École Polytechnique
Laboratoire d'Informatique de l'X (LIX)

GRAPH-OF-WORDS: MINING AND RETRIEVING TEXT WITH NETWORKS OF FEATURES

DISSERTATION

submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in Computer Science

by

FRANÇOIS ROUSSEAU

September 18, 2015

Thesis prepared under the supervision of Michalis Vazirgiannis
at the Department of Informatics of École Polytechnique

Laboratoire d'Informatique de l'X (LIX) – UMR 7161
1 rue Honoré d'Estienne d'Orves
Bâtiment Alan Turing
Campus de l'École Polytechnique
91120 Palaiseau, France

To my mom and dad who have always wanted a doctor,
and to my sister who made me become one.

ABSTRACT

We propose graph-of-words as an alternative document representation to the historical bag-of-words that is extensively used in text mining and retrieval. We represent textual documents as statistical graphs whose vertices correspond to unique terms of the document and whose edges represent co-occurrences between the terms within a fixed-size sliding window over the full processed document. The underlying assumption is that all the words present in a document have some relationships with one another, modulo a window size outside of which the relationship is not taken into consideration. This representation takes better into account word dependency compared to traditional unigrams and even n-grams and can be interpreted as a network of features that captures the relations between terms at the document level or even at the collection level when considering a giant graph-of-words made of the aggregation of multiple smaller graph-of-words, one for each document of the collection.

In our work, we capitalized on the graph-of-words representation to retrieve more relevant information, extract more cohesive keywords and learn more discriminative patterns with successful applications in ad hoc information retrieval, single-document keyword extraction and single-label multi-class text categorization. Experiments conducted on various text datasets with ground truth data, including a Web-scale collection of 25M documents, and using standard evaluation metrics for each task (e. g., MAP, P@10, accuracy and macro-average F1-score) resulted in statistically significant improvements in effectiveness for little to no additional cost in efficiency.

The main findings of our research are: (1) for ad hoc information retrieval, when assessing a query term's weight in a document, rather than considering the overall term frequency of a word and then applying a concave transformation to ensure a decreasing marginal gain in relevance, one should instead consider for each word the number of distinct contexts of co-occurrence with other words so as to favor terms that appear with a lot of other terms, i. e. consider the node degree in the corresponding unweighted graph-of-words; (2) for keyword extraction, humans tend to select as keywords not only central but also densely connected nodes in the corresponding weighted graph-of-words, property captured by reducing the graph to its main core using the concept of graph degeneracy; and (3) for text categorization, long-distance n-grams – defined as subgraphs of unweighted graph-of-words – are more discriminative features than standard n-grams, partly because they can capture more variants of the same set of terms compared to fixed sequences of terms and therefore appear in more documents.

RÉSUMÉ

Nous proposons la représentation de documents par graphe-de-mots comme alternative à la représentation par sac-de-mots qui est largement utilisée en fouille de données et recherche d'information dans les textes. Nous représentons les documents à l'aide de graphes statistiques dont les nœuds correspondent aux uniques termes du document et dont les arêtes représentent les co-occurrences entre les termes dans une fenêtre glissante de taille fixe. L'hypothèse sous-jacente étant que tous les mots d'un document sont en lien, modulo la taille de la fenêtre en dehors de laquelle le lien n'est pas pris en compte. Cette représentation prend mieux en compte la dépendance entre les mots qu'avec une représentation plus traditionnelle se basant sur les unigrammes et même les n-grammes et peut être interprétée comme un réseau de variables qui stocke les relations entre les termes à l'échelle du document ou même à l'échelle d'une collection de documents lorsque l'on considère un graphe-de-mots construit à partir de plusieurs graphe-de-mots plus petits, un par document de la collection.

Au cours de nos travaux, nous avons tiré profit de la représentation par graphe-de-mots pour mieux rechercher les informations les plus pertinentes à une requête, pour extraire des mots-clés plus cohésifs et pour apprendre des motifs plus discriminants, avec des applications en recherche ad hoc d'information, en extraction de mots-clés et en classification de textes. Les expériences menées sur de nombreux jeux de données dont on connaît la vérité terrain, parmi lesquels une collection de 25M de pages Web, et en utilisant les mesures standards d'évaluation pour chaque tâche (MAP, P@10, taux de bonne classification et macro-average F1-score) ont conduites à des améliorations statistiquement significatives en qualité pour peu voire pas de coût supplémentaire en efficacité.

Les principaux résultats de notre recherche sont : (1) en recherche ad hoc d'information, lorsque l'on évalue le poids d'un terme de la requête dans un document, au lieu de considérer la fréquence globale du terme et ensuite lui appliquer une transformation concave pour s'assurer d'un gain marginal en pertinence décroissant, on devrait plutôt considérer pour chaque mot le nombre distinct de contexte de co-occurrences avec les autres mots de façon à favoriser les mots qui apparaissent avec le plus grand nombre de mots différents, c'est-à-dire considérer le degré du nœud dans le graphe-de-mots non pondéré correspondant ; (2) en extraction de mots-clés, les humains ont tendance à sélectionner les nœuds non seulement centraux mais aussi connectés densément aux autres nœuds dans le graphe-de-mots pondéré correspondant, propriété qui se retrouve lorsque l'on réduit un réseau à son core principal en utilisant le principe de dégénérescence de graphe ; et (3) en classification de textes, les n-grammes dits de longue distance – définit comme des sous-graphes de graphe-de-mots – sont plus discriminants que les n-grammes standards, en partie parce qu'ils couvrent plus de variantes du même ensemble de termes comparés à des séquences fixes de termes et ainsi ils apparaissent dans plus de documents.

PUBLICATIONS

The following publications are included in parts or in an extended version in this dissertation:

1) F. Rousseau and M. Vazirgiannis. Composition of TF normalizations: new insights on scoring functions for ad hoc IR. in *Proceedings of the 36th annual international ACM SIGIR conference on research and development in information retrieval*. In SIGIR '13. ACM, 2013, pages 917–920.
2) F. Rousseau and M. Vazirgiannis. Graph-of-word and TW-IDF: new approach to ad hoc IR. in *Proceedings of the 22nd ACM international conference on information and knowledge management*. In CIKM '13. ACM, 2013, pages 59–68. **Best Paper, Honorable Mention**.
3) F. Rousseau and M. Vazirgiannis. Main core retention on graph-of-words for single-document keyword extraction. In *Proceedings of the 37th european conference on information retrieval*. In ECIR '15. Springer-Verlag, 2015, pages 382–393.
4) F. Rousseau, E. Kiagias, and M. Vazirgiannis. Text categorization as a graph classification problem. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing*. Volume 1. In ACL-IJCNLP '15. ACL, 2015, pages 1702–1712.
5) J. Kim, F. Rousseau, and M. Vazirgiannis. Convolutional sentence kernel from word embeddings for short text categorization. In *Proceedings of the 2015 conference on empirical methods in natural language processing*. In EMNLP '15. ACL, 2015.

Furthermore, the following publications were part of my Ph.D. research but are not covered in this dissertation – the topics of these publications are somewhat outside of the scope of the material covered here:

6) M. Karkali, F. Rousseau, A. Ntoulas, and M. Vazirgiannis. Efficient online novelty detection in news streams. In *Proceedings of the 14th international conference on web information systems engineering*. In WISE '13. Springer-Verlag Berlin, 2013, pages 57–71.
7) P. Meladianos, G. Nikolentzos, F. Rousseau, Y. Stavrakas, and M. Vazirgiannis. Degeneracy-based real-time sub-event detection in twitter stream. In *Proceedings of the 9th AAAI international conference on web and social media*. In ICWSM '15. AAAI Press, 2015, pages 248–257.
8) F. Rousseau, J. Casas-Roma, and M. Vazirgiannis. Community-preserving anonymization of social networks. *ACM transactions on knowledge discovery from data*, 2015. Submitted on 24/11/2014.

9) J. Casas-Roma and F. Rousseau. Community-preserving generalization of social networks. In *Proceedings of the social media and risk ASONAM 2015 workshop*. In SoMeRis '15. IEEE Computer Society, 2015.

10) P. Meladianos, G. Nikolentzos, F. Rousseau, Y. Stavrakas, and M. Vazirgiannis. Shortest-path graph kernels for document similarity. In *Proceedings of the 15th IEEE international conference on data mining*. In ICDM '15. IEEE Computer Society, 2015. Submitted on 03/06/2015.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor, Prof. **Michalis Vazirgiannis**, without whom my Ph.D. at École Polytechnique and this dissertation would not have been possible. I have not only gained a lot of valuable knowledge throughout these past three years but he also gave me the opportunity to give back through teaching and supervision of interns. Our various contributions to the field of text mining and my involvement in the research community were born out of our endless discussions and he was the one who encouraged me into pursuing graph-based representations of text, which gave rise to the concept of *graph-of-words*.

Secondly, I would like to thank the collaborators with whom I worked with and co-authored papers, in chronological order: Dr. **Margarita Karkali** and Dr. **Alexandros Ntoulas** who introduced me to the task of novelty detection; Dr. **Jordi Casas Roma** who initiated me to the field of privacy-preserving data mining, in particular in the context of social networks; and **Polykarpos Meladianos** and **Giannis Nikolentzos** who are already extending my work to streams of documents and definitions of kernels between graph-of-words. I also had the chance to supervise four interns throughout the years – **Emmanouil Kiagias**, **Jonghoon Kim**, **Konstantinos Skianis** and **Giannis Bekoulis** – thanks to whom I was able to explore even further and deeper the field while learning to guide people in a research environment and pass my knowledge on to them.

Furthermore, I want to express my gratitude to the distinguished researchers who accepted to be part of my Ph.D. thesis defense committee – Prof. **Stéphane Gaïffas**, Prof. **Patrick Gallinari**, Dr. **Gregory Grefenstette**, Prof. **Pierre Senellart** and Prof. **Jean-Marc Steyaert** – and in particular to Prof. **Éric Gaussier** and Prof. **ChengXiang Zhai** who reviewed this dissertation.

Additionally, I would like to thank my colleagues, officemates and campus friends, with whom I exchanged and shared not only thoughts but food as well: **Kostas Tavlaridis-Gyparakis**, **Pierre Coquelin**, **Arnaud Leloup**, **Arnaud Dumas**, Dr. **Christos Giatsidis**, **Fragkiskos Malliaros**, **Panagiotis Korvesis**, **Vangelis Anagnostopoulos** and **Maria Rossi**. I definitely learnt a lot about the Greek culture, maybe more than expected but very valuable in light of today's news. Ευχαριστώ τους φίλους και συνεργάτες!

Finally, I would like to thank my parents, **Maryannick Guérin** and **Alain Rousseau**, and my sister **Anne-Lise Rousseau**, for their unconditional support throughout these three years and I dedicate this dissertation to them. Thank you for being there and come visit me in this remote and quasi-monastic location that École Polytechnique is!

