being effective, the Text Categorization process should rely on a reasonably dimensioned training set, which in turn should have the same classes of test set documents. A dictionary of strongly related terms between training and test is required as well. In real world, the main problem is that not always training set is available or, anyway, its manual construction inevitably requires non-negligible time. Sometimes, it may happen that we have a training set about a

Author(s)	Task	Markov Chain Usage	MC Nodes	MC Edges
Qui [25]	Pattern discovery in user's search.	RMC modeling Web pages.	URLs	Transition probabilities between URLs.
Miller et al. [22]	Document Retrieval	HMM modeling the generation of a query. One HMM for each document. At each state, a word is generated.	A state modeling a word from the document. A state modeling a word not in the document. Other states for synonyms, etc.	Transition probabilities between words.
Cao et al. [30]	Query (Document) Expansion	HMM modeling query (document).	Terms in query (document).	Transition probabilities between terms, based on co-occurrences between terms.
Pan et al. [29]	Query Expansion	RMC modeling the generation of a query.	Terms in query.	Transition probabilities between terms.
Benoit et al. [48]	Document Retrieval	RMC modeling user uncertainty in Information Seeking (IS).	Current state in user-system interaction.	Transition probabilities between states in IS process.
Sarukkai [23]	Web link prediction and path analysis	HMM modeling user access to Web pages.	URLs	Transition probabilities between URLs.
Brin and Page [26]	PageRank	RMC modeling Web pages.	URLs	Transition probabilities between URLs.

Table 3: Summary of the usage of Markov Chains in the main Information Retrieval works.

Author(s)	Task	Markov Chain Usage	MC Nodes	MC Edges
Yi and Beheshti [39]	Text Categorization	HMM as document generator. One HMM for each category.	Sources of information.	Transition probabilities between different sources of information, computed using *tf-idf*.
Yi and Beheshti [32]	Text Categorization	HMM as document generator. One HMM for each category.	Sources of information.	Transition probabilities between different sources of information. Starting from a state, all future states are equally likely.
Vieira et al. [36]	Text Categorization	HMM as document generator. One HMM for each category.	HMM states represent words. First states encode most relevant words in the corpus, and so on until reaching last states, which encode least relevant words.	Transition probabilities between words.
Xu et al. [31]	Text Categorization	HMM modeling sequence types in Randomized Clinical Trial (RCT) abstracts.	Sentence types (Background, Objective, Methods, Results, Conclusion).	Transition probabilities between states.
Frasconi et al. [37]	Multi-Page Documents Categorization	HMM linking concepts in different pages.	A state for each unique page category.	Transition probability between source state and target state.
Li and Dong [41]	Text Categorization	HMM modeling inter-cluster associations. Each cluster contains some words in a document. A HMM is built for each category.	A state for each cluster.	Transition probabilities between states.
Wren et al. [42]	DNA sequences Categorization	HMM modeling sequences of tags. Each tag constitutes a word (i.e. a particular amino acid).	Tags.	Transition probabilities between adjacent tags.

Table 4: Summary of the usage of Markov Chains in the main Text Categorization works.

Author(s)	Task	Markov Chain Usage	MC Nodes	MC Edges
Li et al. [3]	Document Sentiment Classification	HMM modeling word dependencies.	Words	Transition probabilities embodying word dependencies.
Jin et al. [27]	Document Sentiment Classification	HMM modeling both Part-Of-Speech (POS) tagging and lexicalization.	Couples {word, POS tag}	Transition probabilities between couples.
Nasukawa and Yi [35]	POS tagging	RMC modeling the sequence of tags.	Tags	Transition probabilities between tags. These probabilities are related to co-occurrences in the training set.
Mei et al. [33]	Document Sentiment Classification	HMM jointly modeling topics and sentiments.	A state for each topic, fully connected with all its subtopics, in turn connected with other two states modeling polarity.	Transition probabilities between states.
Choudhury et al. [51]	Texting Language	HMM deals with the problem of translation from compressed language to standard language. A HMM is built for each word.	Graphemes and phonemes.	Transition probabilities between consecutive graphemes or phonemes.

Table 5: Summary of the usage of Markov Chains in the main Sentiment Analysis works.

certain topic and we need classifying a test set about a similar theme, possibly the same, but described with different terms. To deal with this problem, many techniques have been developed in the context of Cross-Domain Text Categorization (CDTC), where information extracted from a source domain is used to classify unlabeled documents belonging to a target domain. CDTC is framed within transfer learning approaches (a great survey about transfer learning can be found in [63]), because a mapping is needed to adapt knowledge obtained from source domain to information to be utilized in target domain, since source and target usually express related concepts by using different words. CDTC is a demanding task, so that all previous methods include sophisticated statistical notions and techniques and require strong parameters calibration.

In literature, many different approaches, both supervised and unsupervised, have been proposed to deal with Cross-Domain Text Categorization problems. When training set labels are known, supervised machine learning approaches have been employed by Joachims in [64], Dumais et al. in [65], Yang and Liu in [66] and Sebastiani in [67]. Instead, when training set labels are unknown, unsupervised techniques have been used, like for example those illustrated by Merkl in [68] and by Kohonen et al. in [69]. In spite of being less powerful than supervised methods, these ones allow finding related documents anyway. The most used documents representation is known as bag-of-words model [67], where a document is seen as a set of words having certain weights. Several techniques have been advanced during the years for the sake of improving this basic model capability, like for instance Latent Semantic Indexing [70] and Latent Dirichlet Allocation [79]. On the other hand, external sources of information, such as the WordNet database [71] and Wikipedia [72] have been used too.

As previously said, transfer learning approaches are required to map source domain to target domain. More specifically, Pan and Yang identified in their survey [63] two transfer mode: instance-transfer and feature-representation-transfer. The former aims to bridge the inter-domains gap by adjusting instances from source to target. Vice versa, the latter pursues the same goal by mapping features of both source and target in a different space. In the Text Categorization context, transfer learning has been fulfilled in some ways, such as through clustering together documents and words [73], through extending Probabilistic Latent Semantic Analysis also to unlabeled instances [74], through extracting latent words and topics, both common and domain specific [75].

2.4.1 *Document Sentiment Categorization approaches*

Similar to Text Categorization, Document Sentiment Classification (DSC), or simply Sentiment Classification, is a particular classification task where categories represent document polarities. For this reason, DSC is also related to Sentiment Analysis, being focused on opinions (e.g. positive, negative, neutral). Apart from the aforementioned ones, involving Markov Chains, a number of different techniques have been developed over the years solely for Document Sentiment Classification. For example, Dave et al. [4] draws on Information Retrieval methods for feature extraction and to build a scoring function based on words found in positive and negative reviews. Tan et al. [6] considered a set of commonly used words in expressing sentiment. They employed it to label a portion of informative examples from a given domain in order to reduce the labeling effort and to use the labeled documents as training set for a supervised classifier. Qui et al. [7] utilized a similar approach: in fact they also relied on a sentiment dictionary to classify some reviews and generate a training set. Melville et al. [8] presented a framework where some lexical information about associations between words and classes can be exploited and refined for specific domains by means of training examples to enhance sentiment analysis accuracy. Paltoglou et al. [5] employed some variants of the well-known *tf-idf* term weighting scheme, using a sublinear function for term frequency weights and document frequency smoothing. Deng et al. [9] proposed a supervised term weighting scheme based on both the importance of a term in a document and the importance of a term for expressing sentiment. Regarding the former they outlined that most of the high-performance approaches introduce some kind of normalization of term frequency, whereas concerning the latter they found the best results using either mutual information or odds ratio as statistical functions. Wu et al. [10] introduced the concept of over-weighting, claiming that all previous works suffer from that problem. To address the issue they proposed two regularization techniques called singular term cutting and bias term. Finally, they also pioneered a new supervised term weighting scheme: regularized entropy.

In the same way of Text Categorization, the main issue in Document Sentiment Classification regards cross-domain problems. In particular, it is likely that people use different words to express opinions in diverse domains. In fact, if we are considering reviews about books and electrical appliances, there will be many domain specific terms expressing positive and negative orientation. For example, a book can be *interesting*, *boring*, *funny*, but the same attributes are meaningless in the description of electrical appliances. On the other hand, an electrical appliance can be *efficient*, *noisy*, *clean*, but again these attributes could not be used for reviewing books. Therefore, what we need is

a sort of mapping between terms appearing in source domain and those in target domain.
Since classifiers trained in one domain do not usually perform well in others, researchers have attempted to address the Cross-Domain Document Sentiment Classification problem in several manners. Aue et al [11] tried some approaches to customize a classifier to a new target domain and discussed their performance. Yang et al. [15] used knowledge transfer to enable cross-domain learning. In [13], Blitzer et al. discovered a measure of domain similarity contributing to a better domain adaptation. In [14] Pan et al. advanced a spectral feature alignment algorithm which aims to align words belonging to different domains into same clusters, by means of domain-independent words. The clusters form a latent space which can be used to improve sentiment classification accuracy of target domain. He et al. [56] extended the Joint Sentiment-Topic (JST) model by adding prior words sentiment, thanks to the modification of the topic-word Dirichlet priors. Bollegala et al. [12] suggested the adoption of a thesaurus containing labeled data from source domains and unlabeled data from both source and target domains. Since the thesaurus can be employed to measure similarity between words, they used it to expand feature vectors in both training set and test set.
In all the previously mentioned techniques some kind of supervision is applied. On the other hand, Document Sentiment Classification may be performed as well by using unsupervised methods. In the latter case, features are overwhelmingly commonly used words in expressing sentiment. For instance, in [17] Turner exposed an algorithm to classify reviews either in positive or negative. He basically evaluated mutual information between the given sentence and two words taken as reference: *excellent* and *poor*. Another interesting work in sentiment extraction from text is that of Taboada et al. [16]. They not only built dictionaries of words annotated with both their semantic polarity and their weights, but also incorporated intensification and negation.
Although Document Sentiment Classification is useful in order to extract opinions from reviews, it is not simply applicable in other domains. In fact, on the one hand forum and blog postings typically evaluate more than one entity, whereas we pointed out that Sentiment Classification can only detect the general polarity inside a document. On the other hand, these traditional techniques are not always effective when we try to classify short texts like for example tweets, because of the lack of a large number of words.

3

THE PROPOSED MARKOV CHAIN BASED METHODS

In this chapter, I will define new Markov Chain based methods. At the beginning, I will introduce the basic model all the presented approaches are built on. Later, I will illustrate how to create the new Sentiment Classification algorithm based on the aforementioned model. Finally, I will show you some variants of this algorithm, trying to motivate the reason behind each different method.

3.1 A BASIC MARKOV MODEL

Let us remind that this work focuses on Sentiment Analysis and in particular on Document Sentiment Classification (DSC). DSC is the task of classifying an opinion document (i.e. a document expressing opinions about something) discussing about a particular topic, as positive, negative or possibly neutral. In a formal fashion, let us give a definition of what DSC is.

Definition Let $D = \{d_1, ..., d_N\}$ be a corpus of documents talking about a certain topic. Let $C = \{c_1, ..., c_M\}$ be a set of categories, each one representing a possible document polarity. Document Sentiment Classification (DSC) is the task that consists in assigning to each document $d_i \in D$ a class $c_j \in C$.

In practice, DSC is a specific kind of Text Categorization where classes are document polarities. According to the given definition, we may notice that the implicit assumption DSC does is considering just the general document polarity, ignoring the potential presence of opinions, possibly conflicting, regarding either more than one entity or more than one aspect related to the same entity. This assumption could be feasible or infeasible depending on what we are looking for. Although any set of polarities can be used for the DSC task, the most meaningful ones as well as the most used in literature are $C = \{positive, negative\}$ and $C = \{positive, negative, neutral\}$. In fact, when people express opinions (i.e. opinion holders, according to the definition in [85]) about something in real contexts, their point of view could have not only a positive or a negative orientation, because they use words that make other readers understand that they take

a particular side, but also a neutral orientation, because either the terms they utilize are quite general or they make use of expressions showing indecision/equilibrium, like *pros and cons* or others. So the 3-classes problem is probably the one describing better what happens in real world. However, if our goal was either understanding if people are generally satisfied/unsatisfied about a certain topic or identifying positive and negative opinions to further investigate the reason why they are for or against the specified topic, then considering a 2-classes problem could be meaningful as well. However, as I said before, other categories could be employed in the DSC process. For example, regarding opinions on TripAdvisor [1], reviewers give a score from 1 to 5 stars to products, places, etc. Besides this, they also write a review. Thus, a meaningful task is, for instance, building a model capable of classifying that review in 5 classes corresponding to stars. Alternatively, this 5-classes problem could be mapped into the previous 3-classes problem, by mapping 5 and 4 stars in the positive class, 3 stars in the neutral class and 2 stars and 1 star in the negative one.

Anyhow, we may notice that in DSC tasks a central problem is finding the so called *opinion words*, namely the terms that show a strong orientation towards a particular class. This means that they guide the classification process, aiding to categorize documents according to their polarity. These terms are usually frequent inside a corpus but at the same time they should not be found in every document, because in that case they would not help in discriminating among different categories. Beyond being frequent, opinion words should also be significant inside a particular document in order to spread their polarity out to the entire document. In accordance with this consideration, their weight should be higher than that of other less important words. We will see some examples of widespread methods to assign a meaningful weight to a term belonging to a corpus of documents, but at the moment let us just focus our attention on the fact that the most relevant terms have the highest weights. Of course, it is infeasible that all the words inside a corpus of documents have the above mentioned characteristics. On the other side, also a less important word could support the classification process in some ways. We will refer to these kind of words as *support words*. So, being opinion words the task driving force, we could think linking support words to them for the sake of increasing the discriminative power while classifying.

In line with what has been stated until now, two main questions arise:

1. How to link words together?

2. How to link a particular word with the document label?

1 http://www.tripadvisor.com

In order to answer to the first question, a straightforward, widely used idea is modeling words co-occurrences inside documents. The idea is that the more two terms co-occur in documents the more their connection will be stronger. Consequently, if one of the two is an opinion word, it not only leads the categorization process of a document towards an orientation, but also pushes other words to do the same. We could see the words co-occurrences modeling as a graph whose nodes represent words and whose edges represent the strength of the connections between two different words (an example is shown in figure 4).

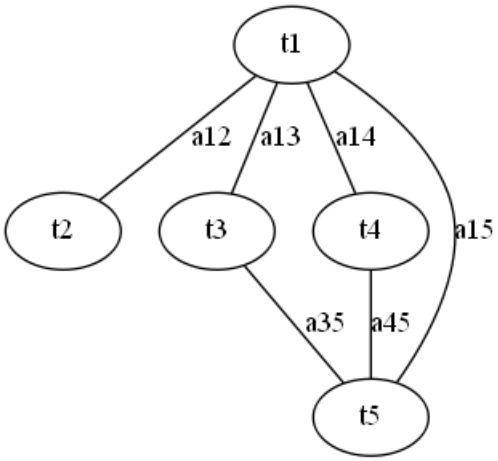

Fig. 4: *The figure shows an example of undirected graph used to model co-occurrences between terms. Each arc between two nodes t_i and t_j has weight a_{ij}.*

Formally, let $D = \{d_1, ..., d_N\}$ be the corpus of documents and let $T = \{t_1, ..., t_k\}$ be the dictionary of terms we are considering. Let $A = \{a_{ij}\}$ be the set of edges between terms, i.e. the set of connection weights between terms t_i and t_j. Then, one meaningful way to compute term co-occurrences is

$$a_{ij} = a_{ji} = \sum_{d=1}^{N} w_{t_i}^d \cdot w_{t_j}^d, \qquad \forall i \neq j \tag{41}$$

where $w_{t_h}^d$ represents the weight of term t_h in document d. In fact, analyzing the relation, we could notice that if two terms co-occur in a document, the connection between them will be strengthen. Moreover, the higher their weights are the more the connection will be strengthen. Finally, if two terms do not co-occur in a document means that at least one of them is absent (of course, $w_{t_i}^d = 0$ if t_i does not exist in document d) and, consequently, the connection will not be strengthen because $w_{t_i}^d \cdot w_{t_j}^d = 0$.
The same strategy could be followed to link a certain word with a particular class, that is, finding the polarity of a certain word. Clearly, to

accomplish this task, knowledge about document polarity is mandatory, unless having an external knowledge base which tells us that a word is intrinsically positive, negative or neutral. In any case, let us imagine we know document polarities inside a corpus. The idea is modeling co-occurrences between a word and a category for every document belonging to the corpus, namely augmenting the strength of the relationship between a word and a class if that specified word appears in a document of that particular class. Again, these co-occurrences can be modeled as a graph. Since we already have a node for each word, we just need adding another node for each class, linking by arcs every couple word-class where that word appears at least in one document belonging to that class (the same example in figure 4 with also class nodes is shown in figure 5).