Merci à tous,

François

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Research is to see what everybody else has seen,
and to think what nobody else has thought.

— Albert Szent-Györgyi de Nagyrápolt

PREFACE

Three years ago, in summer 2012, I decided to embark on a Ph.D. journey that would take me to the frontiers of computer science and in particular to those of data science and natural language processing. During these past years, I have had the chance to be a data miner, an information retriever and a machine teacher. I may not have witnessed machines expressing human intelligence explicitly but I have definitely seen them show some sort of understanding of our world and not just memorization of it. Artificial intelligence has always been a dream if not a fantasy for many of us and I am happy to say that I came closer to achieving that dream during my Ph.D. When it comes to understanding natural language as we do, machines have still a long way to go but eventually they will get there, thanks to a representation of text that encompasses every subtlety we, as humans, decided to put in and that they can finally interpret and later generate on their own. I did not find that ultimate representation but hopefully the one I explored during my research was in the right direction and can help the community go forward.

To me, the core part of a Ph.D. revolves around two main components: (1) *learning* a new field, i. e. "seeing what everybody else has seen", and (2) *contributing* to it, i. e. "thinking what nobody else has thought". I think it is important to really master the tools beforehand in order to be able to build new things with them and propose novel ideas with sound foundations. It does take some time to delve in a field or a particular concept, but it is worth every second invested. I feel like I spent most of my Ph.D. time trying to find the right resources to understand things, i. e. the ones that were not only accurate but that also fit my way of understanding things, and it is in that spirit that I wrote this dissertation, trying to be as didactic as possible. I would definitely recommend teaching a field while learning it to make sure you understand it enough to pass on the knowledge and also to highlight the parts you thought you understood but you are actually unsure about – ideally to a class of students as teaching assistant but interns and friends will also do. Additionally, you will gain confidence in knowing what you know and what you don't.

Finally, I would like to share the recipe I found working best when tackling a new research issue. I think it can be useful especially for students starting a Ph.D. or a research internship. Start by gathering a few relevant research papers – maybe they were given to you by your advisor or you found them from keyword search. You do not need to read them entirely and in a sequence. Actually, I find it better to read them in parallel as follows. First, read all the introductions so as to get familiar with the task and its applications. That way,

you can also discard false positives early on – these papers seemed relevant from their title or abstract but they are not. Then, jump to the experimental sections (excluding the results) to understand how any proposed solution is evaluated and on which datasets, regardless of how the methods actually work. Try to cross-reference the evaluation metrics and the datasets between the papers as much as possible, so as to know which are the standard ones and which aren't – you can keep track of the information in a table if it helps. If you need more papers to reach an amount significant enough, you can start reading the related work sections. Soon, you will start building this mental citation network in your head that should help you decide which paper to read next. If a paper has been referenced in several papers you have been reading so far then it is probably a good candidate. If you stumble upon a good paper, in the sense of well-written and clear, prioritize the papers that are referenced (earlier ones) as well as the papers that referenced it (future ones). Finding the original work(s) that introduced the concept you are interested in is important and you should cite it when you are doing the write-up, but it might not be the best resource for you to understand it. Note that until then, you have not read the core parts of any paper. It may seem counter-intuitive but it avoids the common pitfall of getting stuck on one paper because of complex notions or, worst, complicated formulations. Basically, you do not know any of the solutions to the problem but it should now be well-defined in terms of input, output, evaluation metrics and datasets. You are now ready to delve in the proposed solutions, prioritizing the most common baselines and state-of-the-art methods. Try to reproduce their results on the datasets and using the evaluation metrics you have previously found to be standard before attempting something on your own. Once this framework is in place, it is now easier for you to explore novel ideas because you know what has been done before and what the performances were. It also avoids overfitting to one particular dataset or a non-standard one where your method performs better by chance or because of the wrong metric. If you are lucky enough to tackle a well-known research issue, there might be one or more textbooks that will take you through all these steps without requiring you to read all the papers beforehand, e. g., *Introduction to Information Retrieval* from Manning, Raghavan, et al. (2008) for text mining and retrieval, which I found to be the best. This is the recipe I followed for my own research during my Ph.D., the one I advised my interns to use and the reason why I structured my dissertation the way it is. Happy reading!

1

INTRODUCTION

Since the invention of writing in Mesopotamia circa 3200 BCE, text has rapidly become the preferred means of information storage and knowledge transfer if not the only one over long periods of time. Even in our modern and hyper-connected world, this asynchronous means of communication still trumps speech, the other common form of natural language (e. g., text messages over phone calls, emails over meetings or research papers over conference talks). The radio, the television, the cinema and nowadays more generally the podcasts and videos do play an important role and their use has certainly grown exponential in the past decades but they have their own research challenges beyond the scope of this dissertation. Moreover, many real-world applications combine both oral and written communications and thus the need for a better text processing in any case, e. g., text normalization for speech synthesis or language modeling for automatic captioning.

The unstructured nature of text makes the task of understanding and generating it harder for the machine than with other types of data with pre-defined structures such as relational databases, networks or numerical outputs of various sensors. Natural Language Processing (NLP) and at a higher-level Text Mining (TM) emerged in the 1950s has research fields, aiming at filling the gap between between machines and humans in terms of language comprehension and deriving information from text. Sixty years later, they are still active fields of research with countless problems that the machine seems miles away from solving in the near future, e. g., grasping the concepts of sarcasm, metaphor or paraphrase. Actually, even "simpler" tasks such as Keyword Extraction (KwE) or Text Categorization (TC) that a human would perform trivially are still on-going research topics.

That being said, with the tremendous amount of data available today – the so-called big data era – and text being no exception to its ever increasing scale, machines have become relatively good at tasks that humans cannot even perform anymore because of the daunting scale and the information overload, e. g., Information Retrieval (IR) with Web search engines that index billions of documents and Machine Learning (ML) with spam filters that discard automatically billions of junk emails everyday: two striking examples of the potential of machines, the limits of humans and thus the need for intelligent machines. It is in this spirit that we started the Ph.D., aiming at providing machines with an alternative representation of text, one that could make them better at IR, ML, NLP and more generally TM.

1.1 SCOPE

In this dissertation, we introduce graph-of-words as an alternative document representation to the historical bag-of-words that is extensively used in text mining and retrieval. We represent textual documents as statistical graphs whose vertices correspond to unique terms of the document and whose edges represent co-occurrences between the terms within a fixed-size sliding window over the full processed document. The underlying assumption is that all the words present in a document have some relationships with one another, modulo a window size outside of which the relationship is not taken into consideration. This representation takes better into account word dependency compared to traditional unigrams and even n-grams and can be interpreted as a network of features that captures the relations between terms at the document level or even at the collection level when considering a giant graph-of-words made of the aggregation of multiple smaller graph-of-words.

In our work, we capitalized on the graph-of-words representation to retrieve more relevant information, extract more cohesive keywords and learn more discriminative patterns with successful applications in ad hoc IR, single-document KwE and single-label multi-class TC. Experiments on various text datasets with ground truth data, including a Web-scale collection of 25M documents, and using standard evaluation metrics for each task (e. g., MAP, P@10, accuracy and macro-average F1-score) resulted in statistically significant improvements in effectiveness for little to no additional cost in efficiency.

The main findings of our research presented in this dissertation are: (1) in TM, the seminal TF-IDF and BM25 scoring functions differ only by the concave transformation and the order of composition between all the term frequency normalizations but they satisfy in the same way a set of heuristic retrieval constraints; (2) for ad hoc IR, when assessing a query term's weight in a document, rather than considering the overall term frequency of a word and then applying a concave transformation to ensure a decreasing marginal gain in relevance, one should instead consider for each word the number of distinct contexts of co-occurrence with other words so as to favor terms that appear with a lot of other terms, i. e. consider the node degree in the corresponding unweighted graph-of-words; (3) for KwE, humans tend to select as keywords not only central but also densely connected nodes in the corresponding weighted graph-of-words, property captured by reducing the graph to its main core using the concept of graph degeneracy; (4) for TC, long-distance n-grams – defined as subgraphs of unweighted graph-of-words – are more discriminative features than standard n-grams, partly because they can capture more variants of the same set of terms compared to fixed sequences of terms and therefore appear in more documents; and (5) for short document similarity or when little training data is available, smoothing the implicit delta word kernel behind the traditional n-gram linear kernel by taking account distances between terms in word embeddings results in significantly better performances.

1.2 SOFTWARE AND LIBRARIES

Most of the code pertaining to the projects presented in this dissertation has been developed in Java 1.6.0_65 and Python 2.7.6. For ad hoc IR, we extended the academic search engine `Terrier` version 3.5 (Ounis et al., 2006) and we used the Java `JUNG` library (O'Madadhain et al., 2005) for graph representation and computation. For ML, we worked with the Python `scikit-learn` library (Pedregosa et al., 2011) along with the `igraph` library (Csardi and Nepusz, 2006) for graph representation and computation. Additionally, we modestly contributed to the Python `networkx` library (Hagberg et al., 2008) after getting familiar with the concept of graph degeneracy. We used R (R Core Team, 2012) for plotting, in particular using the `ggplot2` library (Wickham, 2009), and the `Ipe` extensible drawing editor for manual drawings (Schwarzkopf, 1995).

1.3 NOTATIONS, ACRONYMS AND INDEX

Throughout the dissertation, we will be using various notations and acronyms summarized at the end of the manuscript. We refer to page 155 for an overview of the notations we adopted in linear algebra, machine learning and text mining. Generally, for mathematical variables, bold font will indicate vectors while normal font scalars. We refer to page 157 for a list of the acronyms we used, all of them being standard. We added an index of the most important topics we cover in page 159.

1.4 OUTLINE

The rest of the dissertation is organized as follows. Chapter 2 is addressed to anyone, including people less familiar with the field, who wishes to understand the foundations upon which we built our work in terms of standard text representation, various applications of text mining and evaluation procedures to test and validate our proposed approaches. Chapter 3 presents in details the adopted document representation, namely graph-of-words, our motivation behind it and the related work in the literature. Chapter 4 deals with our very first application, ad hoc IR, and we will see how graph-based term weights appear to be more meaningful for search than traditional term frequencies, in particular in terms of concave transformation. Chapter 5 covers the retention of so-called corewords from our graph-of-words as a way to extract not only central but also densely connected set(s) of keywords for single-document KwE. Chapter 6 shed some new light on the common task of TC by considering subgraphs of graph-of-words as features and the task as a graph classification problem for better prediction performances as well as taking into account word similarity to overcome some sparsity issues. Finally, Chapter 7 concludes the dissertation and mentions potential extensions of the work presented here.

2

BACKGROUND

In this chapter, we describe the foundations upon which we conducted our research. Since we were essentially interested in mining and extracting information from textual content, i. e. tackling the field of Text Mining (TM), we discuss in details (1) the **standard text representation** commonly used in the literature so that machines can "understand" or at least process natural language; (2) the various real-world **applications** we considered and their challenges; and (3) the **evaluation** procedures we used in our experiments to measure quantitatively the performances of a given method and its significant improvement (or not) over another one.

2.1 STANDARD TEXT REPRESENTATION

Machines still fail to understand natural language as we do or, more precisely, humans have failed to teach them how to do it yet. In any case, in order to perform tasks that we cannot longer do because of the ever-growing scale of the data, machines need to process automatically text and therefore they need a common way to represent text as presented in this section.

2.1.1 *Collection, document, word and vocabulary*

In TM, a dataset is usually referred to as a **collection**, denoted by $\mathcal{D}$ hereinafter, and each data point of the dataset or example is referred to as a **document**, denoted by $\boldsymbol{d}$ hereinafter. Throughout the rest of this dissertation, N will be the number of documents in the collection. A document corresponds to some piece of raw text: it can be a full Web page or just one paragraph, a tweet, a user review, etc. depending on the task at hand. For instance, when filtering spam, each email is a document; when detecting sub-events in a Twitter stream, the set of tweets from the past minute is the incoming document. In any case, a document can be seen as a sequence of words:

$$\boldsymbol{d} = (w_1, w_2, ..., w_{|\boldsymbol{d}|}) \tag{2.1}$$

where each word w_i belongs to a common **vocabulary** $\mathcal{V}$ (a. k. a. *dictionary* or *lexicon*), potentially of infinite dimension (e. g., the set of words in all Web pages – past, present and future). The process of converting a document into a sequence of words, known as **tokenization**, is out of the scope of this dissertation and assumed to be a solved problem. For European languages, other than Finnish and Hungarian, especially English, this is fairly straightforward and done by splitting on spaces and punctuation characters (with the exception of the apostrophe and the hyphen whose processing depends on the application).

2.1.2 *Bag-of-words document representation*

The unstructured nature of text makes it harder for subsequent automated tasks to use documents as such, hence the definition of a common **document representation**, i. e. a set of features to represent a document. From a Machine Learning (ML) perspective, this corresponds to the **feature extraction** step. This need for an automated translation of text into a format that the machine can understand is at the basis of the field of Natural Language Processing (NLP). The most widely adopted document representation is known as the Bag-Of-Words (BOW) (Harris, 1954). In Computer Science (CS), a bag is a multiset, i. e. a set that also stores the multiplicity of each of its elements. Hence, a document is transformed into a multiset of words and their associated frequencies $tf(\cdot)$ in the document:

$$\boldsymbol{d} = \{(w_i, tf(w_i))\}_{i=1...n'} \tag{2.2}$$

where n' is the number of unique words in the document.

2.1.2.1 *Term independence assumption*

At first glance, the loss in information seems rather important since documents like "Mary is quicker than John" and "John is quicker than Mary" would be projected to the same image in the new space. However, for numerous TM tasks, the **term independence assumption** behind the bag-of-words representation has been shown to work already quite well in practice. The main contribution of this dissertation is an alternative document representation, namely graph-of-words, that we will present in great details in Chapter 3 and that allows to take into account some word dependence, word order, word inversion and subset matching at a relatively small additional cost in efficiency (in the order of the other standard NLP pre-processing steps).

2.1.2.2 *Word versus term: a matter of dimensionality reduction*

In practice, not all words are treated equally, e. g., very frequent words – the so-called *stop words* (Manning, Raghavan, et al., 2008, p. 22) – may be removed, and some of them might be "merged" together depending on the application (e. g., *stemming* or *lemmatization*). From a ML perspective, this can be seen as a form of **dimensionality reduction** (respectively feature selection for stop word removal and lemmatization and feature transformation for stemming). We refer to Cunningham (2007) for an excellent summary on dimensionality reduction. Therefore, it is common to make a distinction between a *word* and a *term* – the latter being the processed version of the former. By abusing the definition, most of the time a bag-of-words is in fact a bag-of-terms:

$$\boldsymbol{d} = \{(t_i, tf(t_i))\}_{i=1...n} \tag{2.3}$$

where n ($< n'$) is the number of unique terms in the document.

Actually, terms do not need to correspond to single words. Indeed, for some applications like text categorization or language modeling, taking into account some word order and word dependency by encoding not only words but also sequences of words has proven to be quite successful. In the literature, a sequence of n words is referred to as an **n-gram**. Generally speaking, it is a sequence of n linguistic units (e. g., word, syllable or character) but in this dissertation, we only consider words – the alternative denomination is w-shingle for sequences of w words (Broder et al., 1997). A word is a unigram, two consecutive words form a bigram, three words a trigram and then 4-gram, 5-gram, etc. A document is therefore represented as a set of terms that are sequences of processed words and, from an ML perspective, as a **feature vector**. Even though they correspond to different data structures, a bag is still represented as a vector so as to have a common representation shared between documents, in particular when considering the collection as a document-term matrix for subsequent ML tasks.

2.1.3 *Term, document and collection frequencies*

A bag-of-words stores along with each term its **term frequency**, i. e. the number of times the term occurs in the document ($tf(\cdot, \boldsymbol{d})$). It is also informative to have that kind of information at the collection level, leading to the definition of the **collection frequency**, i. e. the total number of occurrences of a term in the collection ($cf(\cdot) = \sum_{\boldsymbol{d}} tf(\cdot, \boldsymbol{d})$) and the **document frequency**, i. e. the total number of documents of the collection in which a term occurs ($df(\cdot) = \sum_{\boldsymbol{d}} \mathbb{1}_{tf(\cdot,\boldsymbol{d})>0}$); regardless of the number of times (still positive) the term appears in each document.

The idea of using term frequencies to encode the *importance* of a term inside a document dates back to Luhn (1957) while the idea of using document frequencies as a measure of the *specificity* of a term across a collection comes from Spärck Jones (1972). Actually, it is the inverse of the latter, the so-called Inverse Document Frequency (IDF), that became popular along with the Term Frequency (TF) to form the well-known concept of TF×IDF, in particular for term weighting as we will see later on with scoring functions in ad hoc IR (Section 4.1) and also feature values in text categorization (Joachims, 1998).

The collection frequency has been somewhat less used, mostly because of its redundancy with the term frequency to some extent. Still, it is sometimes used to define stop words and also as a valid alternative to the document frequency, e. g., to compute the Maximum Likelihood Estimate (MLE) of the probability of occurrence of a term in a document given a class in text categorization when assuming a multinomial distribution for the probability of a document given a class (as opposed to a Bernoulli distribution that would require the document frequency). We will come back later on in the dissertation (Equation 6.11) on that notion when describing Naive Bayes (NB), a simple yet effective generative model commonly used in spam filtering for instance.

2.2 APPLICATIONS

In this section, we present the three main real-world applications we considered in our work: ad hoc information retrieval (e. g., Web search engine), text categorization (e. g., spam filtering) and keyword extraction (e. g., word cloud).

2.2.1 *Ad hoc information retrieval*

Information Retrieval (IR) is defined as the task of retrieving data relevant to an **information need**. The term was coined by Mooers (1951). As noted by Lancaster (1968), "an information retrieval system does not inform (i.e. change the knowledge of) the user on the subject of his inquiry. It merely informs on the existence (or non-existence) and whereabouts of documents relating to his request". The information need can either be dynamic or static, present or absent, etc. When the need is not known in advance, changes dynamically and is typically expressed explicitly through a *one-off* user-initiated natural language query, e. g., via a search engine bar or voice action, the subtask is referred to as **ad hoc information retrieval** (in Latin "ad hoc" means "for this", i. e. tailored – here to a specific need) (Manning, Raghavan, et al., 2008, p. 5). When the need is static, e. g., wanting to not receive any spam or to be updated on novel content specified via a *standing* query, the subtask is referred to as **information filtering** (Belkin and Croft, 1992) or *routing* (Hull et al., 1996). Finally, if there is no particular need, the subtask is referred to as **information browsing**, which achieves some kind of *serendipity* (Ford, 1999), e. g., Web pages skim-reading but also by extension some types of passive content recommendation and suggestion.

2.2.1.1 *Problem definition*

The problem of ad hoc IR can be defined as finding a ranked list of the most relevant documents among a collection of documents with regards to an input query $\boldsymbol{q}$. In terms of user experience, this translates into a user entering a query in a search engine bar or asking it to its smartphone with his voice and getting in return a Search Engine Results Page (SERP).

In ad hoc IR, we usually consider that all documents in the collection can be judged as either relevant or non-relevant w. r. t. $\boldsymbol{q}$, similarly to any binary classification task in ML. All documents in the results list are pertinent to some extent and the degree of pertinence should be reflected in the rank. Most of time, including in our research, we assume **document independence** and estimate separately the **relevance** r of each document $\boldsymbol{d}$ w. r. t. $\boldsymbol{q}$ using a **scoring function** $\hat{r}$ (a. k. a. *retrieval model*). This assumption is often violated in real-world applications since as the user goes through the list of results, the marginal relevance of a new document might decrease, in particular for similar results (Carbonell and Goldstein, 1998). In practice, modern (Web) search engines usually cluster together these similar results and try to propose diverse top results. In our research in ad hoc IR, we did not consider *diversity* and *novelty*

though nor any sort of diminishing marginal return that would depend on how many other relevant documents are available and how many have been already consulted (Clarke, Kolla, et al., 2008; Chandar and Carterette, 2010). We focused on the scoring aspect to propose alternative ways to assess the relevance of a document and hopefully retrieve more and rank better relevant documents. We did work on novelty detection in streams of documents and by extension in SERP but this work (Karkali, Rousseau, et al., 2013) is out of the scope of this dissertation. Note that in the real world, search engines use more than a hundred parameters in addition to relevance to score and rank web pages.

2.2.1.2 *Scoring function*

The query is usually interpreted as a (short) document itself that shares the same document representation as the documents of the collection. Therefore, the relevance can be interpreted as a measure of similarity between two documents assessed through a scoring function. Moreover, in the case of a bag-of-words representation and the term independence assumption, the document relevance can be aggregated from the relevance of each query term taken separately. It is usually defined as the sum of each query term's weights in the query, the document and the collection. How much each query term contributes to the overall relevance of the document w. r. t. the query is what makes IR a 60-year old research field.