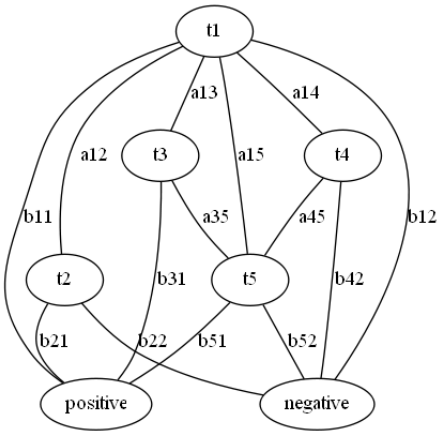

Fig. 5: *The figure shows the same example already shown in figure 4, to which positive and negative classes have been added. From this example, we could notice that t_3 is not linked to negative class and t_4 is not linked to positive class. This means the former never appears in negative documents, whereas the latter never appears in positive documents.*

Formally, let $C = \{c_1, ..., c_M\}$ be a set of categories. Let $B = \{b_{ij}\}$ be the set of edges between a term t_i and a class c_j. The weight between term t_i and category c_j is

$$b_{ij} = \sum_{d=1}^{N} w_{t_i}^d, \qquad c_d = c_j \tag{42}$$

where c_d is the category value of the document d.
Once having represented co-occurrences by means of an undirected graph, we could notice that it is possible to transpose this model to a Regular Markov Chain, where graph vertices are simply mapped to MC nodes and graph edges are split into two directed edges (i.e. the edge linking states t_i and t_j is split into one directed edge from t_i to

t_j and another directed edge from t_j to t_i). Though, this transposition is not enough to say that the model we have produced is a sound MC, because we need proving that the Markov property holds, along with other basic properties related to probability. First of all, probability non-negativity must hold both for states and for arcs, as stated by relations 3 and 4. The former holds because the model includes just states corresponding to terms in the analyzed corpus. The latter holds as well because transition probabilities (modeled by arcs) are proportional to weights between words, which in turn cannot be lower than 0 since they model term co-occurrences. Another property that must hold is probability unitarity 5, namely for each state the sum of all outgoing arcs must be equal to 1. However, in general

$$\sum_{i=1}^{k} a_{ji} + \sum_{i=1}^{M} b_{ji} \neq 1, \qquad \forall j = 1, ..., k$$

because a_{ij} and b_{ij} have been computed separately through formulas 41 and 42, ignoring this constraint. Therefore, a normalization step is required to transform weights into probabilities. Normalization is performed as follows:

$$a'_{ij} = \frac{a_{ij}}{\sum_{z=1}^{k} a_{iz} + \sum_{z=1}^{M} b_{iz}}, \qquad \forall i, j = 1, ..., k, \quad i \neq j \tag{43}$$

$$b'_{ij} = \frac{b_{ij}}{\sum_{z=1}^{k} a_{iz} + \sum_{z=1}^{M} b_{iz}}, \qquad \forall i = 1, ..., k, \quad \forall j = 1, ..., M \tag{44}$$

Now we are sure that probability unitarity property holds

$$\sum_{i=1}^{k} a'_{ji} + \sum_{i=1}^{M} b'_{ji} = 1, \qquad \forall j = 1, ..., k \tag{45}$$

and, at the same time, we have not changed the semantics of the model, keeping the same proportions between weights. As previously said, a Markov Chain is a stochastic process characterized by the so called Markov property, which says that the future state in the system evolution depends only on present state and does not depend on all past states. So, apart from basic probability theorems, we need proving that Markov property holds but, before doing this, let $S = \{s_1, ..., s_k\}$ be the set of Markov Chain states, where each s_i corresponds to a term t_i in the corpus. Let $X = \{x_1, ..., x_T\}$ be the set of states crossed during the system evolution imaging that T iterations occurred, where x_1 represents the first state crossed and x_T the last state crossed. According to these definitions, Markov property says:

$$P(S_{t+1} = x_{t+1} | S_1 = x_1, ..., S_t = x_t) =$$
$$= (S_{t+1} = x_{t+1} | S_t = x_t), \qquad 1 \leq t \leq T - 1 \tag{46}$$

Markov property surely holds in the proposed model, because the probability that system will transit to future state does not depend on all past states, but just on present state. Furthermore, since states represent words, in accordance with equations 43 and 44 this probability is always equal to a'_{ij} and b'_{ij} respectively, being s_i the current state and s_j the future state.

In conclusion, I used Markov Chain model in order to represent words co-occurrences in the considered corpus of documents. Furthermore, it is the first time that classes have been included in a Regular Markov Model to perform Document Sentiment Classification. The main advantages of relying on the soundness of this mathematical theory are:

- A mathematical theory guarantees to easily control the soundness of the entire approach.
- Markov Chains have been widely used in many scientific works (as shown in Chapter 2) and they have already proved being useful.

3.2 THE NEW PROPOSED MARKOV BASED ALGORITHM

After having understood the main reasons behind the utilization of Markov Chains, we are going to explain how to use them to design an algorithm for Document Sentiment Classification. Let us remind that the set of terms is $T = \{t_1, ..., t_k\}$ and the set of categories is $C = \{c_1, ..., c_M\}$. Now let us imagine we have a set of documents $D_{train} = \{d_1^{train}, ..., d_{N_s}^{train}\}$, whose labels are available, and a set of documents $D_{test} = \{d_1^{test}, ..., d_{N_t}^{test}\}$, whose labels are unknown. The goal is assigning to each document $d_i^{test} \in D_{test}$ a class $c_j \in C$. Without lack of generality, let us consider a cross-domain problem, that is, the set of documents D_{train} is about a certain topic and the set of documents D_{test} is about a different topic. Both topics have C as their set of possible labels (notice that, being in a Sentiment Classification context, this is feasible because labels represent polarities). As we have already seen in chapter 2, the main issue in cross-domain is that people could use different words to express opinions. Therefore, what we need is a sort of mapping between terms appearing in source domain and those in target domain. The idea I rely on is that, apart from domain specific words, there is a subset of common terms between the two domains. This is almost always true in Sentiment Classification problems, just think about words like *good*, *bad*, *optimum*, and so on, which are general, independent from the topic discussed. Consequently, these words could act as a bridge between domain specific terms.
Since the goal is classifying target documents, we require exploiting this bridge, in order to allow information about classes flowing from

domain specific terms belonging to the source to domain specific terms belonging to the target. How to realize this?
The answer is straightforward: through modeling term co-occurrences as explained in the previous section. In fact, if two domain specific words belonging to different domains are connected with a general (i.e. common) opinion word (see figure 6), they tend to assume the same orientation.

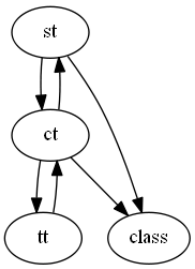

Fig. 6: *The figure shows the mapping between source and target by means of common terms. st and tt represent two domain-specific terms, belonging to source and target respectively. ct represents a common term between the two domains. We could notice that, even if tt is not linked with class, we know that it is related to ct, which in turn is linked with class. This is the way to walk in order to align different domains.*

Once having motivated how to align target with source domain, we could see how the new proposed Markov Chain algorithm works. The algorithm is composed of two parts:

1. the learning phase

2. the classification phase

3.2.1 *The learning phase*

The goal of the learning phase is building a model that can be subsequently used to classify documents in the classification phase. The model produced by the learning phase of the proposed algorithm is a Markov Chain, which could be represented as a Markov Chain transition matrix (MCTM), as pointed out in chapter 1. MCTM, whose basic structure is shown in table 6, is a $(k+M) \times (k+M)$ matrix having current states as rows and future states as columns. As we have stated in the previous section, each state corresponds to either a term or a class. Each entry of MCTM is computed differently depending on the nature of current and future states as described below.

Remind that $A = \{a_{ij}\}$ is the set of connection weights from term t_i to term t_j, as said by relation 41 that we could rewrite as

	$\mathbf{t_1, ..., t_k}$	$\mathbf{c_1, ..., c_M}$
$\mathbf{t_1, ..., t_k}$	$A^{'}$	$B^{'}$
$\mathbf{c_1, ..., c_M}$	E	F

Table 6: This table shows the structure of MCTM. $A^{'}$ represents the set of transition probabilities that, starting from a term, another term will be reached. Similarly, $B^{'}$ represents the set of transition probabilities that, starting from a term, a category will be reached. E represents the set of transition probabilities that, starting from a class, a term will be reached. F represents the set of transition probabilities that, starting from a class, another class will be reached.

$$a_{ij} = a_{ji} = \begin{cases} 0, & i = j \\ \sum\limits_{d \in D_{train} \cup D_{test}} w_{t_i}^d \cdot w_{t_j}^d, & i \neq j \end{cases}, \qquad \forall i, j = 1, ..., k \tag{47}$$

and that $B = \{b_{ij}\}$ is the set of connection weights from term t_i to class c_j, as stated by relation 42 that we could rewrite as

$$b_{ij} = \sum_{d \in D_{train}} w_{t_i}^d, \qquad c_d = c_j, \quad \forall i = 1, ..., k, \quad \forall j = 1, ..., M \tag{48}$$

because we only know class labels for documents belonging to the source set D_{train}. Then, the set of transition probabilities from term t_i to term t_j, namely $A^{'} = \{a_{ij}^{'}\}$, and the set of transition probabilities between term t_i and class c_j, namely $B^{'} = \{b_{ij}^{'}\}$, are represented by relations 43 and 44. On the other hand, regarding the set of transition probabilities from class c_i to term t_j, namely $E = \{e_{ij}\}$, and the set of transition probabilities from class c_i to class c_j, namely $F = \{f_{ij}\}$, the idea is that classes are absorbing states. Therefore, e_{ij} and f_{ij} are computed as follows:

$$e_{ij} = 0, \qquad \forall i = 1, ..., M \quad \forall j = 1, ..., k \tag{49}$$

$$f_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}, \qquad \forall i, j = 1, ..., M \tag{50}$$

In accordance with these definitions, table 7 shows how the example illustrated in figure 5 can be transposed in the MCTM.

The computational complexity of the learning phase is exactly the time required to build the MCTM, say $time(MCTM)$, which is:

$$time(MCTM) = time(A) + time(B) + time(A^{'} + B^{'}) + time(E) + time(F) \tag{51}$$

	$\mathbf{t_1}$	$\mathbf{t_2}$	$\mathbf{t_3}$	$\mathbf{t_4}$	$\mathbf{t_5}$	**positive**	**negative**
$\mathbf{t_1}$	0	$\frac{a_{12}}{den^{t_1}}$	$\frac{a_{13}}{den^{t_1}}$	$\frac{a_{14}}{den^{t_1}}$	$\frac{a_{15}}{den^{t_1}}$	$\frac{b_{11}}{den^{t_1}}$	$\frac{b_{12}}{den^{t_1}}$
$\mathbf{t_2}$	$\frac{a_{12}}{den^{t_2}}$	0	$\frac{a_{23}}{den^{t_2}}$	$\frac{a_{24}}{den^{t_2}}$	$\frac{a_{25}}{den^{t_2}}$	$\frac{b_{21}}{den^{t_2}}$	$\frac{b_{22}}{den^{t_2}}$
$\mathbf{t_3}$	$\frac{a_{13}}{den^{t_3}}$	$\frac{a_{23}}{den^{t_3}}$	0	$\frac{a_{34}}{den^{t_3}}$	$\frac{a_{35}}{den^{t_3}}$	$\frac{b_{31}}{den^{t_3}}$	$\frac{b_{32}}{den^{t_3}}$
$\mathbf{t_4}$	$\frac{a_{14}}{den^{t_4}}$	$\frac{a_{24}}{den^{t_4}}$	$\frac{a_{34}}{den^{t_4}}$	0	$\frac{a_{45}}{den^{t_4}}$	$\frac{b_{41}}{den^{t_4}}$	$\frac{b_{42}}{den^{t_4}}$
$\mathbf{t_5}$	$\frac{a_{15}}{den^{t_5}}$	$\frac{a_{25}}{den^{t_5}}$	$\frac{a_{35}}{den^{t_5}}$	$\frac{a_{45}}{den^{t_5}}$	0	$\frac{b_{51}}{den^{t_5}}$	$\frac{b_{52}}{den^{t_5}}$
positive	0	0	0	0	0	1	0
negative	0	0	0	0	0	0	1

Table 7: This table transposes the example in figure 5 to the MCTM. Notice that: den^{t_1}=$a_{12} + a_{13} + a_{14} + a_{15} + b_{11} + b_{12}$; den^{t_2}=$a_{12} + b_{21} + b_{22}$; den^{t_3}=$a_{13} + a_{35} + b_{31}$; den^{t_4}=$a_{14} + a_{45} + b_{42}$; den^{t_5}=$a_{15} + a_{35} + a_{45} + b_{51} + b_{52}$.

Thus, since time complexity depends on these factors, we have to estimate each of them. The only assumption we can do is that in general $k >> M$.

$$time(A) = O(\frac{k^2}{2} \times (N_s + N_t)) = O(k^2 \times (N_s + N_t))$$

$$time(B) = O(k \times M \times N_s) = O(k \times N_s)$$

$$time(A^{'} + B^{'}) = O(k \times (k + M) + k + M) = O((k + 1) \times (k + M)) = O(k^2)$$

$$time(E) = O(k^2)$$

$$time(F) = O(k \times M) = O(k)$$

Therefore, the computational complexity of the learning phase is:

$$time(MCTM) \simeq time(A) = O(k^2 \times (N_s + N_t)) \tag{52}$$

where I remind that k is the number of terms, N_s and N_t are the number of documents belonging to training (i.e. source in a cross-domain problem) and test (i.e. target in a cross domain problem) respectively.

3.2.2 *The classification phase*

The goal of the classification phase is categorizing documents D_{test} belonging to the target domain by means of the model built in the learning phase. The document representation I rely on is bag-of-words, which is essentially a term-document matrix where each document is

seen as a set (i.e. bag) of words. Each word has a specified weight, usually independent from its position inside the document. Hence, a document d to be classified can be considered as a vector of terms. In accordance with this consideration and with the model output of the learning phase, we could define d as a document having the same terms $t_1, ..., t_k$ of the corpus used to build the MCTM and the same class values. Obviously, all the terms actually in d will have a weight greater than 0, while all missing terms will have a weight equal to 0. Class values will be set to 0 as well because the document has to be classified and of course we could not know in advance its class label. Formally, let $d_t \in D_{test}$ be a document to be classified. According to the bag-of-words representation, it can be expressed as follows:

$$d_t = (w_{t_1}^{d_t}, ..., w_{t_k}^{d_t}, c_1, ..., c_M)$$

where $w_{t_i}^{d_t}$ is the weight of term t_i in document d_t and $c_i \in C$ is one of the available classes as previously defined. Those weights are prior measures, expressing just the initial term relevance in d_t. Thus we can see $w_{t_1}^{d_t}, ..., w_{t_k}^{d_t}$ as the probability distribution representing the initial state of the MCTM, while the prior probability that at the beginning a class is relevant for a test document (and consequently that a class can be the initial state) is trivially equal to 0 (i.e. $c_1 = ... = c_M = 0$). So, we initially hypothesize to be in many different states (i.e. every state t_i so that $w_{t_i}^{d_t} > 0$) at the same time. Then, simulating a single step inside the MCTM, we will obtain a posterior probability distribution not only over terms, but also over classes, because classes are admissible states as well. In such a way, estimating the posterior probability that d_t belongs to a certain class c_i, we could assign to d_t the most likely label $c_i \in C$. The posterior probability distribution after one step in the MCTM, starting from document d_t, is:

$$d_t^* = (w_{t_1}^{d_t^*}, ..., w_{t_k}^{d_t^*}, c_1^*, ..., c_M^*) = d_t \cdot MCTM \tag{53}$$

where the size of d_t is $1 \times (k + M)$ and the size of $MCTM$ is $(k + M) \times (k + M)$. As we have already mentioned before, the category that will be assigned to d_t is computed as follows:

$$c_{d_t} = \arg\max_{i \in C^*} c_i^* \tag{54}$$

where $C^* = \{c_1^*, ..., c_M^*\}$ is the posterior probability distribution over classes.