RETRIEVAL CONSTRAINTS Fang et al. (2004, 2011) formalized the problem by proposing a set of heuristic retrieval constraints that any scoring function should satisfy. These constraints are quite intuitive, e. g., documents matching more query terms than others should be favored or longer documents should be penalized to some extent because they simply have more chance of containing a query term. Note that all these constraints should be interpreted independently of each other, considering everything else being equal. The beauty of this axiomatic approach is that it applies directly to the *vector space* model (TF-IDF (Singhal, Choi, et al., 1999)), the *probabilistic* (BM25 (Robertson, Walker, et al., 1994)), *language modeling* (Dirichlet prior (Zhai and Lafferty, 2001)) and *information-based* (SPL (Clinchant and Gaussier, 2010)) approaches and the *divergence from randomness* framework (PL2 (Amati and van Rijsbergen, 2002)), basically all the state-of-the-art methods to assess the relevance of a document w. r. t. a query in ad hoc IR. Moreover, as the research community finds additional constraints, they can benefit (almost) immediately to all these approaches. For instance, Lv and Zhai (2011b) proposed two additional constraints to prevent longer documents from being too much penalized compared to shorter documents that would contain less query terms or none at all and extended all these approaches accordingly.

SUCCESSIVE NORMALIZATIONS In practice, these constraints translate into successive normalizations to apply on the raw term frequency and document frequency. Early in the Ph.D. (Rousseau and Vazirgiannis, 2013a), we proposed **function composition** as the natural mathematical operator to combine these

normalizations and showed in particular that TF-IDF and BM25 differ only from the order of composition and the concave transformation used but meet the same retrieval goals. We will come back in great details to these notions in Section 4.1.

2.2.1.3 *Indexing vs. query time*

In practice, the task of ad hoc IR is a two-step process: (1) at indexing time, i. e. *offline*, each document of the collection is processed and indexed by the terms it possesses and (2) at query time, i. e. *online*, each query is processed so as to retrieve the relevant documents using the previously built index. In the first process, we have (almost) all the time we want to do any pre-processing needed but the space and the access time of the final index, the so-called **inverted index**, are limited (see the work of Cutting and Pedersen (1990) for techniques to optimize the access and update of the index). In the second process, time is of the essence since users want results as fast as possible and therefore nothing expensive can be performed. We refer to the survey of Zobel and Moffat (2006) for an in-depth review on how the whole pipeline works, from indexing documents to answering queries.

2.2.2 *Single-label multi-class text categorization*

Text Categorization (TC), a. k. a. **text classification**, is defined as the task of *automatically* predicting the class label, a. k. a. category of a given input textual document. It finds applications in a wide variety of domains, from news filtering and document organization to opinion mining and spam filtering. In this dissertation, we only consider the case for which we want to predict a *single* label per document but not necessarily restricted to a binary choice – hence the "single-label multi-class" denomination (multi-label text categorization is usually referred to as **topic spotting** where we want to predict probability weights for each pre-defined topic). Compared to other application domains of the general ML task of classification, its specificity lies in its high number of features, its sparse feature vectors, its multi-class scenario and its skewed category distribution. For instance, when dealing with collections of thousands of news articles, it is not uncommon to have millions of n-gram features, only a few hundreds actually present in each document, tens of class labels – some of them with thousands of articles and some others will only a few hundreds. These particularities have to be taken into account when considering feature selection, learning algorithms and evaluation metrics as well as alternative document representations like in our research.

VARIOUS APPROACHES We refer to Sebastiani (2002) for an in-depth review of the earliest works in the field and Aggarwal and Zhai (2012) for a survey of the more recent works that capitalize on additional meta-information. Most considered approaches were first used with unigrams as features and then extended to n-grams. The difference rather lies in the type of classifiers used. We note in particular the seminal work of Joachims (1998) who was the first to

propose the use of a linear Support Vector Machine (SVM) (cf. Section 6.1.2.5), a *geometric classifier*, with TF×IDF unigram features for the task. This approach is one of the standard baselines because of its simplicity yet effectiveness (unsupervised *n*-gram feature mining followed by standard supervised learning). Logistic Regression (LR) (cf. Section 6.1.2.4) and Linear Least Square Fit (LLSF) have been far less used for TC compared to other ML applications domains; we still note the works of Y. Yang and Chute (1992), Zhang and Oles (2001), and Genkin et al. (2007). Another popular approach is the use of Naive Bayes (NB) (cf. Section 6.1.1.2), a *probabilistic classifier*, and its multiple variants (McCallum and Nigam, 1998), in particular for the subtask of spam filtering (Metsis et al., 2006). Finally, k-Nearest Neighbors (kNN), an *example-based classifier*, has also been considered in the literature (Creecy et al., 1992; Y. Yang, 1994; Larkey and Croft, 1996). More recently, deep learning has started to be used for solving the task as well (Sarikaya et al., 2011).

2.2.3 *Single-document keyword extraction*

Keywords have become ubiquitous in our everyday life. We use them to look up information on the Web (e.g., via a search engine bar), to find similar articles to a blog post we are reading (e.g., using a tag cloud) or, even without realizing it, through online ads matching the content we are currently browsing (e.g., through the AdWords platform). Researchers use them when they write a paper for better indexing but also when they consult or review one to get a gist of the content before reading it. Actually, it is likely that they found or were assigned a paper because of its keywords in the first place. Traditionally, keywords have been manually chosen by the authors or some experts. However, the vast and growing majority of textual documents, especially on the Web (e. g., news articles but really any Web page), do not possess a list of pre-defined keywords to help the users determine the relevance of their content. Since manual keyword annotation is time-consuming and costly, Keyword Extraction (KwE) as an automated process naturally emerged as a research issue to satisfy that need. Its applications are numerous: *summarization*, *classification* (e. g., filtering in a stream of news or an RSS feed), *clustering* (e. g., group together similar documents), *indexing* and even *query expansion* (e. g., suggestion of additional keywords to refine a search engine query based on the ones associated with the top results).

Generally speaking, a keyword corresponds to a special *term* of the document, one that captures (some part of) its main message and that represents it well. It does not have to be a unigram and actually, in practice, authors tend to choose higher order *n*-grams to characterize a document as observed empirically (cf. Section 5.2.2.2). Therefore it is common in the literature to make a distinction between the methods that extract *keywords* (as in unigrams) (Ohsawa et al., 1998; Matsuo et al., 2001b; Mihalcea and Tarau, 2004; Litvak and Last, 2008) and those that extract *keyphrases* (*n*-grams) (Frank et al., 1999; Turney, 1999; Witten et al., 1999; Barker and Cornacchia, 2000; Hulth, 2003). Mihalcea and Tarau (2004) suggested "reconciling" the extracted keywords in keyphrases as a post-

processing step by looking in the original text for adjacent unigrams in order to compare both types of methods. However, it is not clear in the literature how to penalize a method that, given a golden bigram to extract for instance, would return part of it (unigram) or more than it (trigram). Evaluations based on exact matches would artificially lower the effectiveness and explain the low results presented in the papers that deal with extracting keyphrases. Hence, for ease of evaluation and in all fairness to all methods, we advocate for an evaluation based on keywords while still a real-world system based on keyphrases (cf. Section 5.2.3.1).

The notion of KwE is closely related to the one of **summarization**. Indeed, several techniques for document summarization rely on sentence or even keyword extraction to build the document summary. For instance, Luhn (1958), which is probably one of the earliest works in automatic summarization, capitalizes on TF to first extract the most salient keywords before using them to detect the most salient sentences. Later, the research community extended it with additional features such as IDF when dealing with a corpus of documents to summarize and then turned the task into a *supervised learning problem*. In particular, we note the seminal works of Turney (1999) with his GenEx system based on genetic algorithms and of Witten et al. (1999) with their KEA system based on Naive Bayes. We refer to the work of Nenkova and McKeown (2011) for an in-depth review on automatic summarization and by extension on KwE.

TYPES OF KEYWORD EXTRACTION The published works make several distinctions for the general task of keyword extraction: (1) **single-** (Witten et al., 1999; Turney, 1999; Hulth, 2003) vs. **multi-document** (McKeown et al., 2005; Meladianos et al., 2015a) depending on whether the input is from a single document (e. g., a research paper) or multiple ones (e. g., a stream of news), (2) **extractive** (Turney, 1999; Witten et al., 1999; Hulth, 2003) vs. **abstractive** (Blank et al., 2013) depending on whether the extracted content has to appear in the original text or not (e. g., use of a thesaurus or the closest documents to enrich the keywords), (3) **generic** (Turney, 1999; Witten et al., 1999; Hulth, 2003) vs. **query-biased** (Turpin et al., 2007; Bohne et al., 2011) vs. **update** (Karkali, Plachouras, et al., 2012) depending on whether the extracted keywords are generic or biased towards a specific need (e. g., expressed through a query) or dependent of already-known information (e. g., browsing history) and finally (4) **unsupervised** (Mihalcea and Tarau, 2004; Litvak and Last, 2008) vs. **supervised** (Turney, 1999; Witten et al., 1999; Hulth, 2003) depending on whether the extraction process involves a training part on some labeled inputs. In this dissertation, we only consider the task of *generic extractive single-document keyword extraction* – basically, given a document, what are its keywords?

IR OR TEXT CLASSIFICATION PROBLEM? The task of keyword extraction resembles a problem of information filtering in the sense that it also tries to alleviate the *information overload* that users are subject to. Similarly to filtering out unwanted pieces of news from an RSS feed, extracting keywords from a

document gives the user the interesting bits. However, it is a retrieval problem at the document level and not at the collection level in the sense that we want to retrieve the terms that are the most relevant to our *summarization need* rather than the documents that are the most relevant to our information need.

The task of keyword extraction can also be seen as a binary classification problem where terms of a document are either keywords or not. It is the approach followed by the supervised methods aforementioned (Turney, 1999; Witten et al., 1999; Hulth, 2003). Compared to the task of text categorization previously introduced, it is a classification problem at the document level and not at the collection level though. Indeed, the terms – regardless of the documents they belong – are the actual examples to classify and no longer the features. Moreover, the number of considered features per term is actually quite small: its term frequency, its inverse document frequency, the index of its first relative occurrence in the document and maybe some Part-Of-Speech (POS) information.

2.3 EVALUATION

Now that we have concrete applications on which we want the machines to perform well, or at least as good as the humans in terms of quality, we need to define ways to evaluate their performance. We assume a classification task for which we have **ground truth** data, a. k. a. a gold standard for the expected outcome. For instance, it could be the set of truly relevant documents for a query in ad hoc IR, the true class labels in text categorization or the set of manually annotated keywords associated to a document in keyword extraction. In practice, a lot of these tasks have binary outcomes (e. g., relevant/non-relevant, spam/non-spam or keyword/non-keyword) and we refer to one of the two outcomes as the "positive", usually the one corresponding to the task (the other alternative being the default or the "negative" one). Even for other multi-class tasks such as text categorization, we can always successively consider one class as positive and the rest as negative and then average the metrics as we will see later on.

2.3.1 *Types of predictions and confusion matrix*

In this context, given a binary classification task with positive and negative examples, we want to evaluate how well a system fares in terms of **prediction effectiveness**. There are four types of possible predictions (Swets, 1963):

(1) True Positive (TP) – the system correctly predicts a positive class for a positive example, resulting in a **hit**

(2) True Negative (TN) – the system correctly predicts a negative class for a negative example, resulting in a **correct rejection**

(3) False Positive (FP) – the system wrongly predicts a positive class for a negative example, resulting in a **false alarm**

(4) False Negative (FN) – the system wrongly predicts a negative class for a positive example, resulting in a **miss**

This information is usually presented in a 2x2 **confusion matrix**, a. k. a. contingency table, as follows:

		Predicted class	
		positive	negative
Actual class	positive	TP	FN
	negative	FP	TN

Table 2.1 – Confusion matrix for a binary classification task.

We note in particular that there are in total TP + TN + FP + FN examples, TP + FN being actual positive examples and TP + FP being predicted positive examples.

2.3.2 *Simple evaluation metrics*

Based on these ratios, we can first define point estimate metrics to measure quantitatively the performances in prediction of a single system for a single task on a single dataset.

2.3.2.1 *Accuracy*

Intuitively, we would want a system that maximizes TP and TN, i. e. to judge a system on the fraction of its predictions that are correct, which is referred to as Accuracy (Acc) and defined as:

$$\text{Acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.4}$$

ILLUSION Accuracy is a very common metric for classification tasks in ML but far less in Text Mining because of the imbalanced class distribution in many scenarios that lead to artificially high accuracies. Indeed, consider the extreme case where 99.9% of the examples are negative and a system that always predicts the negative class, then its accuracy is of 99.9%! This phenomenon, known as the **accuracy paradox**, happens more than one would think, in particular in the applications we are interested in. For instance, in ad hoc IR, for a given query, almost all the Web pages are non-relevant and therefore a search engine that never returns any results would have a very high accuracy. Same goes for spam emails in text categorization and non-keyword terms in keyword extraction. Therefore, the research community has defined other metrics in addition to accuracy to overcome this pitfall.

2.3.2.2 *Precision and recall*

Focusing on the positive class, we have seen that there are two kinds of errors: misses and false alarms, which most of time act as opposite forces – the more positive predictions the system makes, the less misses but the more false alarms and vice-versa (Cleverdon, 1972; Heine, 1973; Bookstein, 1974).

PRECISION In spam filtering, a false alarm is detrimental since it means tagging a real email as junk and therefore hiding it from the user, which translates into wanting to reduce FP w. r. t. TP, leading to the definition of a metric known as Precision (P):

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{2.5}$$

basically the fraction of correct predictions restricted to the positive class (which alleviates the pitfall of accuracy when the negative examples belong to the majority class).

RECALL Conversely, consider a lawyer that uses a specialized search engine to retrieve all materials relevant to its case – finding a precedent is so crucial that no miss can be tolerated, which translates into wanting to reduce FN w. r. t. TP, leading to the definition of a metric known as Recall (R), a. k. a. *sensitivity*:

$$\mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{2.6}$$

basically the fraction of positive examples captured by the system.

TERMINOLOGY The notions of precision and recall were first introduced by Kent et al. (1955) for IR although precision was referred to as *pertinency* then *relevance ratio* in the literature of the ASLIB Cranfield project (Cleverdon and Mills, 1963) and the term itself only appeared later in (Swets, 1963) before gaining widespread acceptance in the IR community and then the TM and ML communities.

2.3.2.3 *F-measure and F1-score*

As aforementioned, depending on the task at hand and the context, sometimes the sole precision matters or only the recall is of interest but in general both precision and recall are important. However, it is hard to compare two systems where one has a better precision and the other one a better recall. Therefore, the research community devised a **combined measure of precision and recall**. van Rijsbergen (1979, pp. 173–176) proposed a measure of effectiveness E to capture the trade-off between precision and recall and it is actually its complement F ($= 1 - E$) that became popular, known as F-measure (F_β):

$$\mathrm{F}_\beta = \frac{1}{\alpha/\mathrm{P} + (1-\alpha)/\mathrm{R}} = (1+\beta^2)\frac{\mathrm{P} \times \mathrm{R}}{\beta^2\mathrm{P} + \mathrm{R}}, \quad \alpha = \frac{1}{1+\beta^2} \tag{2.7}$$

α controls the relative weight we want to give to precision w. r. t. recall. When we consider that precision and recall are of equal importance (i. e. $\alpha = 1/2,\ \beta = 1$), the metric is usually referred to as F1-score (F_1):

$$F_1 = 2\,\frac{P \times R}{P + R} \tag{2.8}$$

which corresponds to the **harmonic mean** of precision and recall. Note that if the precision and recall are equal then a fortiori they are also equal to the F1-score.

INTERPRETATION Van Rijsbergen provides an extensive theoretical discussion, which shows how adopting a principle of decreasing marginal relevance (at some point a user will be unwilling to sacrifice a unit of precision for an added unit of recall) leads to the harmonic mean being the appropriate method for combining precision and recall rather than the arithmetic mean for instance. Intuitively, the harmonic mean is suited for averaging ratios of constant numerators (TP in the case of P and R) while the arithmetic mean is suited for averaging ratios of constant denominators. Think for instance of average speed (a ratio of distance over time) over trips of equal distance (harmonic mean) vs. of equal duration (arithmetic mean).

2.3.3 *Averaged evaluation metrics*

To assess the robustness of a system, it is often better to test it under various circumstances, be it a range of parameters' values or multiple categories for instance.

2.3.3.1 *Precision/recall curve and average precision*

Often, the classification model behind the system has one or more parameters that can control how much precision/recall we will get, e. g., a threshold on the probability of the example being positive given the data or the number of keywords to extract. By varying that parameter, we get multiple pairs (R, P) that we can plot altogether as a Precision/Recall (P/R) curve. The further away from the origin, the better a P/R curve is (upper right corner). Therefore, if the curve of a system A is consistently over the curve of a system B then we can say that system A is better w. r. t. that metric (cf. Figure 5.4 for an illustration).

AVERAGE PRECISION If the superiority of one system over the other is less clear (crossing curves) then usually the Area Under the Curve (AUC) is preferred and used as a metric to compare systems. It corresponds to the Average Precision (AP) and is very common in ranked retrieval problems such as ad hoc IR where we consider a ranked sequence of documents and we compute a precision at each relevant document, considering all the documents ranked above as retrieved – basically a precision at a fixed level of recall. Note that in

practice, we stop considering documents after a fixed rank (typically 1000 in Text REtrieval Conference (TREC) settings, maybe much lower in a live system).

PRECISION AT K Alternatively, for tasks such as Web search, it is more important to know how many documents are relevant among the top 10 results than how many documents are relevant among the top results up to the 10^{th} relevant document. Hence, another common way of measuring effectiveness in ad hoc IR is to compute precision at a fixed Document Cut-off Value (DCV) (Hull, 1993), leading to the definition of precision at k and more specifically Precision at 10 (P@10), which corresponds to the precision of a search engine on its usual first SERP and thus the precision that the user perceives.

INTERPRETATIONS AP and P@10 have substantial different interpretations: precision at a fixed recall vs. at a fixed document cut-off value, i. e. for a given **performance** vs. for a given **user effort**, and which one is better depends on the application, e. g., patent search vs. Web search. In the literature, when proposing a novel retrieval model, it is common to report both so as to show which applications are better suited for the approach.

PROGRAM In our experiments, we used the standard program `trec_eval` [1] for evaluating IR systems, which computes many measures of ranked retrieval effectiveness including the ones aforementioned. It was developed by Chris Buckley for the TREC evaluations.

RELATIONS WITH OTHER CURVES Davis and Goadrich (2006) explained why the linear interpolation between the points of a P/R curve is incorrect and also its relation with the Receiver Operating Characteristic (ROC) curve (Green and Swets, 1966) used in signal processing and detection theory that plots the recall against the fallout (false alarm probability). The further away from the (0, 1) point, the better a ROC curve is (upper left corner). Additionally, Martin et al. (1997) proposed the Detection Error Trade-off (DET) curve that plots the $\mathbb{1}-$recall (miss probability) against the fallout. The closer from the origin, the better a DET curve is (lower left corner). ROC and DET curves are generally more frequent in detection tasks as opposed to decision tasks where P/R curves are preferred.

2.3.3.2 *Micro-averaged vs. macro-averaged metrics*

Precision, recall and F1-score are defined for a given class of a given classification task. We have already seen that averaging precision over multiple recall values or ranks is closer to the user's perception of the system. For generalization purposes, it can also be useful to assess the **average prediction effectiveness over multiple classes or tasks**, e. g., all categories in text catego-

1. http://trec.nist.gov/trec_eval/

rization or multiple queries in ad hoc IR. By multiple tasks, we do not mean switching from keyword extraction to text categorization but rather either using a different dataset (e. g., a different document in single-document keyword extraction) or changing the goal (e. g., a different query in ad hoc IR). Indeed, it seems rather intuitive to average the performances over multiple queries to get a better estimate of a search engine's effectiveness.

TWO APPROACHES There are two ways to average a metric: by **micro-averaging** it, i. e. by pooling per-example decisions across classes or tasks and then compute the metric on the pooled contingency table, and by **macro-averaging** it, i. e. by taking the arithmetic mean of the metric over all classes or tasks. The differences between the two methods can be large. Macro-averaging gives equal weight to each class/task, whereas micro-averaging gives equal weight to each per-example decision. Because the F1-score ignores true negatives and its magnitude is mostly determined by the number of true positives, large classes/tasks dominate small classes/tasks in micro-averaging.

MEAN AVERAGE PRECISION Therefore, for multiple tasks, it is more common to use macro-averaged metrics since you do not want to favor for instance documents with more keywords or queries with more relevant documents (assuming the tasks are indeed independent). This is why in ad hoc IR another standard metric is the macro-averaged AP known as Mean Average Precision (MAP) that has no useful micro-averaged equivalent. Conversely, for multi-class classification, micro-averaged metrics can give an estimate of how the system fares generally (performing badly on a class with only a few examples would artificially lower the macro-averaged metric but not the micro-averaged one).