The computational complexity of the classification phase of the algorithm is:

$$time(CLASS) = time(MatProd) + time(Max)$$

where $time(MatProd)$ is the time required to compute the d_t^* values, whereas $time(Max)$ is that to compute the c_{d_t} values.

$$time(MatProd) = O((k + M)^2 \times N_t) = O(k^2 \times N_t)$$

$$time(Max) = O(M \times N_t)$$

Therefore, the computational complexity results:

$$time(CLASS) \simeq time(MatProd) = O(k^2 \times N_t) \tag{55}$$

where, again, k is the number of terms and N_t is the number of documents belonging to test (i.e. target in a cross-domain problem).
Finally, the computational complexity of the entire algorithm is given by

$$\begin{aligned} time(MC_{Algorithm}) = time(MCTM) + time(CLASS) \simeq \\ \simeq time(MCTM) = O(k^2 \times (N_s + N_t)) \end{aligned} \tag{56}$$

We would like to point out that the just described algorithm is absolutely general, because it is sound if we consider both a Single-Domain problem and a Cross-Domain problem. In fact, although this approach was designed to deal with Cross-Domain Sentiment Classification tasks, there is no need to change anything if training set and test set belong to the same domain (i.e. for Single-Domain Sentiment Classification). Obviously, Single-Domain DSC is a simpler task than Cross-Domain DSC, because in the former there are usually more common words between training set and test set than in the latter, belonging these two sets to the same domain. Another important feature of the algorithm is its language independence: there is no assumption about the nature of words, because the only thing we need to know is the weight of each term inside each document (i.e. we only need to represent each document as a vector of terms). Related to that just described, the last significant advantage is the simplicity of this approach, since the only parameter to be decided is the measure to be employed for the computation of term weights inside documents.

3.3 MARKOV BASED ALGORITHM: SOME VARIANTS

In this section we will see some possible variants of the general approach just introduced. I will call the original algorithm $MC_{Algorithm}$ for disambiguation purposes. The majority of the methods I will describe aim to expand documents. Explaining better, for every document d the weight $w^d_{t_i} > 0$ if and only if the term $t_i \in d$. However, it is possible that inside the corpus there are other terms $t_j \notin d$ that are semantically related to t_i (examples of semantic relationships between words are synonymy, antonymy, hyponymy, hypernymy, and so on). In this case, it would be meaningful expanding the initial probability distribution over words inside document d, by including semantically related terms as well. The variants that fulfil this goal I will propose in this work are:

- Document expansion by means of Markov Chain Stationary Distribution (MCSD)

- MCTM expansion by means of MCSD
- Document expansion by means of words distance

On the other hand, another modification slightly different from document expansion arises answering to the question: *Do connection weights always represent term co-occurrences*? Finally, the last variant I would like to discuss is a totally diverse approach to Cross-Domain Sentiment Classification. As previously said, one of the biggest issue of this task is finding a set of common words between source and target, that can guide the classification process. Nevertheless, it is not always simple to extract a general feature set. To overcome this limitation, my idea consists in using more than just one source domain when building the MCTM model. I will call this variant Multi-Source approach, written as MC_{MS}.

3.3.1 *Document expansion by means of Markov Chain Stationary Distribution*

We have already discussed the reason why document expansion could be useful. Now let us see how a document can be expanded according to the Markov Chain model. Remember that at the beginning a document is expressed as a set of weights and that executing a single step in the MCTM means simulating a state transition, hypothesizing to start from every state corresponding to a non-null weight. Consequently, the idea is to allow the simulation of T steps rather than just one, in order to spread the term relationships out, involving as many semantically related terms as possible.
Moreover, another consideration to be done is that initial term weights could not to reproduce the trend of the entire corpus, introducing classification issues. However, we know from Markov Chain theory that if a MC is ergodic, then it has a stationary distribution, which can be interpreted as a new term weighting (TW), a posterior probability distribution stating the relevance of each term. According to what Hoenkamp et al. said in [19], it is trivial to prove that a MC built on a bag-of-words representation is always ergodic. Therefore, we could think about substituting the initial term weights in each document with the ones returned by the computation of the Markov Chain Stationary Distribution (MCSD). Subsequently, the expansion process will be executed over the documents just modified.
Formally, the entire expansion process, performed between the learning phase and the classification phase, is as follows:

1. Compute the $MCSD = \{w_{t_1^{st}}, ..., w_{t_k^{st}}\}$. This is possible because when MC is ergodic, beyond the existence of MCSD, it is also guaranteed that it is the unique solution to $A^{'}x = x$, where $A^{'}$ is the same MCTM submatrix as defined in 43. Thus we could solve $(A^{'} - I)x = 0$, replacing the first row with the constraint

that says that the sum of x coordinates must be equal to 1, since x represents a probability distribution. Let v be a column vector of size $k \times 1$, where

$$v = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Then, the stationary distribution $MCSD$ can be computed as follows:

$$(MCTM^t - I) \cdot MCSD^t = v \Rightarrow$$

$$\Rightarrow MCSD = ((MCTM^t - I)^{-1} \cdot v)^t \tag{57}$$

where the exponent t states the transpose operator and $I \in (k \times k)$ is the identity matrix of order k.

2. For each document $d_t \in D_{test}$, $d_t = \{w_{t_1}, ..., w_{t_k}, 0_1, ..., 0_M\}$, substitute d_t with a new document $d_t^* = \{w_{t_1^*}, ..., w_{t_k^*}, 0_1, ..., 0_M\}$, where

$$w_{t_i^*} = \begin{cases} 0, & w_{t_i} = 0 \\ w_{t_i^{st}}, & w_{t_i} > 0 \end{cases}, \qquad \forall i = 1, ..., k \tag{58}$$

3. For each document $d_t \in D_{test}$, expand d_t by executing the product:

$$d_t^{exp} = d_t \cdot MCTM^T \tag{59}$$

where $MCTM^T$ means that T steps in the MCTM have to be executed.

Henceforth, this variant will be referred as $MC_{DocExp_{MCTM}}$.

3.3.2 *MCTM expansion by means of Markov Chain Stationary Distribution*

As we have seen, the MCSD can be interpreted as a new term weighting, a new probability distribution taking into account the entire corpus of documents involved in the Sentiment Classification task. Apart from documents to be classified, we could think about expanding training documents as well. The idea is the following:

1. Compute the MCSD as in 57.
2. For each document $d \in D_{train} \cup D_{test}$, expand d_t in the same way explained by relations 58 and 59.

3. Use the new documents to create an expanded MCTM, as in the learning phase of the standard $MC_{Algorithm}$.

The new MCTM models term co-occurrences as before, but taking into consideration also semantically related terms. In fact, once expanded, documents contain both original terms and semantically related ones. Obviously, the probability distribution over terms inside a document is different from the initial one due to the expansion step. Notice that this approach, to which we will refer as $MC_{MCTMExp}$, does not exclude $MC_{DocExp_{MCTM}}$. Therefore, we could combine these two variants, by expanding documents in order to both create a new MCTM and classify test documents executing a step in the MCTM itself.

3.3.3 *Document expansion by means of words distance*

A diverse approach to document expansion, referred as $MC_{DocExp_{wd}}$, relies on distance among words. The goal is finding the distance between the words inside a document and all other terms in the whole corpus, giving to the new encountered words a weight that decreases by augmenting the distance. To accomplish this task, a key question is: *How to compute the distance between two words*?
I would like to make you remember that each step in the MCTM simulates a state transition, starting from every MC state corresponding to the words inside a document. Consequently, as I have already pointed out, the probability distribution over words in a document changes after each step, so that after having performed the step, a term t whose weight was $w_t = 0$ could have a new weight $w_t^* > 0$. If this happens, it means that from the initial probability distribution t has been reached thanks to the last state transition (i.e. the step in which the weight of t has changed from w_t to w_t^*). Hence, we could define the distance as the number of steps required to hit a new term. Formally, let $dist_{t_i}^d$ be the distance of the term t_i from document d. Then, we can say that

$$dist_{t_i}^d = z_i, \qquad z_i \in \mathbb{N} \tag{60}$$

if $\exists T_i \in \mathbb{N}$ so that

$$w_{t_i} = \begin{cases} 0, & T_i < z_i \\ w_{t_i}^*, & T_i \geq z_i, \quad w_{t_i}^* > 0 \end{cases}$$

where T_i represents the number of iterations in the MCTM needed to reach t_i, whereas $w_{t_i}^*$ is the weight of t_i after T_i iterations.
Now that we know how to calculate the distance value for each word, we could design another variant of the $MC_{Algorithm}$ introducing a document expansion phase in this way:

1. Compute the MCSD, with $MCSD = \{w_{t_1^{st}}, ..., w_{t_k^{st}}\}$, as in 57.

2. For each document $d \in D_{train} \cup D_{test}$, with $d = \{w_{t_1}, ..., w_{t_k}, c_1, ..., c_M\}$, expand d in d^* in the following way:

$$d^* = \{w_{t_1^{st}}^{z_1+1}, ..., w_{t_k^{st}}^{z_k+1}, c_1, ..., c_M\} \tag{61}$$

where z_i is the distance of term t_i from document d, as indicated by equation 60. Notice that this relation allows achieving the established goal, namely the more terms are distant from the document the more their weight should be low. In fact, being MCSD a probability distribution, the following relation holds

$$0 \leq w_{t_i^{st}} \leq 1, \qquad \forall i = 1, ..., k$$

and, consequently,

$$\lim_{z_i \to \infty} w_{t_i^{st}}^{z_i+1} = 0$$

3. Use the expanded documents to create an expanded MCTM, as in the learning phase of the standard $MC_{Algorithm}$.

3.3.4 *Do connection weights always represent term co-occurrences?*

Until now, I have affirmed that Markov Chains are suitable to model term co-occurrences. According to this, every transition probability between two MC states is generally function of connection weights among terms, which in turn are usually the higher the more two words co-occur. In particular, in $MC_{Algorithm}$ the relationship between a term t_i and a class c_i depends on the weights t_i has in documents of class c_i and those weights are higher the more t_i occurs in those documents. This means that t_i has a certain polarity if it frequently appears in documents having the same polarity. In spite of being reasonable assumptions, another remark could be done. Imagine we had a technique capable to establish how much t_i really affects the classification process; for each term t_i a ranking would be computed and it could be exploited in $MC_{Algorithm}$. This is feasible because in literature there are plenty of supervised techniques that give terms scores proportional to their classification capabilities.
Formally, let $R = \{r_1, ..., r_k\}$ be the set of ranking returned by a supervised technique for each term in the corpus. Let $w_{t_i}^d$ be the original weight of a term t_i in a document d. Then, we could change $w_{t_i}^d$ in a weight $w_{t_i^*}^d$ computed as follows:

$$w_{t_i^*}^d = w_{t_i}^d \cdot r_i \tag{62}$$

Both the learning phase and the classification phase of $MC_{Algorithm}$ remain unaltered, but every weight $w_{t_i}^d$ will be substituted by $w_{t_i^*}^d$.

3.3.5 *Multi-Source approach*

The last variant we are going to discuss, referred as MC_{MS}, is not explicitly related to document expansion. The idea is similar to that of Bollegala et al. in [57], who proposed using more than just one source domain to improve the performance of Cross-Domain task. They relied on a sentiment sensitive thesaurus, where semantically related words are grouped, in order to perform a meaningful document expansion. Instead, the aim of MC_{MS} is just providing a bigger and heterogeneous feature set, which could help the mapping between domain specific words belonging to the source domain and those belonging to the target domain.
Due to this consideration, the main structure of $MC_{Algorithm}$ is unchanged, whilst the Multi-Source approach directly affects both the content of the MCTM and the way the classification phase is performed.

3.3.5.1 *The learning phase*

The structure of the learning phase is almost stable, except for the fact we need to take into account some source domains. For this purpose, let $DOM = \{dom_1, ..., dom_h\}$ be the set of the considered source domains. As a consequence, the training set becomes

$$D_{train} = \{D_{train^{dom_1}}, ..., D_{train^{dom_h}}\}$$

where $D_{train^{dom_i}} = \{d_{1^{dom_i}}, ..., d_{N_s^{dom_i}}\}$, and the set of classes become

$$C = \{C^{dom_1}, ..., C^{dom_h}\}$$

where $C^{dom_i} = \{c_{1^{dom_i}}, ..., c_{M^{dom_i}}\}$. So, the total number of classes is $M = M^{dom_1} + ... + M^{dom_h}$. Furthermore, remind that the MCTM can be considered being composed of 4 submatrices, namely $A^{'}$, $B^{'}$, E and F (see table 6). Under the above mentioned considerations, these submatrices still remain expressed as they were in the relations 43, 44, 49 and 50.

3.3.5.2 *The classification phase*

As for the learning phase, also the structure of the classification phase is unaltered. However, we could notice that we need dealing with more classes. This makes us understand that each domain (i.e. both sources and target) must have the same set of categories in order to perform classification in the right way, formally $C^{dom_1} = ... = C^{dom_h} = C^{test}$. Due to this reason, we have two basic viable ways to categorize test documents after having computed the posterior probability distribution over classes with equation 53. The first consists in choosing

the most likely class, as in relation 54. On the other hand, the second assigns the category c_{d_t} as follows:

$$c_{d_t} = \arg\max_{i \in C^*_{sum}} c^*_{sum_i} \tag{63}$$

where $c^*_{sum_i}$ is the sum of probabilities of class $c_{i^{dom_j}}$ in every source domain, that is

$$c^*_{sum_i} = \sum_{j=1}^{h} c_{i^{dom_j}}$$

and C^*_{sum} is the final probability distribution over classes that will be used to compute the most likely category, namely

$$C^*_{sum} = \{c^*_{sum_1}, ..., c^*_{sum_M}\}$$

Both alternatives are sound and require just a little modification of the classification phase.

4

FRAMEWORK AND IMPLEMENTATION

This chapter explains how the methods proposed in chapter 3 have been implemented to further carry analysis out. Firstly, the employed framework is introduced, giving an idea of its architectural design. Subsequently, some examples are shown about how the standard $MC_{Algorithm}$ and its variants have been realized within the framework.

4.1 THE CONCURRENT FRAMEWORK

Obviously, after having created a model representing the essential characteristics a system fulfilling a purpose should have, it is necessary to create a framework in order to realize it. Lots of tools are available for data analysis: for instance, *Weka* [1], which is a largely used Java [2] software, including a collection of machine learning algorithms, suitable for data pre-processing, classification, clustering, and so on. One important advantage is the fact that Weka makes a set of APIs available, so that it is simple both using the algorithms from the tool provided and calling them from Java code. Another relevant instrument for data analysis purposes is *R* [3], which is a functional programming language designed for statistical computing and graphics. In fact, R has a broad range of statistical packages, easily invokable to perform simple analysis. Moreover, also an IDE called *RStudio* [4] is available in order to facilitate R usage.
However, due to flexibility reason, I preferred not relying on neither *Weka* nor *RStudio*. More precisely, I chose developing my own framework in Java, providing some classes to make the interfacing with both Weka and R possible. In fact, Java is suitable on the one hand because Weka APIs are directly callable from code, since Weka is Java-based as well. On the other hand, also R scripts can be effortlessly called from Java code. Though, apart from considerations about Weka and R, I believe that the usage of a general purpose language is the best way to walk for the sake of developing new algorithms like those introduced in chapter 3. In fact, I did not need using well-known

1 http://www.cs.waikato.ac.nz/ml/weka/
2 https://www.java.com/
3 http://www.r-project.org/
4 http://www.rstudio.com/

algorithms, already proved useful, but just implementing novel techniques and conducting several tests. Parametrization, need for parallelism, algorithm performance evaluation are surely good reasons to choose realizing a new framework.