EQUATIONS Assuming n classes or tasks and i the index of a particular class or task (hence TP_i being the number of true positives for class or task i for instance), micro-average and macro-average precisions, recalls and F1-scores are defined as follows:

$$\text{micro-avg } \mathrm{P} = \frac{\sum_i \mathrm{TP}_i}{\sum_i \mathrm{TP}_i + \sum_i \mathrm{FP}_i} \tag{2.9}$$

$$\text{micro-avg } \mathrm{R} = \frac{\sum_i \mathrm{TP}_i}{\sum_i \mathrm{TP}_i + \sum_i \mathrm{FN}_i} \tag{2.10}$$

$$\text{micro-avg } \mathrm{F}_1 = 2\,\frac{\text{micro-avg } \mathrm{P} \times \text{micro-avg } \mathrm{R}}{\text{micro-avg } \mathrm{P} + \text{micro-avg } \mathrm{R}} \tag{2.11}$$

$$\text{macro-avg } \mathrm{P} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{P}_i \tag{2.12}$$

$$\text{macro-avg } \mathrm{R} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{R}_i \tag{2.13}$$

$$\text{macro-avg } \mathrm{F}_1 = \frac{1}{n}\sum_{i=1}^{n} \mathrm{F}_{1i} \tag{2.14}$$

PITFALL In particular, the macro-average F1-score is *not* the harmonic mean of the macro-average precision and recall – this is a common mistake seen in some papers (T.-Y. Wang and Chiang, 2007; Moreira-Matias et al., 2012) and other forums like Kaggle competitions [2] and Stack Exchange answers [3]. Indeed, consider a binary classification task with 1M positive examples and 1 negative example, and a system that always predicts the negative class except for one positive example for which it correctly predicts its class. This is a pretty bad system and the metric should reflect that. For the positive class, the precision equals 1 and the recall is close to 0 (10^{-6}) so an F1-score of 0 (10^{-6}). For the negative class, the precision is close to 0 (10^{-6}) and the recall equals 1 so an F1-score of 0 (10^{-6}). Macro-average precision and recall are both of 0.5 (arithmetic mean between 0 and 1) *but* the macro-average F1-score is 0 (arithmetic mean between 0 and 0) and not 0.5 (harmonic mean between 0.5 and 0.5). The counter-intuitive F1-score below both the precision and recall is actually a good indicator of an unstable system, which in this case has very high precision/low recall for one class and very high recall/low precision for the other one.

RELATION WITH ACCURACY Note that in single-label classification (binary or multi-class), the micro-average precision, the micro-average recall and a fortiori the micro-average F1-score are all equal to the accuracy since the total number of false positives is equal to the total number of false negatives as noted by Manning, Raghavan, et al. (2008, p. 281). Indeed, **a miss for a class is a false alarm for another one**. This is not true in general for multi-label classification (except if we fix the number of labels per document and all labels are predicted the same number of times) or when averaging over multiple tasks (e. g., a non-relevant document for one query is not bound to be relevant for another query or a non-keyword term for one document does not have to be a keyword in another document).

2.3.4 *Statistical significance of improvement*

When comparing two systems (let's say A and B, in our case a novel method (A) w. r. t. to a baseline (B)), we need a method to decide whether system A performs better than system B w. r. t. a chosen evaluation metric. Obviously, we assume that there is an improvement, i. e. a positive difference in evaluation scores in favor of system A. We then need to quantify the improvement and decide if we consider it meaningful or simply due to chance. This decision could be made using an informal rule, e. g., in the early days of IR, Spärck Jones and Bates (1977, p. A25) deemed an improvement of at least 5% significant ("noticeable" by the user) and of at least 10% very significant ("material"). It is probably the only way to make such a decision for an individual class or task since we only have a single point estimate for the metric.

2. https://www.kaggle.com/c/lshtc/details/evaluation
3. http://stats.stackexchange.com/q/44261

SIGNIFICANCE TESTING However, when evaluation measures are averaged over a number of classes or tasks, we can obtain an estimate of the error with that measure and **statistical significance testing** becomes applicable. Intuitively, the decision to consider an improvement significant is strengthened (1) when the difference values are relatively high; or (2) when these values are, more or less, always in favor of one system; and (3) when the sample size grows (Hull, 1993). In the case of ad hoc IR, given a set of queries large enough, we would prefer for instance a system that achieves higher AP or P@10 than the other one on most queries. Indeed, the overall average could hide the fact that on most queries the system performs actually worse (by only a little) but on some queries it performs way better (hence a higher macro-average but not a significant one).

NULL HYPOTHESIS The preliminary assumption, or **null hypothesis** H_0, is that the systems are equivalent in terms of performances (i. e. the observed improvement is due to chance). The significance test will attempt to disprove this hypothesis by determining a p-value, i. e. a measurement of the probability that the observed improvement could have occurred by chance. Under H_0, each test computes a statistic T and calculates the achieved significance level of this test, which is the probability of observing a value at least as extreme as T when H_0 holds: $p = \mathbb{P}(X \geq T)$ where X is the random variable for the statistic and $\mathbb{P}$ the assumed probability distribution, typically the standard normal distribution. If this probability is less than a pre-defined *significance level* α (typically 0.05 or 0.01 to be more conservative), which corresponds to the Type I error rate we are willing to accept, we may reject the null hypothesis with $1 - \alpha$ (i. e. 95% or 99%) confidence and conclude that the two systems are significantly different, i. e. that the **alternative hypothesis is at least more likely than the null hypothesis**. The *effect size* on the other hand measures the magnitude of the improvement and relates in that sense more to the idea of noticeable and material changes from Spärck Jones and Bates (1977).

DISTRIBUTION ASSUMPTIONS Significance tests fall into a number of different categories, in particular *parametric* vs. *non-parametric* (a. k. a. distribution-free) depending on whether we make specific assumptions about the distribution of the measurements and their errors. We considered in our experiments two **paired tests**: **sign test** (Conover, 1980, pp. 122–129) and **Student's t-test** (Conover, 1980, pp. 290–292). Paired tests (as opposed to independent tests) are the most suitable for comparing values produced by two systems for the same set of independent observations (in our case queries or documents at the macro-level and classification decisions at the micro-level) and also because there is no risk of "contamination" between the two automated systems (as opposed to let's say having each patient taking the drug and the placebo). These tests are from far the most used in TM and IR (Hull, 1993; Savoy, 1997; Y. Yang and X. Liu, 1999).

MICRO- VS. MACRO-AVERAGED METRICS We considered micro- and macro-averaged metrics, which means than when comparing two systems, we can perform a pairwise comparison of respectively the per-example binary decisions and the per-class/per-task metrics. Per-example binary decisions (e. g., relevant/non-relevant, keyword/non-keyword or spam/non-spam) can only tell us if the two systems differ and if so, which one was right. Per-class/per-task metric (e. g., F1-score for a given category or AP for a given query) can also help us quantify the magnitude of the difference.

SIGN TEST VS. T-TEST The sign test looks only at which system performed better: if one system performs better than the other far more frequently than would be expected on average, then this is strong evidence that it is superior. The sign test can be used for both micro- and macro-averaged metrics. The Student's t-test compares the magnitude of the difference between systems to the variation among the differences. If the average difference is large compared to its standard error, then the systems are significantly different. The t-test assumes that the difference follows the normal distribution, but it often performs well even when this assumption is violated (Hull, 1993; Savoy, 1997; Y. Yang and X. Liu, 1999). Since we are considering binary classification decisions, the t-test only makes sense for macro-averaged metrics where we can measure the magnitude of the difference between real-valued metrics.

NOTATIONS Let's denote by a_i (resp. b_i) the measure of success for system A (resp. B) on the i^{th} decision (micro-level) or i^{th} task (macro-level) and n the number of times that a_i and b_i differ (therefore ties reduce the number of **trials**). For instance, a_i is equal to 1 at the micro-level if system A correctly classified the i^{th} example (0 otherwise) and to 0.82 at the macro-level if system A achieved that precision or whatever metric is of interest on the i^{th} class or query.

SIGN TEST For the sign test, we consider k to be the number of times that $a_i > b_i$ (the number of **successes**). H_0 corresponds then to k following a binomial distribution $Bin(n, p = 0.5)$ – we would expect on average k to be $0.5n$, i. e. half the times system A and B differ, system A is better than system B. Under H_0, the one-sided p-value can thus be expressed as:

$$p = \mathbb{P}(X \geq k) = \sum_{i=k}^{n} \binom{n}{i} \times 0.5^n \tag{2.15}$$

For $n > 12$, the one-sided p-value can be approximately computed using the standard normal distribution and the complementary of its Cumulative Distribution Function (CDF):

$$p = \mathbb{P}(Z \geq T) = 1 - \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{T} e^{-t^2/2} \, dt, \quad T = \frac{k - 0.5n}{0.5\sqrt{n}} \tag{2.16}$$

T corresponds to the standard score (a. k. a. z-score) of k ($= {}^{(k-\mu)}/_{\sigma}$) since we want to measure where the statistic falls on the *standard* normal distribution.

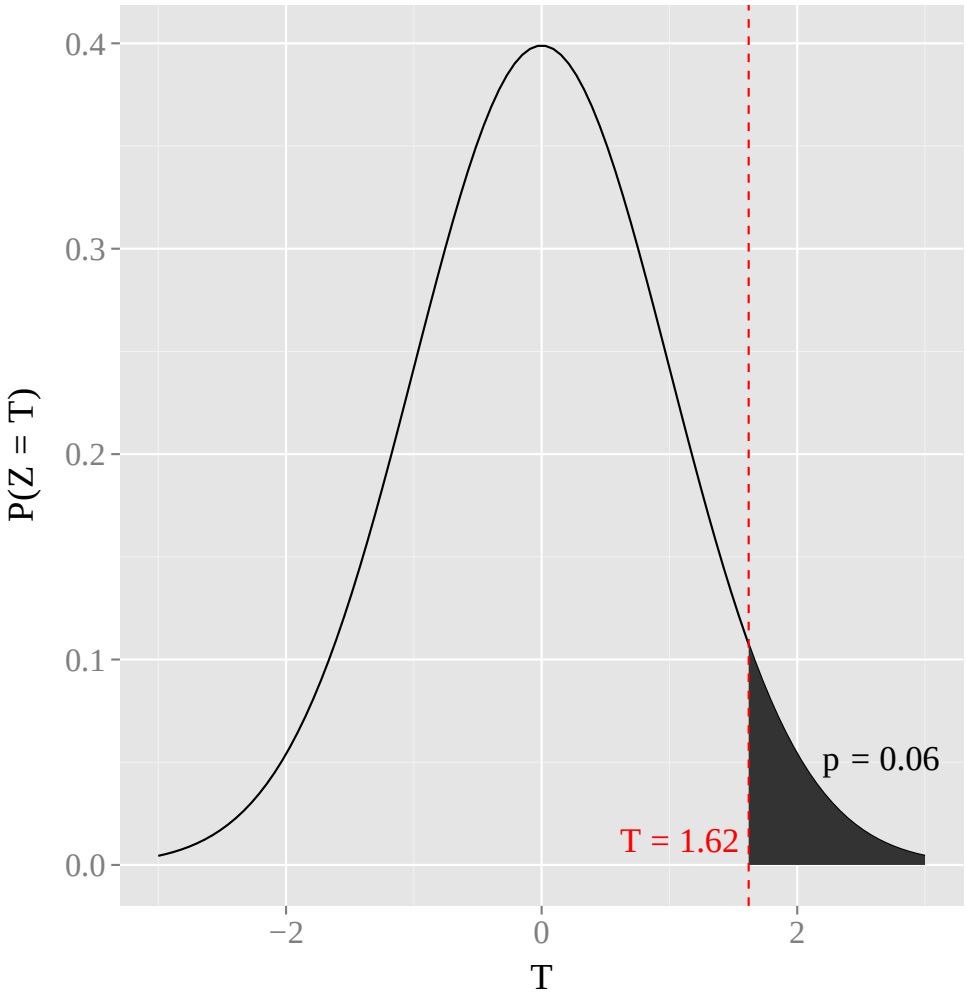


Figure 2.1 – Standard normal distribution – example of T value (dashed line) and the corresponding p-value $= \mathbb{P}(Z \geq T)$ (black shaded area).