4.1.1 *Framework architecture*

Hereinafter, I will overlook some details, like for example how my framework interfaces with Weka and R, and I will focus on the main architectural components needed for performing analysis.
The framework architecture I designed can be divided into some main macro-parts:

- *Models*, containing the basic data modeling. Data are expressible in many different ways, so that any diverse dataset might have its own. Nevertheless, it is important to rely on an internal data representation, which is general, independent from datasets representation and, at the same time, comfortable for analysis.
- *Filters*, including any technique suitable for data pre-processing. Depending on the task we are required to accomplish, some filters could be demanding. In particular, in Text Mining, data are almost always long plain texts, which are not manageable at all. Therefore, some transformations could be necessary in order to map these texts to the internal model used for analysis.
- *Algorithms*, incorporating every proposed variant discussed in chapter 3. Each algorithm takes in input data expressed in the internal model defined by *Models*, executes the learning phase and the classification phase and returns a result expressed in the internal model as well.
- *Input-Output*, providing utilities useful to input-output interaction. In fact, texts have to be scanned and imported in framework before being analyzed. Likewise, after having performed experiments, the corresponding results might be printed in output, for example to the console or to some kind of file.
- *Concurrent Architecture*, incorporating the concurrent architecture necessary to perform large-scale tests. As already pointed out, algorithms have some parameters and, as a consequence, many initial configurations are required to be tested. Therefore, it is appropriate a support allowing the concurrent and parallel execution of many experiments at a time.

The whole framework has been implemented in Java, using Eclipse IDE [5] to ease the development. Every macro-part has been mapped

5 https://eclipse.org/

into a Java package. Below we can find a bit more detailed description of what kind of Java classes have been included in each macro-part. UML diagrams will be included just regarding the concurrent support, that is the most interesting macro-part of the built architecture.

4.1.1.1 *Models*

Models contains the basic data modeling and it is mapped into the package *models*. Essentially, internal representations of both document corpus and analysis results are required.

- *Couple*: a POJO representing a couple of values. This class is mainly used in order to model a term in the corpus, to which both an identifier and a term weighting are associated.

- *Category*: a POJO modeling available categories. This class stores information about the name used to refer to category and about its list of admissible values.

- *Document*: a POJO representing a document, namely any text, such as a comment, a review, and so on. A document is modeled as a list of *Couple* and is identified by a *Category*. Obviously, every utility method has to be provided, like for instance changing a term weight or searching if a term exists inside the document.

- *Dictionary*: a POJO used to store the word dictionary. Each word is trivially modeled as a java.lang.String.

- *BagOfWords*: a POJO modeling the whole document corpus, basically represented as a list of *Document*. Notice that the term bag-of-words usually indicates a term-document matrix, having a row for each word in the considered dictionary, a column for each document in the corpus and where an entry represents the weight of certain term into a particular document. On the other hand, the internal representation of a bag-of-words is a list of *Document*, where in turn each document is a list of *Couple* embodying term weights. This class also refers *Dictionary*.

- *CategoryMeasure*: a POJO representing the analysis results with respect to a certain *Category*. It includes methods to compute accuracy, precision, recall, f1-measure, and so on.

- *Result*: a POJO used to model the whole analysis results. It is constituted by a list of *CategoryMeasure*, typically long as the number of possible class values. This list is used to calculate the overall performance.

4.1.1.2 *Filters*

Filters includes every technique suitable for data pre-processing. It is implemented in two packages, namely *filters* and *filters.en*, respectively containing generic filters and specific filters appropriate for English. *filters* incorporates the following classes and interfaces:

- *ITextFilter*: a general interface, implemented by any generic filter, whose unique method is

  ```
  String[] filter(String textToFilter);
  ```

- *Tokenizer*: a filter useful to split a sentence into some words.
- *PunctuationRemover*: a filter useful to remove any punctuation mark, like commas, dots, semicolons, colons, question marks, exclamation marks, and so on.
- *NumberRemover*: a filter useful to remove any number found in text.
- *CaseFolder*: a filter converting every character to lower case.
- *StopWordsRemover*: a filter useful to discard every stop word. Stop words are terms that appear pervasively. Due to this reason, their presence does not help the classification process, because they tend to occur in any document regardless its category.
- *Lemmatizer*: a filter useful to lemmatize words, namely to determine the lemma for a given word. The lemma is a particular word representing a family of words, where each word in that family appears in an inflected form. For example, both *democratic* and *democratization* will be reduced to the same lemma, which is *democracy*. *Lemmatizer* is abstract, since stemming is heavily dependent on the specific language. Thus, it needs to be extended by other classes, containing specific stemming algorithms for each particular language.
- *Stemmer*: a filter useful to stem words to reduce inflected words. The reduction does not necessarily correspond to the lemma, because a morphological root is not demanded. In fact, related words just need to be reduced to the same stem. An example of stemming is *automate*, *automatic*, *automation*, which will be mapped into *automat*. However, stemming is similar to lemmatization, because it is a language dependent task and, consequently, requires relying on specific algorithms. Therefore, *Stemmer* is an abstract class as well.
- *TextFilters*: a class that allows applying filters one by one. *TextFilters* can be configured, including any number of *ITextFilter*, applicable in whatever order.

On the other hand, *filters.en* contains specific filters, suitable for English language. This package is mainly used to host classes implementing any possible extension to previously mentioned filters, useful to perform experiments. Primarily, *Stemmer* and *Lemmatizer* might be extended, because they are highly dependent on language.

4.1.1.3 *Algorithms*

Algorithms involves the basic $MC_{Algorithm}$ and its variants, proposed in chapter 3. All this is implemented in the package *algorithms*, containing the following classes:

- *MarkovChainAlgorithm*: a class implementing the basic $MC_{Algorithm}$, providing parametric methods for both the learning phase and the classification phase.

- *MCADocumentExpansion*: a class that extends *MarkovChainAlgorithm*, allowing the computation of the stationary distribution and overriding the classification phase of the basic $MC_{Algorithm}$ according to the description illustrated in 3.3.1.

- *MCAMCTMExpansion*: a class that extends *MarkovChainAlgorithm*, allowing the computation of the stationary distribution and overriding the learning phase of the basic $MC_{Algorithm}$ according to the description illustrated in 3.3.2.

- *MCADocAndMCTMExpansion*: a class that extends *MarkovChainAlgorithm*, allowing the computation of the stationary distribution and overriding both the learning phase and the classification phase of the basic $MC_{Algorithm}$, combining *MCADocumentExpansion* with *MCAMCTMExpansion*.

- *MCADistanceExpansion*: a class that extends *MarkovChainAlgorithm*, allowing the computation of the stationary distribution and overriding both the learning phase and the classification phase of the basic $MC_{Algorithm}$ according to the description illustrated in 3.3.3.

- *MCAWithRanking*: a class that extends *MarkovChainAlgorithm*, overriding both the learning phase and the classification phase of the basic $MC_{Algorithm}$, according to the description illustrated in 3.3.4.

- *MCAMultiSource*: a class that extends *MarkovChainAlgorithm*, overriding both the learning phase and the classification phase of the basic $MC_{Algorithm}$, according to the description illustrated in 3.3.5.

4.1.1.4 *Input-Output*

Input-Output is mapped into the package *io*, which includes every utility to be employed for input-output interaction.

- *DatasetReader*: a class containing methods to read a dataset in some formats, converting it to the internal bag-of-words model.
- *DatasetWriter*: a class containing methods to export a dataset, converting the internal bag-of-words model to some formats.
- *ResultReader*: a class including utilities to read the result of an experiment and store it in an instance of class *Result*.
- *ResultWriter*: a class including utilities to write an object of class *Result* to console, to a file, and so on.

4.1.1.5 *Concurrent Architecture*

Concurrent Architecture is the macro-part of the framework needed to perform large-scale analysis. Its structure is shown in the UML diagram [6] in figure 7, while every class has been implemented in the package *concurrent*.

- *ITask*: a general interface, which will be implemented by any task representing an analysis to be performed.
- *ITaskExecutor<T>*: an interface that extends *java.lang.Runnable* and that will be implemented by an active entity, called *TaskExecutor*.
- *TaskExecutor*: a class implementing ITaskExecutor<T>. Its instances are active objects whose mission is executing an *ITask*, namely carrying an analysis out.
- *ITaskExecutorFactory<T>*: an interface to be implemented for the sake of creating new instances of *TaskExecutor*.
- *TaskExecutorFactory*: a factory implementing *ITaskExecutorFactory<ITask>*. This class must create new instances of *TaskExecutor*.
- *FixedContinuousPool<T>*: a class representing a thread pool, that is a set of active entities (*TaskExecutor*), responsible to perform analysis. In particular, *FixedContinuousPool<T>* creates a certain number of *TaskExecutor*, sharing a queue containing pending tasks; then, it waits for their termination.

6 www.uml.org/

- *MultithreadedSupport*: a class used to handle the *FixedContinuousPool<T>*. More precisely, *MultithreadedSupport* has the responsibility to collect tasks in a task queue. Then, it instantiates a *FixedContinuousPool<ITask>*, sharing the task queue. Finally, it waits for the tasks termination. Notice that there is no assumption about the nature of tasks: they could also be heterogeneous, solely they need to implement *ITask*. Each diverse task is collected by *MultithreadedSupport* through a separate method; in other words, a method is required for any different kind of task to be executed.

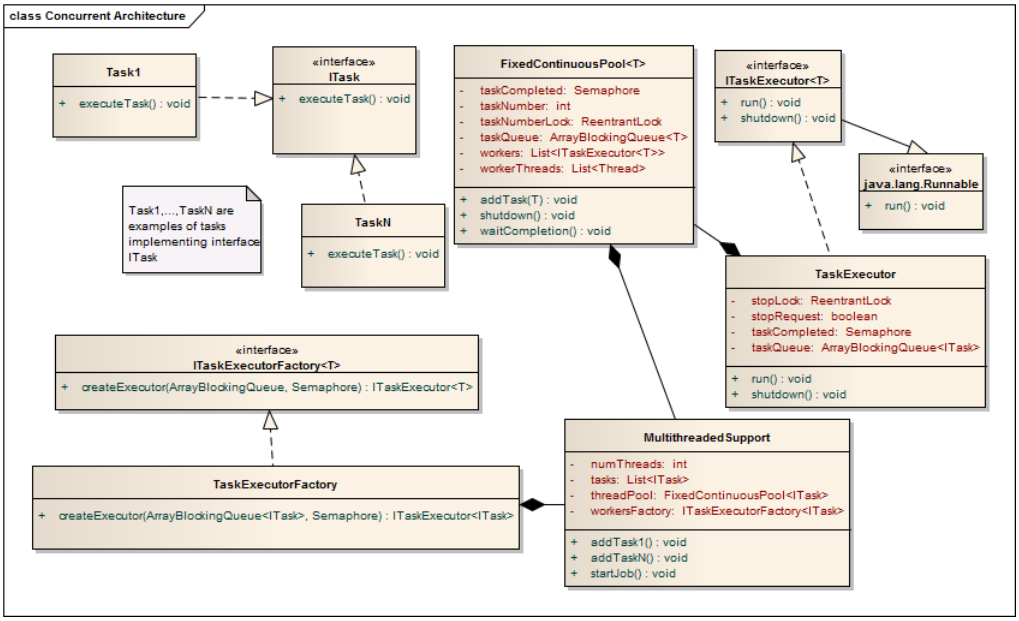

Fig. 7: *The figure shows the UML class diagram representing the concurrent architecture the framework relies on. Notice that ITask is a powerful abstraction, because the concurrent architecture just depends on it. In order to perform the analysis denominated Task1, just two things are required: the first is that Task1 needs to implement ITask, whereas the second is that a method (called for instance addTask1()) has to be created in MultithreadedSupport in order to instantiate Task1.*

4.2 MARKOV CHAIN ALGORITHM IMPLEMENTATION

So far we have seen the basic framework architecture, underlining the flexibility of the macro-part delegated to handle concurrency, essential in order to perform large-scale analysis. Now I would like to show you just an example about how an analysis is carried out and, since I have already illustrated the mechanism of the basic $MC_{Algorithm}$, I will describe how to conduct an experiment within the just presented

framework.
Let us imagine to have a tool to be exploited in order to launch an analysis over a dataset. After having loaded two labeled datasets by means of *DatasetReader*, one to be used as source domain (i.e. training set) and the other as target domain (i.e. test set), we could decide to execute $MC_{Algorithm}$ with default parameters (a detailed discussion about parameters calibration will be carried out in the following chapter). The macro-steps that will be performed are:

1. Data pre-processing
2. $MC_{Algorithm}$ learning phase
3. $MC_{Algorithm}$ classification phase
4. Results printing

The first thing that should be done by framework is data pre-processing. Plain text documents are filtered through *TextFilters*, which applies in order *Tokenizer*, *PunctuationRemover*, *NumberRemover*, *CaseFolder* and *StopWordsRemover*. Notice that a wordlist should be given in input to allow *StopWordsRemover* accomplishing its duty. Contrarily, a default English wordlist will be used. Once performed those tasks, my framework starts building two *BagOfWords*, one for source domain and another for target domain. First of all, terms appearing both in source and in target are merged into a unique list and utilized as *Dictionary*. After that, term weights can be computed somehow (for instance, applying some term weighting measures based on term frequency) and, then, *Document* can be created. Remind that I have defined a *Document* as a list of *Couple* {termID-weight}. Since both source and target rely on the same *Dictionary*, the term index can be used as *termID*. Finally, a *BagOfWords* can be built starting from the unique *Dictionary* and the list of *Document* just created.
Subsequent to pre-processing, both the learning phase and the classification phase are performed by $MC_{Algorithm}$. In the former, $MC_{Algorithm}$ takes in input the two *BagOdWords* representing source domain and target domain respectively and uses them to build the MCTM, relying on weights inside each *Document* and following the procedure described in 3.2.1. Afterwards, the MCTM is used in the latter to perform the classification phase according to what stated in 3.2.2. At the end, a *Result* is outputted and can be printed either to console or to a file by using *ResultWriter*.
Notice that, according to the concurrent architecture illustrated before, these four macro-steps are included in a task implementing *ITask* interface. This task is added to the pending task queue through a method defined in *MultithrededSupport*, which is also responsible to start the analysis process. Remember that if some parameters have to be tested, many instances of the same task can be created and a cer-

tain number of threads is fielded in order to execute every pending task in the task queue.

5

ANALYSIS AND RESULTS

This chapter shows the analysis that have been performed in order to test the methods presented in chapter 3, in both Single-Domain and Cross-Domain Sentiment Classification. Firstly, I will introduce the datasets used. Subsequently, analysis will be discussed in detail, explaining the reasons behind the execution of some tests and how modifications can affect results. Moreover, other experiments that have not been performed due to time constraints will be mentioned, illustrating how they are thought to be helpful to further improve performance.

5.1 DATA SOURCES

In this work three different data sources have been analyzed. The first is from Pan et al [55]. It contains a collection of *Amazon* reviews about Books (B), DVDs (D), Electronics (E) and Kitchen appliances (K). This collection has been used for testing performance in both papers I will cite in this dissertation to compare results, namely *Cross-Domain Sentiment Classification via Spectral Feature Alignment*, referred as *SFA*, a work presented by Pan et al. [55] in 2010 and *Automatically extracting polarity-bearing topics for cross-domain sentiment classification*, referred as *PBT*, introduced by He et al. [56] in 2011. The second and the third datasets have been provided by *Telematic Engineering Department at Charles III University of Madrid*. They both include comments retrieved from Facebook pages, respectively related to the energy company *Repsol* and the air company *Iberia*.

5.1.1 *Amazon dataset*

As previously mentioned, the first dataset is from Pan et al [55]. It contains a collection of *Amazon* reviews about Books (B), DVDs (D), Electronics (E) and Kitchen appliances (K). For each domain, we have a set of 2000 pre-classified reviews, 1000 having a positive polarity and 1000 having a negative polarity. Reviews are written in plain English, thus we need performing some data pre-processing steps before starting experiments, applying the filters illustrated in chapter 4. In particular, the following tasks have been done:

1. *Tokenization*, which involves the split of terms by whitespaces.

2. *Punctuation removal*, which consists in discarding commas, dots, semicolons, and so on.

3. *Number removal*, which lies in removing numbers.

4. *Case-folding*, where every token is reduced to lower case.

5. *Stop words removal*, which allows to eliminate words that appear pervasively. This task is highly relevant, because words occurring in every document regardless its polarity do not add useful information to the classification process. On the contrary, these terms could even perturb results; so it is a good practice to discard them. In literature there are a lot of pre-compiled stopword-lists (or stoplists), but they have to be used carefully. In fact, words as negative verbs like *couldn't, isn't, aren't*, and so on, are often included in those stoplists. However, while in Text Classification problems this is almost always irrelevant, in Document Sentiment Classification discarding them is a big issue, because they usually overturn sentence polarity. In accordance with this, stopwords like negative verbs have been maintained during the analysis.

Since in the Amazon collection there are four domains, we can form 12 cross-domain problems, such as $B \rightarrow D$, $B \rightarrow E$, $B \rightarrow K$, $D \rightarrow B$, $D \rightarrow E$, $D \rightarrow K$, $E \rightarrow B$, $E \rightarrow D$, $E \rightarrow K$, $K \rightarrow B$, $K \rightarrow D$ and $K \rightarrow E$. Due to time constraints, I excluded from the analysis the reviews about Kitchen appliances, so I just focused on 6 cross-domain problems, namely $B \rightarrow D$, $D \rightarrow B$, $B \rightarrow E$, $E \rightarrow B$, $D \rightarrow E$, $E \rightarrow D$.

5.1.2 *Iberia and Repsol datasets*

The second and the third analyzed data sources have been provided by *Telematic Engineering Department at Charles III University of Madrid*. *Iberia* and *Repsol* contain comments from the official Facebook pages of the Spanish air company and of the Spanish energy company respectively. Iberia (I) is composed of 1693 comments, 287 having positive polarity, 516 having negative polarity and 890 having neutral polarity. Instead, Repsol (R) comprises 1001 comments, of which 180 positive, 236 negative and 585 neutral. The most relevant difference with respect to the Amazon datasets is that both these data sources are written in Spanish. For every experiment performed, I did the same pre-processing tasks executed for the Amazon datasets analysis, with the only obvious difference that in this case we need Spanish stoplists.

5.2 PLANNED TESTS

Tests have been split into several parts, each one described in a different section in this chapter as follows:

1. At the beginning, the basic algorithm parametrization was tested over Amazon datasets, taking into account the problem of assigning initial term weights in documents. In this phase, I will also show you experiments relying on document expansion, in accordance with the variants already introduced in chapter 3.

2. In the second part, a detailed analysis looking for the best set of features $t_1, ..., t_k$ to be used in all methods will be shown. For this purpose, another aforementioned variant of $MC_{Algorithm}$, called $MC_{Ranking}$, will be presented. This study about dictionary, orthogonal to $MC_{Algorithm}$, is necessary to enhance the Sentiment Classification task performance.

3. Then, I will present experiments over another discussed variant, called Multi-Source approach, which relies on a completely different idea of what a cross-domain problem is.

4. Subsequently, I would like to show you some single-domain analysis, where my algorithm reaches promising results in comparison with other approaches.

5. The last section will briefly illustrate tests and results over Iberia and Repsol datasets.

Every test over Amazon datasets has been performed by splitting the 2000 instances into 1600 (80%) to be used for training (i.e. when the dataset is employed as source domain) and 400 (20%) to be used for test (i.e. when the dataset is employed as target domain). This split percentage is the same utilized in the aforementioned papers and, as a consequence, it allows comparing results with the state of the art. Unless otherwise stated, the metric used to establish how much a result is relevant is classification *accuracy*, which should be theoretically computed for every different category. However, in a 2-class problem where categories are c_1 and c_2 accuracy is acc_{tot}=acc_{c_1}=acc_{c_2}. Therefore, we simply indicate it as acc, computed as follows:

$$acc = \frac{tp + tn}{tp + tn + fp + fn} \tag{64}$$

where tp is the number of true positives, tn is the number of true negatives, fp is the number of false positives, fn is the number of false negatives. A better understanding of what tp, tn, fp, fn are is given in table 8.

	Assigned category	Real category
tp	c_i	c_i
tn	c_j	c_j
fp	c_i	c_j
fn	c_j	c_i

Table 8: This table explains the meaning of true positives (tp), true negatives (tn), false positives (fp) and false negatives (fn). tp, tr, fp and fn have to be related to a particular class. c_i is the class to which we are referring, whereas c_j stands for any other category.

5.3 TESTING THE BASIC ALGORITHM PARAMETRIZATION

In this section, some tests involving the basic parameters of the algorithm will be performed. As we said while we were describing $MC_{Algorithm}$, the most relevant parameter is how to assign term weights $w^d_{t_i}$ useful to build the MCTM. Nevertheless, other considerations could affect performance as well, like for example extending the preprocessing phase by applying well-known techniques like stemming or lemmatization. Finally, also experiments regarding $MC_{DocExp_{MCTM}}$, $MC_{MCTMExp}$ and $MC_{DocExp_{wd}}$ will be discussed here. Explaining better, this phase of tests aims to tune the basic $MC_{Algorithm}$, taking into account also little variations like document expansion. Afterwards, the algorithm will be compared with both SFA and PBT.

5.3.1 *Term weighting*

Term weighting consists in assigning scores to terms so that the most relevant terms must have higher scores and the least relevant terms must have lower scores. There exists a lot of well-known techniques that have been studied over the years. For this analysis, I compared four term weighting (TW) techniques, namely TF_{rel}, Log_{1+TF}, $Normal_{MI}$ and $Normal_{OR}$, the last two of which were introduced by Deng et al. in [9]. Formally, let t_i be a term inside a document d. Let w_{t_i} be the weight of t_i. Let TF_{t_i} be the term frequency of t_i inside d (i.e. the number of times t_i occurs in d). Then, the first two TW measures are defined as follows:

$$TF_{rel}(t_i) = w_{t_i} = \frac{TF_{t_i}}{\sum_{j=1}^{k} TF_{t_j}} \tag{65}$$

$$Log_{1+TF}(t_i) = w_{t_i} = \log(1 + TF_{t_i}) \tag{66}$$

Instead, in both $Normal_{MI}$ and $Normal_{OR}$, w_{t_i} is computed as:

$$w_{t_i} = ITD(t_i, d) \cdot ITS(t_i)$$

where ITD measures the importance of t_i inside d and ITS measures the importance of t_i in expressing sentiment. ITD is calculated as

$$ITD(t_i, d) = 0.5 + \frac{0.5 \cdot t_i}{\max_z t_z}, \qquad t_z \in d$$

and ITS is calculated as

$$ITS(t_i) = max\{Measure(t_i, D^{pos}_{train}), Measure(t_i, D^{neg}_{train})\}$$

where in turn *Measure* is Mutual Information (MI) in the case of $Normal_{MI}$ and Odds Ratio (OR) in the case of $Normal_{OR}$. Moreover, let $n^c_{t_i}$ be the number of documents belonging to class c that contain t_i, and let $\overline{n^c_{t_i}}$ be the number of documents belonging to class c that do not contain t_i. In accordance with this, MI is computed as

$$MI(t_i, D^c_{train}) = \log \frac{n^c_{t_i} \cdot (|D^{pos}_{train}| + |D^{neg}_{train}|)}{(n^{pos}_{t_i} + n^{neg}_{t_i}) \cdot |D^c_{train}|} \tag{67}$$

and OR is computed as

$$OR(t_i, D^c_{train}) = \log \frac{n^c_{t_i} \cdot (|D^{pos}_{train}| + |D^{neg}_{train}| - |D^c_{train}| - \overline{n^c_{t_i}})}{(|D^c_{train}| - n^c_{t_i}) \cdot \overline{n^c_{t_i}}} \tag{68}$$

The dictionary of features used for this analysis is

$$Dict = \{t_k | DF_{t_k} \geq 50\}$$

where DF_{t_k} is the Document Frequency of the term t_k, defined as follows:

$$DF_{t_k} = \sum_{d_{t_k} \in D_{train} \cup D_{test}} 1$$

where d_{t_k} is a document containing t_k. This parameter is not discussed here, because the whole following section is about analysis involving the dictionary of features calibration. Similarly, for this experiment also the Porter stemmer [58] was used, but it will be illustrated later.

The results of the utilization of these four measures into the standard $MC_{Algorithm}$ have been reported in table 9, where we could see that TF_{rel} and Log_{1+TF} achieve comparable performance. However, since TF_{rel} is better than Log_{1+TF} on average, hereinafter we will always use that TW measure in the following experiments. The reason why both $Normal_{MI}$ and $Normal_{OR}$ do not perform well has to be found into their complexity. In fact, they already model term relationships and, moreover, they do this in a supervised way, because both MI and OR take into account categories. This could be a problem, because $MC_{Algorithm}$ itself is designed to accomplish this task. So, in spite of being proved useful in [9], $Normal_{MI}$ and $Normal_{OR}$ do not fit well with $MC_{Algorithm}$.

	$\mathbf{TF_{rel}}$	$\mathbf{Log_{1+TF}}$	$\mathbf{Normal_{MI}}$	$\mathbf{Normal_{OR}}$
B → D	79.93%	75.08%	49.95%	49.95%
D → B	78.58%	78.43%	50.00%	50.00%
B → E	71.99%	69.08%	49.97%	49.97%
E → B	68.23%	68.93%	50.03%	50.03%
D → E	72.64%	75.99%	49.97%	49.97%
E → D	72.84%	71.99%	52.63%	51.48%
Average	74.04%	73.25%	50.43%	50.23%

Table 9: This table shows the accuracy of $MC_{Algorithm}$, by using *Porter stemmer*, $DF_{min} = 50$ and changing the way term weights have been computed.

5.3.2 *Stemming vs lemmatization*

Another important consideration is related to the kind of features used. Linguistic processes like stemming and lemmatization have been proved helpful in improving performance of Text Mining algorithms. Hence, I compared these two widely used techniques, repeating the previous experiment. Concerning stemming, I made use of an implementation [1] of the well-known *Porter stemmer* [58]. On the other hand, for lemmatization I relied on the online tool *Adorner* [2]. While stemming and lemmatization are general techniques, the implementations used are just suitable for English.
The tests I performed are available in table 10, where we could see that, to this end, stemming is more useful in comparison with lemmatization. The motivation behind this outcome is quite simple, just think that stemming assembles many related words into the same stem, usually more than lemmatization. Since it is likely that those clustered words have the same polarity, assembling them is a good way to walk in order to extract their semantics. As a consequence of this analysis, subsequent experiments will rely on stemming.

5.3.3 *Document and MCTM expansion*

In chapter 3 I introduced some variants of the standard $MC_{Algorithm}$, such as for instance $MC_{DocExp_{MCTM}}$ and MC_{MCTM}. I would like to remind that the former involves just test documents expansion during the classification phase of the algorithm, whilst in the latter both training set and test set are expanded, for the sake of building an augmented MCTM during the learning phase. Consequently, notice that $MC_{DocExp_{MCTM}}$ and MC_{MCTM} can be combined, since they involve two different phases of the algorithm. Any possible combi-

1 http://tartarus.org/martin/PorterStemmer/
2 http://devadorner.northwestern.edu/maserver/

	Stemming	**Lemmatization**
B → D	79.93%	77.24%
D → B	78.58%	76.51%
B → E	71.99%	71.82%
E → B	68.23%	67.02%
D → E	72.64%	71.88%
E → D	72.84%	72.52%
Average	74.04%	72.83%

Table 10: This table shows the accuracy of $MC_{Algorithm}$, by using $DF_{min} = 50$, $TW = TF_{rel}$ and comparing *Porter stemmer* with *Adorner lemmatizer* for English language.

	$\mathbf{MC_{Algorithm}}$	$\mathbf{MC_{DocExp_{MCTM}}}$	$\mathbf{MC_{MCTM}}$	$\mathbf{MC_{DocExp_{MCTM}} \cup MC_{MCTM}}$
B → D	79.93%	76.68%	78.48%	76.18%
D → B	78.58%	72.12%	78.98%	70.82%
B → E	71.99%	69.48%	70.54%	69.38%
E → B	68.23%	67.13%	67.93%	66.33%
D → E	72.64%	71.39%	72.19%	71.24%
E → D	72.84%	69.33%	72.29%	69.48%
Average	74.04%	71.02%	73.40%	70.57%

Table 11: This table shows the accuracy of $MC_{Algorithm}$ and some of its variants, namely $MC_{DocExp_{MCTM}}$ and MC_{MCTM}, by using *Porter stemmer*, $DF_{min} = 50$, $TW = TF_{rel}$ and $T = 4$.

nation should be investigated. To perform those tests, whose results have been reported in table 11, I set T=4 without calibrating the parameter, because I thought this could be a meaningful number of iterations for document expansion. The outcome of this investigation suggests that the simplest method, i.e. the standard $MC_{Algorithm}$ without document expansion, is the best one. This is not surprising in my opinion and the motivation is closely related to what I said regarding the poor performance of term weighting measures like $Normal_{MI}$ and $Normal_{OR}$. In fact, document expansion deals with the idea of extracting relationships between semantically related words, just like those TW measures do trying to find connections between terms and categories. But once again, term-term and term-class relationships have been already modeled by $MC_{Algorithm}$, which has proved to be negatively affected by the attempt of improving this capability.

5.3.4 *Document expansion by means of words distance*

The previous tests suggest that document expansion could not to be an useful approach in order to improve $MC_{Algorithm}$ performance.

	$\mathbf{MC_{Algorithm}}$	$\mathbf{MC_{DocExp_{wd}}}$
B → D	79.93%	75.43%
D → B	78.58%	77.07%
B → E	71.99%	67.53%
E → B	68.23%	67.23%
D → E	72.64%	73.94%
E → D	72.84%	72.49%
Average	74.04%	72.28%

Table 12: This table shows the comparison between the standard $MC_{Algorithm}$ and one of its variants designed for document expansion, that is $MC_{DocExp_{wd}}$. Other parameters used for this analysis are *Porter stemmer*, $DF_{min} = 50$, $TW = TF_{rel}$ and $T = 4$. The results display the accuracy reached by the employed algorithm.

Nevertheless, I believe that the idea behind document expansion by means of words distance (illustrated in chapter 3) is interesting and, as a consequence, can be worthy to assay it. I would like to remind that the goal is finding the distance between the words inside a document and all other terms in the whole corpus, giving to the new encountered words a weight that decreases by augmenting the distance. The main difference between this expansion method and the previous ones is that, in this case, terms initially in the document have higher weights than semantically related words. Of course, expanding document by distance is meaningless if we do not create a second MCTM like in MC_{MCTM}. The reason why we need to do this is that, in this expansion process, initial term weights are substituted with those of stationary distribution and, therefore, the newer ones model a posterior probability distribution while the elders represent a prior probability distribution. The outcome in table 12 shows that results are comparable regarding certain couples source-target, whereas this approach is worse than the standard $MC_{Algorithm}$ in other cases and on average. Again, this confirms that document expansion by adding semantically related terms cannot help the original Markov Chain based algorithm I propose in this dissertation.

5.3.5 *First comparison: $MC_{Algorithm}$ vs SFA and PBT*

After these analyses we could make a comparison between the current best version of the $MC_{Algorithm}$ and the state of the art. Remind that the current best version of the $MC_{Algorithm}$ has the following parameters:

- *Porter stemmer* utilization, apart from the standard pre-processing tasks

	$\mathbf{MC_{Algorithm}}$	**SFA**	**PBT**
B → D	79.93%	81.50%	81.00%
D → B	78.58%	78.00%	79.00%
B → E	71.99%	72.50%	78.00%
E → B	68.23%	75.00%	73.50%
D → E	72.64%	77.00%	79.00%
E → D	72.84%	77.50%	76.00%
Average	74.04%	76.92%	77.75%

Table 13: This table shows the comparison of $MC_{Algorithm}$ with both SFA and PBT. The parameters used for $MC_{Algorithm}$ are *Porter stemmer*, $DF_{min} = 50$, $TW = TF_{rel}$. Each entry represents the classification accuracy reached by an algorithm on a certain couple source-target.

- $TW = TF_{rel}$
- $DF_{min} = 50$

In literature, the two best working methods for Document Sentiment Classification are *Cross-Domain Sentiment Classification via Spectral Feature Alignment*, written by Pan et al. [55], and *Automatically extracting polarity-bearing topics for cross-domain sentiment classification*, introduced by He et al. [56]. As previously stated, I will refer to these works as *SFA* and *PBT* respectively. Furthermore, remind that both in those papers and in my analysis, algorithms performance have been tested over the *Amazon* datasets by using 1600 instances for source domain and 400 for target domain, due to comparison purposes. The comparison is shown in table 13, where we could see that accuracy values are comparable regarding $B \rightarrow D$ and $D \rightarrow B$. Instead, $MC_{Algorithm}$ is outperformed by both *SFA* and *PBT* in all other source-target couples (apart from $B \rightarrow E$ in comparison with *SFA*).