And for a binomial distribution $Bin(n, p)$, the mean is np $(= 0.5n)$ and the standard deviation $\sqrt{np(1-p)}$ $(= 0.5\sqrt{n})$.

ILLUSTRATION Figure 2.1 above illustrates this process for $n > 12$. In this example, a T statistic of 1.62 yields a p-value of 0.06, which corresponds to the value of the black shaded area under the curve, i. e. the probability of observing a value at least as extreme as T under H_0.

EXAMPLE 1 Consider a binary classification task with 1000 examples (500 positive and 500 negative), a system B that always predicts the negative class (accuracy of 50%) and a system A that achieves 55% in accuracy. System A performs better than system B, by 10%, but is it significant? As noted in Section 2.3.3.2, accuracy corresponds to micro-averaged F1-score so we should compare per-example binary decisions. (1) Let's say that system A always predicts the positive class except for 50 negative examples (hence the additional

5% in accuracy compared to B). In that scenario, $n = 950$ and $k = 500$ leading to a T value of 1.62 and a p-value of 0.06 (as shown on Figure 2.1), value too high to declare system A significantly better than system B (assuming $\alpha = 0.05$). (2) Let's now say that system A always predicts the negative class like system B except for 50 positive examples. In that scenario, $n = 50$ and $k = 50$ leading to a T value of 7.07 and a p-value of 10^{-15}, low enough to declare system A significantly better than system B! Intuitively, in the second scenario, system A is objectively strictly better than system B because it does not make mistakes that system B does not and it makes less mistakes than system B does. In the first scenario, system A makes a lot of mistakes that system B does not and therefore the 50 mistakes that system A does not compared to B is not as significant. Indeed, in terms of significance, the difference between 0 and 50 is much higher than between 450 and 500.

STUDENT'S T-TEST For the t-test, we consider $\mathbf{d}$ the set of differences $d_i = a_i - b_i$ for $i = 1...n$ between the two systems on each macro-level metric value (e. g., in text categorization the difference in F1-score between the two systems on each category or in ad hoc IR the difference in AP on each query). We denote by $\overline{\mathbf{d}}$ the arithmetic sample mean of $\mathbf{d}$ (which is the unbiased estimate of the population mean) and $s.e.(\overline{\mathbf{d}})$ $(=s_{n-1}/\sqrt{n})$ the standard error of $\overline{\mathbf{d}}$ where s_{n-1} $(= \sqrt{\frac{1}{n-1}\sum_i (d_i - \overline{\mathbf{d}})^2})$ is the unbiased estimate of the population variance of $\mathbf{d}$. H_0 corresponds then to each d_i following a normal distribution of mean 0. Under H_0, the one-sided p-value can be computed using the Student's t-distribution with $\nu = n - 1$ degrees of freedom and the complementary of its CDF:

$$p = \mathbb{P}(X \geq T) = \frac{1}{2} I_{\frac{\nu}{\nu+T^2}}\left(\frac{\nu}{2}, \frac{1}{2}\right), \quad T = \frac{\overline{\mathbf{d}}}{s.e.(\overline{\mathbf{d}})} \tag{2.17}$$

where I is the regularized incomplete beta function (Pearson, 1968, p. ix). For $n > 40$, similarly to the sign test, the p-value can be approximately computed using the standard normal distribution (as the degrees of freedom increase, the Student's t-distribution gets closer to the standard normal distribution).

EXAMPLE 2 Consider an ad hoc IR task with $n = 1000$ queries, a system B that always yields for any given query an AP of 0.5 (MAP of 0.5) and a system A that achieves a MAP of 0.55. System A performs better than system B, by 10% ($\overline{\mathbf{d}} = 0.05$), but is it significant? (1) Let's say that we have $s_{n-1} = 0.98$ meaning a residual sum of squares of 950 over the 1000 queries – for instance a difference in AP of almost 1 on each query one way or the other, which is quite substantial. The corresponding T value would be of 1.62 and thus a p-value of 0.06, probability too high to reject the null hypothesis. (2) Let's now say that we have $s_{n-1} = 0.22$ meaning a residual sum of squares of 50 over the 1000 queries and leading to a T value of 7.07 and a p-value of 10^{-15}, probability low enough to reject the null hypothesis and consider system A significantly better than system B. Intuitively, system A is on average better than system B and in the second scenario consistently since the residual sum of squares is rather low.

The variance in the first scenario is way higher hence a non-significant overall improvement.

PROGRAM For all our experiments, we used the R implementation for the t-test (`t.test {stats}`) and we implemented the sign test using R's binomial test (`binom.test {stats}`) with "greater" as value for the *alternative* parameter (default is "two-sided"). We recommend the G*Power tool (Faul et al., 2007) for understanding and visualizing the various statistical tests and the statistical power we are assuming based on the chosen significance level α, sample size and obtained effect size.

ADDITIONAL TESTS The **Wilcoxon signed-rank test** (Conover, 1980, pp. 280–288) has also been considered as an alternative non-parametric paired significance test by the IR research community. It replaces each difference with the rank of its absolute value among all differences. These ranks are then multiplied by the sign of the difference and the sum of the ranks of each group is compared to its expected value under the assumption that the two groups are equal. It was however discarded by the community (Sanderson and Zobel, 2005; Smucker et al., 2007) and thus not used in our experiments even though a recent paper advocates its use again (Urbano et al., 2013). In ML, another paired significance test used for binary classification is **McNemar's chi-squared test** (McNemar, 1947). Using the same notation as for the sign test, its T statistic is $(2k-n)^2/n$ and the p-value is computed using the χ^2-distribution with 1 degree of freedom. Just like the sign test, it can be used for both micro- and macro-averaged results.

STATISTICAL REFORM Sakai (2014) recently advocated for a true statistical reform in the field of ad hoc IR, recognizing the efforts made in the past decade in terms of reporting p-values and statistical significance for MAP and P@10 results but also asking the IR research community to go further by reporting effect sizes and confidence intervals as well, following other communities from the fields of medicine, psychology and ecology (Fidler et al., 2004). Surprisingly, significance testing is far less present in NLP and ML papers when reporting accuracy and macro-average F1-score (assuming enough classes or tasks) but we think these fields should also adhere to these standards and start undertaking their own statistical reform. Even more recently, Leek and Peng (2015) stated that "p-values are just the tip of the iceberg" when it comes to assess the validity of a whole data science pipeline, from the experimental design to the final predictions.

3

CHALLENGING THE HISTORICAL BAG-OF-WORDS

In this chapter, we introduce graph-of-words as an alternative document representation to the historical bag-of-words. Motivated by the idea that neighboring words in a document share some special relationship worth capturing for subsequent text mining applications, we explored a way of encoding word dependencies while relaxing the exact matching condition behind document similarity based on n-grams, the natural extension of words to sequences of words. We trace first the history of works that led from simple multiset representations of text to graph-based ones. We then present our approach in details, its variants and various properties and their interpretations. Actual applications are left to the next three chapters.

3.1 MOTIVATION

Harris (1954) stated that words that occur in the same contexts tend to have similar meanings, property known as the **distributional hypothesis**. Firth (1957, p. 179) then popularized the underlying idea that a word is characterized by "the company it keeps", which was tested empirically by Rubenstein and Goodenough (1965). It is for instance at the heart of the Vector Space Model (VSM) from Salton, Wong, et al. (1975) for vectorized *document similarity* and of the Latent Semantic Indexing (LSI) technique introduced by Deerwester et al. (1990) for *topic modeling*, basically Singular Value Decomposition (SVD) on the document-term matrix. In these early works, the context in question was the document, which naturally gave birth to the (unigram) Bag-Of-Words (BOW) representation for text: regardless of their respective positions in the document, all the words of a document share some common relationship.

3.1.1 *Going beyond unigram bag-of-words*

Nevertheless, as the text collections became heterogeneous, especially in document length and vocabulary, the research community started exploring additional representations and more fine-grained contexts of co-occurrence than the full document, challenging the well-established unigram bag-of-words.

LONG DOCUMENTS On one hand, as the document length increases, the relationship between words at the beginning and at the end of the document might start to fade. Therefore, in more recent works, a second context has been considered so as to better capture the interactions between neighboring

words, typically over a window of 2 (bigram) to 10 words (10-gram), which is commonly referred to as a **phrase**. For instance, somewhat recently, *word2vec* [1], a 300-dimensional word embedding, was trained over billions of such phrases extracted from Google News using either the Skip-gram or the Continuous Bag-Of-Words model (Mikolov, Chen, et al., 2013), predicting respectively surrounding words given the current word and the current word based on the context. Classically, people have been considering **n-grams**, i. e. sequences of n words, rather than simple unigrams to try to capture some word dependency and word order at the phrase level, be it for (1) Text Categorization (TC) with additional features (Cavnar and Trenkle, 1994; Fürnkranz, 1998); (2) Keyword Extraction (KwE) considering keyphrases (Witten et al., 1999; Turney, 1999); (3) ad hoc Information Retrieval (IR) with phrasal indexing (Salton, C.-S. Yang, et al., 1975) or proximity search (van Rijsbergen, 1977; Tao and Zhai, 2007); (4) statistical machine translation with phrase-based approaches (Koehn et al., 2003); or (5) language modeling with higher order n-gram models (Brown, deSouza, et al., 1992).

Going further, Mitra et al. (1997) opposed **syntactic phrases** to *statistical* phrases (i. e. based on co-occurrences) for ad hoc IR as used by Fagan (1987), which allow for gaps between words in the original text (e. g., skipping adjectives) because the sequences are considered over dependency tree sentence representations for instance. Goodman (2001) and Guthrie et al. (2006) discussed how subsequences of initial n-grams, so-called **skip grams**, can help overcome the sparsity issue in MLE estimation. Similarly, Bassiou and Kotropoulos (2010) introduced the notion of **long-distance bigrams** to enhance word clustering. Through these extensions, the research community still wanted to capture the word dependencies but the exact matching on the sequences might be to rigid to improve document similarity and therefore should be relaxed, considering subsequences like in these works or better, considering subsets to capture some word inversion for instance; at least that was our initial thought behind a graph representation. Indeed, as we will see later on in Section 3.2.3, we will be able to interpret subgraphs of our graph-of-words as sets of co-occurring words that we will refer to as *long-distance n-grams*.