5.4 ANALYSIS OF THE WORDS DICTIONARY

In the last section, we have seen that $MC_{Algorithm}$ performance is not as good as those of *SFA* and *PBT*. Furthermore, variants incorporating document expansion cannot help improving my method. I have already pointed out that accuracy is comparable regarding $B \rightarrow D$ and $D \rightarrow B$, whereas performance is very poor every time Electronics is involved. We could think that this outcome is related to the nature of terms in reviews, in accordance with the fact that Books and DVDs are more similar than either Books and Electronics on the one hand or DVDs and Electronics on the other hand. In fact, while in the first couple it is likely that words used to connote positive and negative sentiments are alike, the same does not hold in general if considering

	$\mathbf{MC_{Algorithm}}$	$\mathbf{MC_{CommFeat}}$
B → D	79.93%	79.50%
D → B	78.58%	79.00%
B → E	71.99%	68.75%
E → B	68.23%	68.75%
D → E	72.64%	72.43%
E → D	72.84%	70.25%
Average	74.04%	73.11%

Table 14: This table shows the comparison between $MC_{Algorithm}$ and $MC_{CommFeat}$. The parameters of the former are *Porter stemmer*, $TW = TF_{rel}$ and $DF_{min} = 50$, while in the latter the last parameter becomes $DF_{min} = 25$ and a constraint is added, namely only common words between source and target must be in dictionary.

a review of an electronic product. This consideration could imply a greater difficulty in $MC_{Algorithm}$ while aligning polarity words belonging to completely different domains. Therefore, a targeted analysis of the dictionary is necessary, because selecting the best feature set is the key to enhance classification accuracy. So in this section we will look at the analysis performed in order to choose the best dictionary of features, which should help increasing classification performance.

5.4.1 *Testing common and domain specific features*

The first analysis performed consists in keeping just common words between source and target. The idea behind this experiment is to bypass the inter-domains alignment problem. Since all domain specific features have been discarded, to avoid considering just few terms, I set $DF_{min} = 25$ instead of $DF_{min} = 50$ in this case. In table 14 we could see the comparison between the standard $MC_{Algorithm}$ and the just described variant, referred as $MC_{CommFeat}$. Contrary to what I expected, performance does not improve, even if the number of selected features is between 800 and 900 in any source-target couple. Apparently, there are some domain specific terms that could ease sentiment classification. Therefore, I executed another analysis, by selecting the most frequent 2000 common features, 1000 source specific features and 1000 target specific features. However, the results reported in table 15, where MC_{4KFeat} indicates this new variant, say that neither this attempt is useful to enhance accuracy.

	$\mathbf{MC_{Algorithm}}$	$\mathbf{MC_{4KFeat}}$
B → D	79.93%	73.68%
D → B	78.58%	77.69%
B → E	71.99%	69.00%
E → B	68.23%	68.50%
D → E	72.64%	69.00%
E → D	72.84%	69.75%
Average	74.04%	71.27%

Table 15: This table shows the comparison between $MC_{Algorithm}$ and MC_{4KFeat}. The parameters of the former are *Porter stemmer*, $TW = TF_{rel}$ and $DF_{min} = 50$, while in the latter the last parameter is substituted by introducing the constraint that the most frequent 2000 common features, 1000 source specific features and 1000 target specific features must be kept.

5.4.2 *Testing supervised term weighting techniques*

So far we have just tried performing feature selection (FS) by means of an unsupervised technique, that is Document Frequency. On the other hand, plenty of supervised measures have been developed by researchers during the years. Before I showed the use of supervised term weighting techniques just for the sake of assigning weights to terms. However, some of those techniques might be employed to accomplish feature selection, keeping just terms closely connected with classes. In particular, for this purpose I evaluated the most relevant terms according to the score given by Information Gain (IG), Gain Ratio (GR), Chi-Squared ($CHI2$) and a variant of Relevance Frequency (RF_{var}). I defined those measures as a function of a, b, c and d, like it was suggested by Lan et al. in [59], where:

- a is the number of documents in the positive category which contain this term
- b is the number of documents in the positive category which do not contain this term
- c is the number of documents in the negative category which contain this term
- d is the number of documents in the negative category which do not contain this term

In accordance with this definition, if N is the number of document in the collection, the aforementioned measures can be expressed in the following way:

$$IG = \frac{a}{N} \cdot \log \frac{a \cdot N}{(a+c) \cdot (a+b)} + \frac{b}{N} \cdot \log \frac{b \cdot N}{(b+d) \cdot (a+b)} + \\ + \frac{c}{N} \cdot \log \frac{c \cdot N}{(a+c) \cdot (c+d)} + \frac{d}{N} \cdot \log \frac{d \cdot N}{(b+d) \cdot (c+d)} \tag{69}$$

$$GR = \frac{IG}{-\frac{(a+b)}{N} \cdot \log \frac{(a+b)}{N} - \frac{(c+d)}{N} \cdot \log \frac{(c+d)}{N}} \tag{70}$$

$$CHI2 = N \cdot \frac{(a \cdot d - b \cdot c)^2}{(a+c) \cdot (b+d) \cdot (a+b) \cdot (c+d)} \tag{71}$$

$$RF_{var} = \log(2 + \frac{a}{c}) \cdot \log(2 + \frac{c}{a}) \tag{72}$$

5.4.2.1 *An interesting remark*

Obviously, supervised methods have to be used only on the training set, because we could not know in advance test set labels. Instead, if this was feasible, we could select the perfect (i.e. ideal) feature set, which would enhance performance as much as possible. Due to this reason, the accuracy reachable with $MC_{Algorithm}$ considering as dictionary the perfect feature set can be seen as an upper bound to the classification accuracy achievable with my algorithm. Thus, I analyzed this ideal case, assuming that categories of documents belonging to the test set are known and trying to extract the best feature set through applying a supervised technique. For this experiment I used just IG, keeping the best 250 terms according to the IG score, then I performed sentiment classification through $MC_{Algorithm}$. In order to validate results, 10 random partitions containing 1600 documents for training set and 400 documents for test set have been chosen. The average output of this analysis, referred as $MC^{UB}_{Algorithm}$, is reported in table 16, along with the previous best experiment ($MC_{Algorithm}$), SFA and PBT. This trial states that, in the ideal case, my algorithm would achieve an accuracy $5 - 6\%$ higher than the state of the art methods. Further tests I did, which have not been reported for time and space reasons, reveal that on average 98% of the 250 features selected belong to the training set and, as a consequence, could be retrieved in a supervised way. Moreover, discarding the remaining 2% of features, non-retrievable because we could not know in advance test set labels, and executing the analysis again, accuracy does not change significantly. This unequivocally proves that $MC_{Algorithm}$ just needs a good feature selection process in order to outperform both SFA and PBT.

	$\mathbf{MC_{Algorithm}}$	$\mathbf{MC^{UB}_{Algorithm}}$	**SFA**	**PBT**
B → D	79.93%	86.03%	81.50%	81.00%
D → B	78.58%	86.30%	78.00%	79.00%
B → E	71.99%	83.44%	72.50%	78.00%
E → B	68.23%	78.17%	75.00%	73.50%
D → E	72.64%	83.07%	77.00%	79.00%
E → D	72.84%	80.17%	77.50%	76.00%
Average	74.04%	82.86%	76.92%	77.75%

Table 16: This table shows the accuracy of $MC_{Algorithm}$ in comparison with its upper bound $MC^{UB}_{Algorithm}$ and with the state of the art. $MC^{UB}_{Algorithm}$ have been computed by keeping the best 250 terms according to the IG score and applying the standard $MC_{Algorithm}$. $MC^{UB}_{Algorithm}$ results have been averaged on 10 random partitions containing 1600 documents for training set and 400 documents for test set.

5.4.2.2 *Testing IG, GR, CHI2 and RF_{var}*

Hereinafter, due to the previous consideration about the potential capability of my Markov Chain based algorithm, I will try finding a FS method able to extract the needed feature set. At the beginning, I basically choose features from the training set based on supervised measures, possibly adding other different features by selecting from the test set all terms having a certain DF_{min} (evaluated on the test set itself). Many parameters have been taken into account, such as the measure used to assign a score to features in source domain (M^{source}), the number of features in source domain to be retrieved (NF^{source}) and the minimum Document Frequency value used to select features from target domain (DF^{target}_{min}). The parameter M^{source} could be embodied by various supervised techniques, such as IG, GR, $CHI2$, RF_{var}, possibly multiplied by $LogDF_{min}$, with

$$LogDF_{min} = \log(1 + DF_{min})$$

where in turn DF_{min} could be computed over the training set (DF^{s}_{min}), over the test set (DF^{t}_{min}) or over their union (DF^{u}_{min}). Similarly, NF^{source} could be equal to 250, 500, 750, 1000 or ALL, where ALL stands for every feature belonging to the source domain having a score higher than 0. Finally, DF^{target}_{min} could be 5, 10 or INF, where INF means that no feature must be added different from those extracted from source domain.
Every possible combination of parameters was tested but, due to space limit, only the best configurations have been reported in table 17, where each result is named in the same way of the feature selection method applied. All names have the form M^{source}-NF^{source}-DF^{target}_{min}. Tests indicate an average enhancement of performance in

	$\mathbf{MC_{Algorithm}}$	$\mathbf{IG\_LogDF^{s}_{min}}$-250-INF	$\mathbf{CHI2\_LogDF^{s}_{min}}$-250-INF	SFA	PBT
B → D	79.93%	78.52%	78.52%	81.50%	81.00%
D → B	78.58%	78.89%	80.35%	78.00%	79.00%
B → E	71.99%	72.21%	72.09%	72.50%	78.00%
E → B	68.23%	72.56%	71.79%	75.00%	73.50%
D → E	72.64%	74.74%	75.52%	77.00%	79.00%
E → D	72.84%	73.67%	72.73%	77.50%	76.00%
Average	74.04%	75.10%	75.17%	76.92%	77.75%

Table 17: This table shows the accuracy of $MC_{Algorithm}$ to which a feature selection method was previously applied. In $IG\_LogDF^{s}_{min}$-250-INF, scores have been computed by means of $IG \cdot LogDF^{s}_{min}$, keeping the best 250 features belonging to the source, while no feature belonging just to the target has been maintained. Similarly in $CHI2\_LogDF^{s}_{min}$-250-INF, with the only exception that scores have been computed by means of $CHI2 \cdot LogDF^{s}_{min}$.

comparison with the basic $MC_{Algorithm}$, where feature selection was performed just by considering $DF_{min} = 50$. Nevertheless, the improvement is not enough to bridge the gap with the literature.

5.4.3 *Testing opinion words*

Another attempt we could do involves the utilization of opinion wordlists, namely lists of general words that are known to have a certain polarity. The English wordlist used in this dissertation, proposed by Liu in [60], contains 2003 positive words and 4780 negative words. I tested these opinion words (OW) both using just them as feature and combining them with the previous mentioned FS methods. In the latter case, two ways of blending OW and FS have been considered, namely intersection and union, but the second one gives better performance. In table 18 and in figure 8, we could find various tests: one where the only features selected are OW, referred as OW, and others where the union between OW and FS has been used to build the dictionary. Alike the preceding tests, these methods are referred as M^{source}-NF^{source}-DF^{target}_{min}-un (notice that the suffix un means that a union between the two mentioned FS techniques has been used). The performance of $MC_{Algorithm}$ using just OW as features demonstrates their utility. However, many opinion words were being retrieved by FS methods as well. It is for this reason that accuracy does not increase enough to outperform the state of the art.

	OW	**CHI2_LogDF$^{s}_{min}$-500-INF-un**	**GR-250-INF-un**	**GR-500-INF-un**	**SFA**	**PBT**
B → D	79.20%	79.75%	81.00%	84.00%	81.50%	81.00%
D → B	79.15%	80.75%	80.75%	80.00%	78.00%	79.00%
B → E	73.93%	73.18%	70.93%	71.43%	72.50%	78.00%
E → B	70.10%	70.75%	70.50%	67.50%	75.00%	73.50%
D → E	73.18%	74.44%	74.75%	71.50%	77.00%	79.00%
E → D	71.43%	72.00%	73.75%	71.50%	77.50%	76.00%
Average	74.50%	75.15%	75.28%	74.32%	76.92%	77.75%

Table 18: This table shows the accuracy of $MC_{Algorithm}$ to which a feature selection method was previously applied. *OW* states that only opinion words have been selected as features. In all others, apart from opinion words, other features have been selected according to scores assigned by a FS method. In $CHI2\_LogDF^{s}_{min}$-250-*INF-un*, scores have been computed by means of $CHI2 \cdot LogDF^{s}_{min}$, keeping the best 250 features belonging to the source, while no feature belonging just to the target has been maintained. Similarly in *GR*-250-*INF-un*, with only exception that scores have been computed by means of *GR*. Again, in *GR*-500-*INF-un* scores have been computed by means of *GR*, but this time keeping the best 500 features belonging to the source.

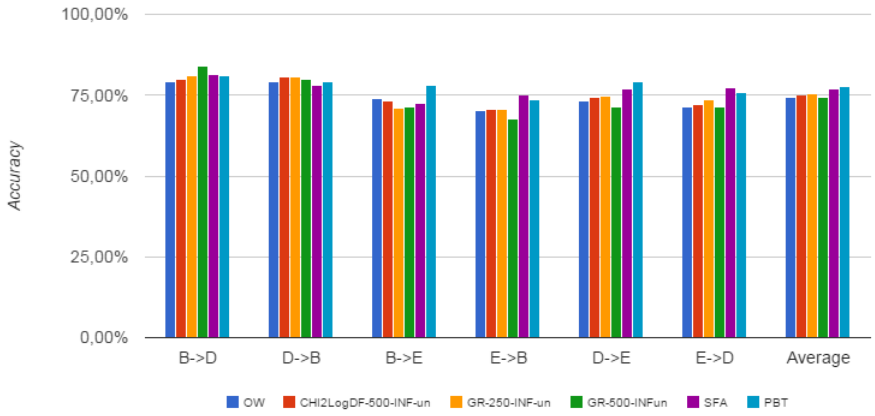

Fig. 8: *The figure shows a comparison between some proposed methods combining opinion words with other FS techniques on the one hand and SFA and PBT on the other hand.*

5.4.4 *Adding feature selection ranking to the MCTM*

Careful readers might have noticed that, until now, many supervised FS methods have been applied, but just to choose the feature set to be used in the standard $MC_{Algorithm}$. However, since every FS method produces a feature ranking, we could consider scores as relevance factors. In fact, the more score is high the more the corresponding feature will be important for sentiment classification. Therefore, we could think adding this ranking in $MC_{Algorithm}$, in particular multiplying the initial term weight by its rank when constructing the MCTM, as proposed by equation 62 in chapter 3. According to this little modification in the model, the same tests performed both with and without employing opinion words have been repeated. With respect to previous experiments, a suffix *rank* is added to distinguish these attempts. The outcome in table 19 evidences good accuracy values, better than previous ones and comparable with the state of the art. Moreover, we could notice that there is not a variant which always outperforms the others. Therefore, being these techniques little variants of the same general algorithm, this proves the stability of the basic $MC_{Algorithm}$. In fact, despite not having found yet the ideal feature set, which as we have seen would allow my algorithm outperforming both *SFA* and *PBT*, accuracy is never under a certain threshold. In other words, variance between one test and another is not significant on average.

	RF_{var}_$LogDF^{s}_{min}$-1000-INF-rank	**GR-500-INF-un-rank**	**IG_$LogDF^{s}_{min}$-250-INF-rank**	**CHI2-250-INF-rank**	**SFA**	**PBT**
B → D	82.25%	79.75%	78.00%	76.92%	81.50%	81.00%
D → B	82.50%	80.50%	78.89%	78.79%	78.00%	79.00%
B → E	70.93%	73.68%	72.47%	74.80%	72.50%	78.00%
E → B	70.50%	72.25%	73.33%	71.65%	75.00%	73.50%
D → E	71.68%	74.75%	75.52%	79.21%	77.00%	79.00%
E → D	73.25%	72.50%	73.92%	73.91%	77.50%	76.00%
Average	75.19%	75.57%	75.36%	75.88%	76.92%	77.75%

Table 19: This table shows the accuracy of $MC_{Algorithm}$. Term weights have been modified multiplying them by the rank returned from the applied FS method. Features belonging just to the target have never been maintained. In $RF_{var}\_LogDF^{s}_{min}$-1000-INF-$rank$, scores have been computed by means of $RF_{var} \cdot LogDF^{s}_{min}$, keeping the best 1000 features belonging to the source. Similarly, in $IG\_LogDF^{s}_{min}$-250-INF-$rank$ scores have been computed by means of $IG \cdot LogDF^{s}_{min}$, keeping the best 250 features belonging to the source. In GR-500-INF-un-$rank$, opinion words have been retrieved and scores have been computed by means of GR, keeping the best 500 features belonging to the source. Finally, in $CHI2$-250-INF-$rank$ scores have been calculated by means of $CHI2$, keeping the best 250 features belonging to the source.

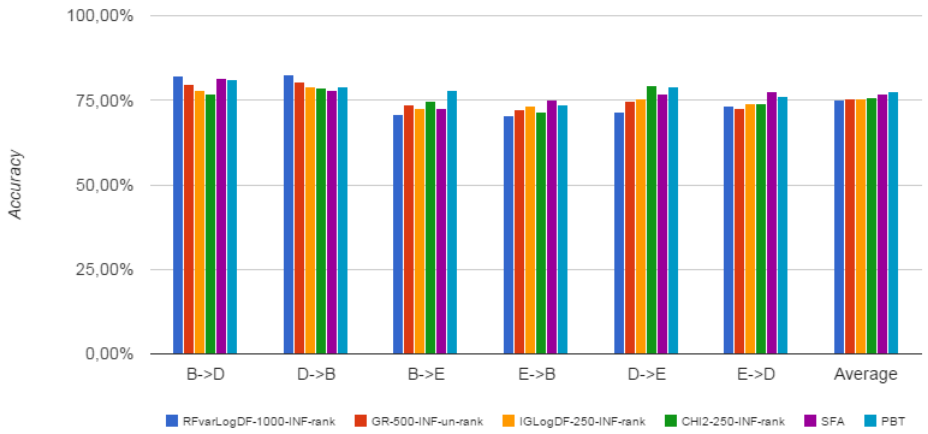

Fig. 9: *The figure shows a comparison between some proposed methods on the one hand and SFA and PBT on the other hand. The proposed techniques analyzed here make use of opinion words and a supervised FS method to build the dictionary of terms to be used for classification. Then, a term weighting proportional to the feature selection ranking is applied while constructing the MCTM.*

5.5 TESTING MULTI-SOURCE APPROACH

An unusual way to deal with a Cross-Domain Sentiment Classification problem is the Multi-Source approach, introduced in chapter 3. Remind that the idea consists in having more than just one source domain that can be exploited in order to build the MCTM. The aim is creating a more general model taking into account features extracted from possibly completely different domains. In this fashion, when considering a diverse domain, it should be simpler mapping its domain specific terms to other words already in the model and, as a consequence, it should be simpler classifying target documents. It is important to note that this variant of $MC_{Algorithm}$, referred as MC_{MS}, is not comparable with neither *SFA* nor *PBT*, since it is a conceptually different approach to Cross-Domain Sentiment Classification.
In order to test the method, the same experiments I have shown until now regarding the standard Cross-Domain Sentiment Classification approach have been reproduced. Since we are considering Books, DVDs and Electronics, there are only three cross-domain problems, namely $BD \rightarrow E$, $BE \rightarrow D$ and $DE \rightarrow B$. I would like to remind that the classification phase could be performed not only in the typical way, but also as explained by relation 63. I will refer to the latter way by adding the suffix *comb* in any presented variant employing it. The best attempts are shown in table 20, where we could see that *GR*

	GR-250-INF-rank-comb	**GR-250-INF-rank**	**RF_{var}_LogDF$^{s}_{min}$-250-5-rank-comb**	**RF_{var}_LogDF$^{s}_{min}$-250-5-rank**
BD → E	73.68%	73.18%	68.00%	67.50%
BE → D	74.44%	74.69%	78.50%	78.75%
DE → B	75.69%	75.69%	75.50%	76.75%
Average	74.60%	74.52%	74.00%	74.33%

Table 20: This table shows the performance of MC_{MS}. Several FS methods have been used to choose the dictionary, but the best results are those with GR and $RF_{var} \cdot LogDF^{s}_{min}$, always with 250 terms. RF_{var} relies on more than 250 features, because $DF^{target}_{min} = 5$ is applied as well. Furthermore, all the best working variants make use of the supervised ranking while constructing the MCTM. Finally, classification is possibly performed by combining posterior class probabilities.

and RF_{var} achieve comparable performance on average. Nevertheless, variance is lower when using GR rather than RF_{var}.
Differently from what I expected, the Multi-Source approach does not enhance performance with respect to the standard Cross-Domain Sentiment Classification procedure, probably because of its complexity.

5.6 ANALYSIS IN SINGLE-DOMAIN SENTIMENT CLASSIFICATION

The same analysis performed in Cross-Domain Sentiment Classification have been repeated in Single-Domain Sentiment Classification. For the sake of evaluating performance, $MC_{Algorithm}$ has been compared again with SFA and PBT, because in those papers have been also reported accuracy values over Books, DVDs, Electronics and Kitchen appliances in Single-Domain DSC. Once more, Kitchen appliances has been excluded from the analysis. The best results are visible at a glance in figure 10, whereas accuracy values are shown in table 21. The first thing that catches the eye is the supervised FS method employed: Gain Ratio is always the best working. Along this, another important factor is the usage of opinion words. On the other hand, features ranking affects performance, but not in a significant way. Anyway, my methods are completely aligned with the state of the art.
The most important consideration to be done is that my algorithm is absolutely general. In other words, without changing anything, just repeating the same analysis already carried out in the context of Cross-Domain DSC, performance are aligned with both SFA and PBT baselines, namely in Single-Domain Sentiment Classification. Further analysis might confirm these promising results in 2-classes DSC.

	GR-250-INF-un	GR-250-INF-un-rank	GR-500-INF-un-rank	SFA	PBT
Books	82.00%	81.00%	80.75%	81.40%	79.96%
DVDs	84.25%	85.00%	85.00%	82.55%	81.32%
Electronics	81.25%	82.50%	83.25%	84.60%	83.61%
Average	82.50%	82.83%	83.00%	82.85%	81.63%

Table 21: This table shows the performance of some variants of $MC_{Algorithm}$ in Single-Domain Sentiment Classification. To accomplish this experiment, both opinion words and supervised FS methods have been taken into account. Moreover, apart from *GR-250-INF-un*, also feature ranking has been used to assign weights while constructing the MCTM.

Moreover, it should be appropriate to test the algorithm also in a *n*-classes DSC problem, in order to extend its already large applicability.

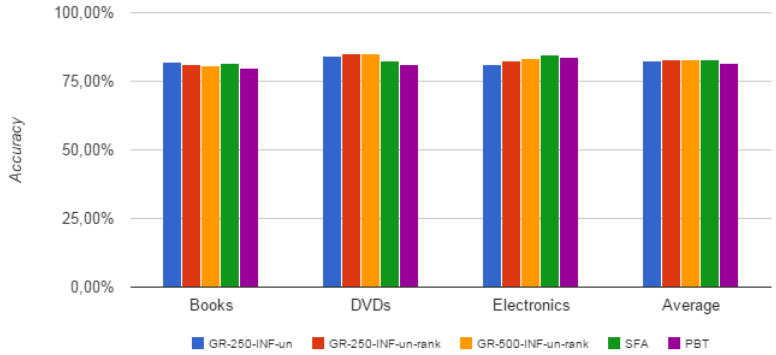

Fig. 10: *The figure compares some variants of $MC_{Algorithm}$ with both SFA and PBT, in a Single-Domain Sentiment Classification context. As immediately visible, results are approximately the same.*

5.7 TESTING REPSOL AND IBERIA

After having compared some variants of $MC_{Algorithm}$ with *SFA* and *PBT*, their performance have been evaluated over *Iberia* and *Repsol* as well. Remember that both datasets have been provided by the *Telematic Engineering Department at Charles III University of Madrid* and, therefore, comparisons with other approaches are not feasible. Remind also that both these datasets contain comments that people wrote on Facebook in Spanish language. This has several negative impacts on performance: firstly, comments are shorter than reviews

on average and it is not rare to find also sentences composed of single terms. Secondly, they are written in Spanish and, even if my method is language independent, Porter stemmer cannot be utilized. Despite the fact in Spanish some stemming algorithms are available, they have not been used in the analysis due to time limit. On the other hand, an opinion wordlist suggested in [61] have been used to execute some tests. Moreover, what could affect more the performance is the presence of 3 class labels: *positive*, *negative* and *neutral*. Finally, *F1-measure* is used instead of *accuracy* because the former is more suitable when analyzing DSC with more than two classes. In fact, even if a classifier could have a good accuracy, F1-measure could not to be good as much, because of the poor performance in one of the n classes. This is particularly true when there is skewness among classes. F1-measure is defined as follows:

$$F1 - measure = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{73}$$

where precision is defined as

$$precision = \frac{tp}{tp + fp} \tag{74}$$

and recall is defined as

$$recall = \frac{tp}{tp + fn} \tag{75}$$

In table 22 and in figure 11 are shown the best results the variants of my algorithm achieve in Single-Domain Sentiment Classification. Since the number of instances is lower in Iberia and Repsol than in Amazon, the chosen split percentage between training set and test set is 66%-33%, in order to have a reasonable number of documents to be tested. As expected, general performance is not so good, regardless the FS method, the number of terms retrieved and the usage of the opinion wordlist. In particular, OWs achieve the worst F1-measure if used alone in the FS process.
Further analysis might include a stemming technique, as previously suggested. Another possible strategy could be making use of a Natural Language Processing technique like Part-Of-Speech tagging to mark terms up by their corresponding parts of speech. In this way, it would be possible to define term weights also considering this aspect, which will be important above all due to the shortness of comments. At the end, it is likely that performance would enhance if a better FS method could be found.

The same experiments have been repeated also in Cross-Domain Sentiment Classification. Of course, there are just two cross-domain problems manageable, such as $R \rightarrow I$ and $I \rightarrow R$. Again, we could see the best analysis in table 23 and in figure 12. At a first glance we might notice that, both here and in Single-Domain DSC, the best

	OW	GR-250-INF-un	GR-500-INF-un	IG-ALL-3	CHI2_LogDF$^{s}_{min}$-500-INF-un
Iberia	58.52%	59.15%	60.43%	60.08%	62.20%
Repsol	53.41%	60.69%	61.64%	60.80%	62.94%
Average	55.97%	59.92%	61.04%	60.44%	62.57%

Table 22: This table shows the comparison between some variants of $MC_{Algorithm}$ in Single-Domain Sentiment Classification over *Iberia* and *Repsol* datasets. We could see that F1-measures are approximately the same, regardless the particular FS method, the exact number of features and the usage of opinion words. OWs do not help so much in enhancing results.

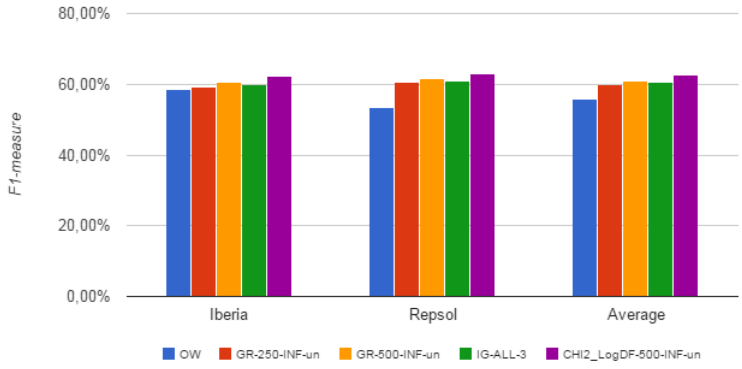

Fig. 11: *The figure shows the comparison between some variants of* $MC_{Algorithm}$ *in Single-Domain Sentiment Classification over Iberia and Repsol datasets. F1-measure is the metric used for the comparison.*

performing FS methods are the same. Another thing I would like to point out is that $I \rightarrow R$ always achieves better F1-measures than $R \rightarrow I$, regardless the parameters employed. Probably, the reason why this happens is that, since in Repsol we only have 1001 documents, less than 700 instances used as training set are not enough to build a good model. On the other hand, in Iberia we have 1693 documents and, therefore, more than 1000 have been utilized as training set. Nevertheless, performance is generally quite poor regarding these Spanish datasets, especially in comparison with the Amazon ones. For the sake of enhancing results, further analysis should be done, according to what previously said talking about Single-Domain DSC tests.

	OW	GR-250-INF-un	GR-500-INF-un	IG-ALL-3	CHI2_LogDF$^{s}_{min}$-500-INF-un
R → I	41.68%	42.61%	45.42%	48.39%	46.19%
I → R	48.17%	49.09%	50.82%	55.04%	50.86%
Average	44.93%	45.85%	48.12%	51.72%	48.53%

Table 23: This table shows the comparison between some variants of $MC_{Algorithm}$ in Cross-Domain Sentiment Classification over *Iberia* and *Repsol* datasets. Notice that F1-measures are always better over $I \rightarrow R$ than over $R \rightarrow I$.

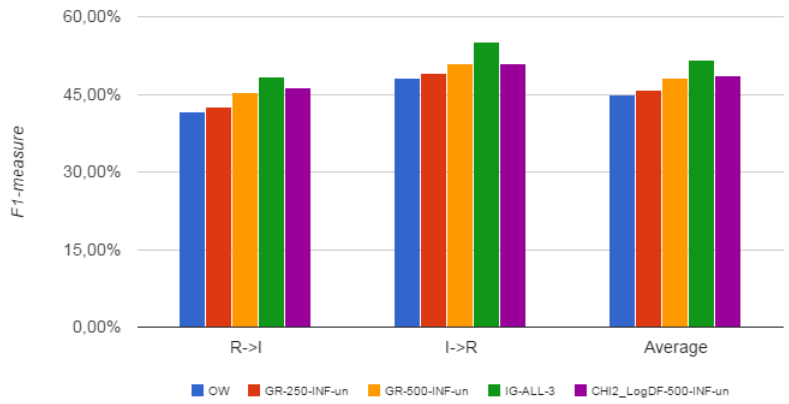

Fig. 12: *The figure shows the comparison between some variants of $MC_{Algorithm}$ in Cross-Domain Sentiment Classification over Iberia and Repsol datasets. F1-measure is the metric used for the comparison.*

CONCLUSIONS

This dissertation focuses on Document Sentiment Classification, both in single-domain and in cross-domain tasks. In the latter, the biggest problem consists in bridging the gap between source domain and target domain, which often express semantically related concepts by means of different terms. This trouble has induced some researchers to develop complex techniques, such as clustering, feature augmentation algorithms, hierarchical thesaurus, and so on, for the sake of aligning two domains. Furthermore, after having reduced the inter-domains gap, a classification algorithm is always needed to perform Sentiment Classification. On the other hand, this work proposes a unique model to deal with both problems, based on the mathematical theory of Markov Chains. This choice is worthy because a mathematical theory guarantees to easily control the soundness of the entire approach and, moreover, Markov Chains have been proved useful in many scientific works.

New algorithms have been introduced based on Markov Chains. Basically, the idea these novel techniques rely on is modeling term-term relationships. Each connection weight is higher the more two terms co-occur in documents and the more they are relevant. In such a way, simply adding class labels in the model, we could also link terms with categories and exploit this information to perform classification. At the same time, connections between terms allow aligning dissimilar domains. The only assumption needed is that source and target must have some terms in common, which is a reasonable hypothesis because there are a lot of domain independent words used to express sentiment, such as good, bad, and so on. Firsts experiments show good performance in cross-domain, but not as much as the state of the art. Other advanced variants have been employed in order to improve performance. On the one side, document expansion by means of Markov Chain Stationary Distribution has been proved not being useful. Furthermore, a multi-source approach to Document Sentiment Classification, where more than one source domain is utilized in the learning phase in order to increase the number of common terms, does not help as well. Probably, the reason why both techniques do not work well is due to their complexity, which adds noise to the algorithm rather than incrementing its classification capability. On the other side, an enhancement of performance can be achieved thanks to a good feature selection process. Three strategies that could aid the sentiment classification process have been tried. The first one involves the usage of an opinion wordlist, including well-known polarity words. The second one is the application of a su-

pervised features selection technique, which enables to choose terms that can positively affect the categorization. The last one consists in exploiting the ranking returned by those feature selection methods, in order to change term weights inside the Markov Chain Transition Matrix, so that they are proportional to their ranking. Combining these three variants, it is possible to achieve results comparable with the state of the art in 2-classes Cross-Domain Document Sentiment Classification. In particular, the last two mentioned are enough, because the supervised feature selection process already chooses the majority of polarity terms. The same experiments have been carried out in single-domain contexts too. Again, results are comparable with the literature, confirming the algorithm capability. Finally, analysis in a 3-classes context shows not so good performance. The first motivation is that the two datasets analyzed contain very short comments. The second one is that they are written in Spanish and no stemming algorithm has been used due to time limit.

An advantage of the method proposed in this dissertation is simplicity, because the same model allows both bridging source-target gap and executing classification. What is more, few parameters are required to be calibrated. Apart from being simpler than previous approaches, another killer feature is generality. Firstly, the algorithm is language independent, since it is just based on term co-occurrences. Then, it works well both in single-domain and in cross-domain problems.

A performed analysis has demonstrated that an adequate feature selection method is the bottle neck, because it could potentially allow the proposed algorithm outperforming earlier approaches. Therefore, further analysis could focus on trying to extract the best feature set. A possible way to walk is adapting the technique illustrated by Domeniconi et al. in [62], which iteratively refined target category representation in Cross-Domain Text Categorization context, to the feature selection process. A different alternative consists in introducing also bigram features, namely features composed of two terms. Since it is feasible to find words like *not* and *good* in a sentence, bigrams might help because for example *not_good* expresses a completely different polarity from *good*. Another future work regards extending the validity of results to n-classes Document Sentiment Classification, performing a more detailed analysis to deeply understand the reason behind the lack of performance in 3-classes datasets. Furthermore, due to its generality, the model could theoretically be applied to Text Categorization. So, it would be interesting to investigate its applicability in this context too. Finally, another possibility is trying to extend the consideration done in this dissertation regard Regular Markov Models to Hidden Markov Models, exploiting their greater expressive power.

BIBLIOGRAPHY

[1] PANG, Bo; LEE, Lillian. Opinion mining and sentiment analysis. Foundations and trends in information retrieval, 2008, 2.1-2: 1-135.

[2] JINDAL, Nitin; LIU, Bing. Identifying comparative sentences in text documents. In: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2006. p. 244-251.

[3] LI, Fangtao; HUANG, Minlie; ZHU, Xiaoyan. Sentiment Analysis with Global Topics and Local Dependency. In: AAAI. 2010.

[4] DAVE, Kushal; LAWRENCE, Steve; PENNOCK, David M. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In: Proceedings of the 12th international conference on World Wide Web. ACM, 2003. p. 519-528.

[5] PALTOGLOU, Georgios; THELWALL, Mike. A study of information retrieval weighting schemes for sentiment analysis. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, 2010. p. 1386-1395.

[6] TAN, Songbo; WANG, Yuefen; CHENG, Xueqi. Combining learn-based and lexicon-based techniques for sentiment detection without using labeled examples. In: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2008. p. 743-744.

[7] QIU, Likun, et al. Selc: a self-supervised model for sentiment classification. In: Proceedings of the 18th ACM conference on Information and knowledge management. ACM, 2009. p. 929-936.

[8] MELVILLE, Prem; GRYC, Wojciech; LAWRENCE, Richard D. Sentiment analysis of blogs by combining lexical knowledge with text classification. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2009. p. 1275-1284.

[9] DENG, Zhi-Hong; LUO, Kun-Hu; YU, Hong-Liang. A study of supervised term weighting scheme for sentiment analysis. Expert Systems with Applications, 2014, 41.7: 3506-3513.

[10] WU, Haibing; GU, Xiaodong. Reducing Over-Weighting in Supervised Term Weighting for Sentiment Analysis.

[11] AUE, A.; GAMON M. Customizing sentiment classifiers to new domains: a case study. In Proceedings of Recent Advances in Natural Language Processing (RANLP-2005), 2005.

[12] BOLLEGALA, Danushka; WEIR, David; CARROLL, John. Cross-domain sentiment classification using a sentiment sensitive thesaurus. Knowledge and Data Engineering, IEEE Transactions on, 2013, 25.8: 1719-1731.

[13] BLITZER, J.; DREDZE M.; PEREIRA F. Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL-2007), 2007.

[14] PAN, Sinno Jialin, et al. Cross-domain sentiment classification via spectral feature alignment. In: Proceedings of the 19th international conference on World wide web. ACM, 2010. p. 751-760.

[15] YANG, Hui; CALLAN, Jamie; SI, Luo. Knowledge Transfer and Opinion Detection in the TREC 2006 Blog Track. In: TREC. 2006.

[16] TABOADA, Maite, et al. Lexicon-based methods for sentiment analysis. Computational linguistics, 2011, 37.2: 267-307.

[17] TURNEY, Peter D. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In: Proceedings of the 40th annual meeting on association for computational linguistics. Association for Computational Linguistics, 2002. p. 417-424.

[18] GREENGRASS, Ed. Information retrieval: A survey. 2000.

[19] HOENKAMP, Eduard, et al. An effective approach to verbose queries using a limited dependencies language model. Springer Berlin Heidelberg, 2009.

[20] LANGVILLE, Amy N.; MEYER, Carl D. A survey of eigenvector methods for web information retrieval. SIAM review, 2005, 47.1: 135-161.

[21] WEI, Xing; CROFT, W. Bruce. LDA-based document models for ad-hoc retrieval. In: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2006. p. 178-185.

[22] MILLER, David RH; LEEK, Tim; SCHWARTZ, Richard M. A hidden Markov model information retrieval system. In: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 1999. p. 214-221.

[23] SARUKKAI, Ramesh R. Link prediction and path analysis using markov chains. Computer Networks, 2000, 33.1: 377-386.

[24] MITTENDORF, Elke; SCHÄUBLE, Peter. Document and passage retrieval based on hidden Markov models. In: SIGIR'94. Springer London, 1994. p. 318-327.

[25] QIU, Liwen. Markov models of search state patterns in a hypertext information retrieval system. Journal of the American Society for Information Science, 1993, 44.7: 413-427.

[26] PAGE, Lawrence, et al. The PageRank citation ranking: Bringing order to the web. 1999.

[27] JIN, Wei; HO, Hung Hay; SRIHARI, Rohini K. OpinionMiner: a novel machine learning system for web opinion mining and extraction. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2009. p. 1195-1204.

[28] XU, Jinxi; WEISCHEDEL, Ralph. Cross-lingual information retrieval using hidden Markov models. In: Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13. Association for Computational Linguistics, 2000. p. 95-103.

[29] PAN, Yi-Cheng; LEE, Hung-Yi; LEE, Lin-Shan. Interactive spoken document retrieval with suggested key terms ranked by a Markov decision process. Audio, Speech, and Language Processing, IEEE Transactions on, 2012, 20.2: 632-645.

[30] CAO, Guihong; NIE, Jian-Yun; BAI, Jing. Using markov chains to exploit word relationships in information retrieval. In: Large Scale Semantic Access to Content (Text, Image, Video, and Sound). LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE, 2007. p. 388-402.

[31] XU, Rong, et al. Combining text classification and hidden Markov modeling techniques for structuring randomized clinical trial abstracts. In: AMIA Annual Symposium Proceedings. American Medical Informatics Association, 2006. p. 824.

[32] YI, Kwan; BEHESHTI, Jamshid. A hidden Markov model-based text classification of medical documents. Journal of Information Science, 2008.

[33] MEI, Qiaozhu, et al. Topic sentiment mixture: modeling facets and opinions in weblogs. In: Proceedings of the 16th international conference on World Wide Web. ACM, 2007. p. 171-180.

[34] ZHOU, Xueyuan; LI, Chunping. Text Classification by Markov Random Walks with Reward. In: DMIN. 2005. p. 275-278.

[35] NASUKAWA, Tetsuya; YI, Jeonghee. Sentiment analysis: Capturing favorability using natural language processing. In: Proceedings of the 2nd international conference on Knowledge capture. ACM, 2003. p. 70-77.

[36] VIEIRA, Adrián Seara; IGLESIAS, Eva Lorenzo; DIZ, Lourdes Borrajo. Study and application of Hidden Markov Models in scientific text classification.

[37] FRASCONI, Paolo; SODA, Giovanni; VULLO, Alessandro. Hidden markov models for text categorization in multi-page documents. Journal of Intelligent Information Systems, 2002, 18.2-3: 195-217.

[38] LI, Kairong; CHEN, Guixiang; CHENG, Jilin. Research on hidden markov model-based text categorization process. International Journal of Digital Content Technology and its Application, 2011, 5.6: 244-251.

[39] YI, Kwan; BEHESHTI, Jamshid. A text categorization model based on Hidden Markov models. In: Proceedings of the 31st Annual Conference of the Canadian Association for Information Science. 2003. p. 275-287.

[40] LI, Jin; YUE, Kun; LIU, WeiYi. An adaptive Markov model for text categorization. In: Intelligent System and Knowledge Engineering, 2008. ISKE 2008. 3rd International Conference on. IEEE, 2008. p. 802-807.

[41] LI, Fang; DONG, Tao. Text Categorization Based on Semantic Cluster-Hidden Markov Models. In: Advances in Swarm Intelligence. Springer Berlin Heidelberg, 2013. p. 200-207.

[42] WREN, Jonathan D., et al. Markov model recognition and classification of DNA/protein sequences within large text databases. Bioinformatics, 2005, 21.21: 4046-4053.

[43] NEUMANN, Marion; AHMADI, Babak; KERSTING, Kristian. Markov Logic Sets: Towards Lifted Information Retrieval Using PageRank and Label Propagation. In: AAAI. 2011.

[44] ZHANG, Hua-Ping, et al. Chinese lexical analysis using hierarchical hidden markov model. In: Proceedings of the second SIGHAN workshop on Chinese language processing-Volume 17. Association for Computational Linguistics, 2003. p. 63-70.

[45] MILBRETT, Jessica L. Aspect and Sentiment Unification Model for Online Review Analysis. 2011.

[46] MORENCY, Louis-Philippe; MIHALCEA, Rada; DOSHI, Payal. Towards multimodal sentiment analysis: Harvesting opinions from the web. In: Proceedings of the 13th international conference on multimodal interfaces. ACM, 2011. p. 169-176.

[47] GHOSE, Udayan; RAI, C. S.; SINGH, Yogesh. Application of Markov Process Model and Entropy Analysis in Data Classification and Information Retrieval. International Journal on Computer Science and Engineering, 2010, 2.3.

[48] BENOIT, G. Weighted Markov chains and graphic state nodes for information retrieval. Proceedings of the American Society for Information Science and Technology, 2002, 39.1: 115-123.

[49] MORENCY, L.; QUATTONI, Ariadna; DARRELL, Trevor. Latent-dynamic discriminative models for continuous gesture recognition. In: Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on. IEEE, 2007. p. 1-8.

[50] JINDAL, Nitin; LIU, Bing. Mining comparative sentences and relations. In: AAAI. 2006. p. 1331-1336.

[51] CHOUDHURY, Monojit, et al. Investigation and modeling of the structure of texting language. International Journal of Document Analysis and Recognition (IJDAR), 2007, 10.3-4: 157-174.

[52] AUSTIN, Tom, et al. Introducing the High-Performance Workplace: Improving Competitive Advantage and Employee Impact. Gartner Research, 16th May, 2005.

[53] BERRY, Michael W. Survey of text mining. Computing Reviews, 2004, 45.9: 548.

[54] GUPTA, Vishal; LEHAL, Gurpreet S. A survey of text mining techniques and applications. Journal of emerging technologies in web intelligence, 2009, 1.1: 60-76.

[55] PAN, Sinno Jialin, et al. Cross-domain sentiment classification via spectral feature alignment. In: Proceedings of the 19th international conference on World wide web. ACM, 2010. p. 751-760.

[56] HE, Yulan; LIN, Chenghua; ALANI, Harith. Automatically extracting polarity-bearing topics for cross-domain sentiment classification. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1. Association for Computational Linguistics, 2011. p. 123-131.

[57] BOLLEGALA, Danushka; WEIR, David; CARROLL, John. Using multiple sources to construct a sentiment sensitive thesaurus for cross-domain sentiment classification. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1. Association for Computational Linguistics, 2011. p. 132-141.

[58] PORTER, Martin F. An algorithm for suffix stripping. Program, 1980, 14.3: 130-137.

[59] LAN, Man, et al. Supervised and traditional term weighting methods for automatic text categorization. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2009, 31.4: 721-735.

[60] LIU, Bing. Sentiment analysis and opinion mining. Synthesis Lectures on Human Language Technologies, 2012, 5.1: 1-167.

[61] MOLINA-GONZÁLEZ, M. Dolores, et al. Semantic orientation for polarity classification in Spanish reviews. Expert Systems with Applications, 2013, 40.18: 7250-7257.

[62] DOMENICONI, Giacomo, et al. Cross-domain Text Classification through Iterative Refining of Target Categories Representations.

[63] PAN, Sinno Jialin; YANG, Qiang. A survey on transfer learning. Knowledge and Data Engineering, IEEE Transactions on, 2010, 22.10: 1345-1359.

[64] JOACHIMS, Thorsten. Text categorization with support vector machines: Learning with many relevant features. Springer Berlin Heidelberg, 1998.

[65] DUMAIS, Susan, et al. Inductive learning algorithms and representations for text categorization. In: Proceedings of the seventh international conference on Information and knowledge management. ACM, 1998. p. 148-155.

[66] YANG, Yiming; LIU, Xin. A re-examination of text categorization methods. In: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 1999. p. 42-49.

[67] SEBASTIANI, Fabrizio. Machine learning in automated text categorization. ACM computing surveys (CSUR), 2002, 34.1: 1-47.

[68] MERKL, Dieter. Text classification with self-organizing maps: Some lessons learned. Neurocomputing, 1998, 21.1: 61-77.

[69] KOHONEN, Teuvo, et al. Self organization of a massive document collection. Neural Networks, IEEE Transactions on, 2000, 11.3: 574-585.

[70] WEIGEND, Andreas S.; WIENER, Erik D.; PEDERSEN, Jan O. Exploiting hierarchy in text categorization. Information Retrieval, 1999, 1.3: 193-216.

[71] SCOTT, Sam; MATWIN, Stan. Text classification using WordNet hypernyms. In: Use of WordNet in natural language processing systems: Proceedings of the conference. 1998. p. 38-44.

[72] GABRILOVICH, Evgeniy; MARKOVITCH, Shaul. Computing Semantic Relatedness Using Wikipedia-based Explicit Semantic Analysis. In: IJCAI. 2007. p. 1606-1611.

[73] DAI, Wenyuan, et al. Co-clustering based classification for out-of-domain documents. In: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2007. p. 210-219.

[74] XUE, Gui-Rong, et al. Topic-bridged PLSA for cross-domain text classification. In: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2008. p. 627-634.

[75] LI, Lianghao; JIN, Xiaoming; LONG, Mingsheng. Topic Correlation Analysis for Cross-Domain Text Classification. In: AAAI. 2012.

[76] HOWARD, Ronald A. Dynamic Probabilistic Systems, Volume I: Markov Models. Courier Corporation, 2012.

[77] KEMENY, John G.; SNELL, J. Laurie. Finite continuous time Markov chains. Theory of Probability & Its Applications, 1961, 6.1: 101-105.

[78] BLEI, David M.; NG, Andrew Y.; JORDAN, Michael I. Latent dirichlet allocation. the Journal of machine Learning research, 2003, 3: 993-1022.

[79] RABINER, LAWRENCE R. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. PROCEEDINGS OF THE IEEE, 1989, 77.2: 257.

[80] CHAN, Lois Mai. Cataloging and classification: an introduction. Scarecrow Press, 2007.

[81] MANNING, Christopher D. Foundations of statistical natural language processing. MIT press, 1999.

[82] NORRIS, James R. Markov chains. Cambridge university press, 1998.

[83] SERFOZO, Richard. Basics of applied stochastic processes. Springer Science [**?**] Business Media, 2009.

[84] AGGARWAL, Charu C.; ZHAI, ChengXiang. Mining text data. Springer Science & Business Media, 2012.

[85] WEST, Douglas Brent, et al. Introduction to graph theory. Upper Saddle River: Prentice hall, 2001.

[86] BAUM, Leonard E., et al. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. The annals of mathematical statistics, 1970, 164-171.