SHORT DOCUMENTS On the other hand, with very short documents such as tweets, queries or comments that have become numerous thanks to the rapid growth of the Web 2.0, there can be some **sparsity issue** due for instance to word synonymy and some documents with similar meaning will be found to have no overlapping words and therefore low similarity if only looking at exact "surface matching" as explained by F. Wang et al. (2014). This notion of *surface* is to be opposed to the *underlying concepts* that could be used to compare words instead of the naive exact matching. We tackled this particular issue during the Ph.D. (J. Kim et al., 2015), capitalizing on word embeddings to define **polynomial word kernels** as opposed to the implicit *delta* word kernels behind the traditional linear document kernel in the n-gram feature space. We will come back to this work in Section 6.4.

1. https://code.google.com/p/word2vec/

HARD VS. SOFT MATCHING Long or short documents, in any case, the idea is to relax the definition of a match between pairs of n-grams while still capturing some word dependency at the phrase level, be it by comparing sets of words instead of sequences or using some word similarity (and by extension phrase similarity) to soften the matching. In this dissertation, we will deal with both aspects focusing more on the first one since the statistical graph-based representation only helped for documents with at least a hundred words. Indeed, there are little word repetitions in sentences in practice and therefore *statistical* graphs-of-words lose in effectiveness as opposed to similar representations that encode in addition *syntax* or *semantics* for instance. This is actually why we represented *multiple* documents by a single statistical graph-of-words when considering collections of tweets in our research (Meladianos et al., 2015a).

CO-OCCURRENCE VS. COLLOCATION So far, we have only been discussing **co-occurrences** of words within a document or a phrase. But sometimes phrases have more meaning than just the association of each word's meaning, e. g., "weapons of mass destruction". In the literature (Firth, 1957, p. 181; Sinclair, 1991, p. 115-116; Manning and Schütze, 1999, sec. 5.5), this corresponds to the distinction made with **collocations** that are basically *significant* co-occurrences. Actually, Manning and Schütze (1999, p. 153–154) proposed the Student's t-test (cf. Section 2.3.4) to separate meaningful co-occurrences from those due to chance while Church and Hanks (1990) capitalized on the mutual information between pairs of words. Finally, we ought to mention the several works of Smadja (1990; 1991; Apr. 1991; Mar. 1993) who dedicated his Ph.D. dissertation to filtering n-grams extracted from large corpora in order to get collocations from mean and variance analysis. In our work, we considered all co-occurrences assuming that the relevant subsequent processing steps will take care of whatever discrimination is needed, e. g., through some feature selection procedures. This also alleviates the need of using language-specific large collections of documents and favors off-the-shelf solutions.

3.1.2 *Graph-based text representations*

In this section, we focus the literature review on graph representations of text that are to some extent similar to the one we adopted and that we present in the next section. We refer to the work of Blanco and Lioma (2012, sec. 2.2) and references therein for a broader review on the subject as well as the more recent survey from Sonawane and Kulkarni (2014). We also added some references of our own that were missing or that are newer. The earliest work we found and surprisingly rarely cited was the one from Ohsawa et al. (1998).

REPRESENTATIONS Briefly, a graph consists of a set of nodes or vertices related to one another via edges. In the case of text, a vertex corresponds to some meaningful linguistic unit: (1) a **paragraph** (Balinsky et al., 2011); (2) a **sentence** (Erkan and Radev, 2004; Mihalcea and Tarau, 2004); (3) a **phrase** (Xie,

2005; Velikovich et al., 2010); (4) a **word** (Ohsawa et al., 1998; Ferrer i Cancho and Solé, 2001; Schenker et al., 2005); (5) a **syllable** (Soares et al., 2005); or even (6) a **character** (Crochemore and Vérin, 1997; Giannakopoulos et al., 2008). Actually, in some works, some nodes corresponded to words and others to sentences, forming a bipartite graph (Zha, 2002) or not (Wan, J. Yang, et al., 2007). We voluntarily omit works where nodes are concepts or topics – we were only interested in observable characteristics of a document in our research. An edge corresponds to some relationship between two vertices, which can be: (1) **statistical**, e. g., simple co-occurrences (Matsuo et al., 2001b) or collocations (Ferret, 2002; Bordag et al., 2003) in a window (Dorogovtsev and Mendes, 2001), a sentence (Ohsawa et al., 1998), the full document (Sandler et al., 2009) or in the definition of one of the two words from a dictionary (Blondel and Senellart, 2002); (2) **syntactic**, e. g., an adjective pointing to the noun it modifies and more generally grammatical relationships between pairs of words (Widdows and Dorow, 2002; Ferrer i Cancho, Solé, and Köhler, 2004); or (3) **semantic**, e. g., based on synonymy (Leskovec et al., 2004; Kozareva et al., 2008). Moreover, the graph can represent: (1) a **sentence** such as a dependency tree commonly used in NLP (Tratz and Hovy, 2011); (2) a **single document** like in most related works; (3) **multiple documents** (Meladianos et al., 2015a); or even (4) the entire **collection** of documents (Ferret, 2002; W. Wang et al., 2005; Sandler et al., 2009; Velikovich et al., 2010) depending on the application and the granularity we wish to achieve.

DENOMINATIONS In the literature, these representations have been referred to as *term co-occurrence graphs* (Ohsawa et al., 1998), *collocation networks* (Ferret, 2002), *dictionary graphs* (Blondel and Senellart, 2002), *word lattices* (Barzilay and Lillian Lee, 2003), *text graphs* (Mihalcea and Tarau, 2004), *term-distance graphs* (Gamon, 2006), *term graphs* (Palshikar, 2007), *lexical graphs* (Velikovich et al., 2010), *word graphs* (Filippova, 2010) and *word networks* (Lahiri and Mihalcea, 2013). Additionally, there exist some representations usually referred to as *association graphs* (Arora and Nyberg, 2009; Jiang et al., 2010) that mix linguistic units and linguistic labels for nodes and edges – to some extent, they are closer to the syntactic parse trees from NLP because of the use of POS tags as node labels. We proposed a novel denomination – *graph-of-words* by analogy with bag-of-words – because we think it better describes its goal and not only its representation.

APPLICATIONS The applications of graph-based representations of text that have been considered in the literature are numerous: (1) *text summarization* (Erkan and Radev, 2004; Leskovec et al., 2004; Ganesan et al., 2010; Balinsky et al., 2011); (2) *keyword extraction* (Matsuo et al., 2001b; Litvak and Last, 2008; Lahiri, S. R. Choudhury, et al., 2014) (3) *ad hoc information retrieval* (Blanco and Lioma, 2007) (4) *text categorization* (Markov et al., 2007; Hassan et al., 2007; Valle and Öztürk, 2011); (5) *native language identification* (Lahiri and Mihalcea, 2013); (6) *novelty detection* (Gamon, 2006); and more generally (7) *network analysis* of written human language (Masucci and Rodgers, 2006; Lahiri, 2014).

LANGUAGES Most of the works considered English documents but there have been a few others on various languages: (1) German, Czech and Romanian (Ferrer i Cancho, Solé, and Köhler, 2004); (2) Hindi and Bengali (M. Choudhury et al., 2007); (3) Russian (Kapustin and Jamsen, 2007); (4) Portuguese (Antiqueira et al., 2007; Abilhoa and de Castro, 2014); (5) Chinese (Zhu et al., 2003; Wei Liang et al., 2009); (6) French (Boudin, 2013); and (7) Croatian (Beliga and Martinčić-Ipšić, 2014) among others. In this dissertation, we solely focus on English collections of textual documents.

3.2 GRAPH-OF-WORDS: OUR REPRESENTATION

In this section, we define the concept of graph-of-words as introduced in (Rousseau and Vazirgiannis, 2013b), discuss its variants and report its known network properties.

3.2.1 *Model definition*

Given a document, its **graph-of-words** representation is defined as the *statistical* network whose vertices correspond to unique terms of the document and whose edges represent co-occurrences between the terms within a fixed-size sliding window over the full processed document ignoring sentence boundaries. We highlight the fact that we are considering *terms* rather than *words* (cf. Section 2.1.2.2) because any dimensionality reduction pre-processing steps (e. g., stop word removal or stemming) has been applied beforehand, hence a *processed* document. One could argue that it is in fact a graph-of-terms but we chose this denomination by analogy with the bag-of-words, which is in practice a bag-of-terms as well. The underlying assumption is that all the words present in a document have some relationships with one another, modulo a window size outside of which the relationship is not taken into consideration. Note that in order to prevent *self-loops* (i. e. a node sharing an edge with itself), we shorten the sliding window when two terms are equal, which happens more frequently with stop word removal and stemming.

The optional **edge direction** (e. g., based on the natural flow of the text or a syntactic dependency) and **edge weight** (e. g., based on the number of co-occurrences) are considered variants of the representation rather than different models, mostly because they depend on the application considered as observed in our experiments. Therefore, this is the same representation as the one used by Ohsawa et al. (1998), Ferrer i Cancho and Solé (2001), Schenker (2003), and Mihalcea and Tarau (2004) – we only cite here the papers that did not reference any previous work for their representation and that served as references for other papers later on. Indeed, the novelty of our work was not so much in the representation itself – even though we gave it a name that hopefully should catch on – but rather in the applications and the various findings we report in this dissertation.

PROCEDURE Figure 3.1 illustrates the process of generating a graph-of-words representation for a given textual document (original text in upper box). The figure has been manually created using the `Ipe` extensible drawing editor based on an automated graph visualization from the `JUNG` library that used the Kamada-Kawai force-based algorithm (Kamada and Kawai, 1989). First, the document is (1) lowercased; (2) tokenized; (3) stripped out of non-alphanumeric characters and stop words; and (4) stemmed. This results in a processed text that corresponds to a single sequence of terms. The set of n unique terms constitute the vertices of the graph-of-words. The edges are then drawn between terms co-occurring within a fixed-size sliding window W (of size 3 for the example) over the sequence, i. e. capturing trigrams. Multiple co-occurrences might be encoded through the edge weight as illustrated (the thicker the edge, the higher the weight on the figure). Here we do not direct the edges, for instance according to the natural flow of the text, and thus "information retrieval" and "retrieve information" are both mapped to inform—retriev in the example. Overall, the complexity is $\mathcal{O}(nW)$ in time and $\mathcal{O}(n+m)$ in space with m being the final number of edges (at most nW).

3.2.2 *Variants*

There have been many variants of the graph-of-words definition and building procedure considered in the literature as well as explored by us in our research in terms of edge direction, edge weight, pre-processing steps and sliding window.

EDGE DIRECTION Naturally, we would direct the edges according to the flow of the text (i. e. left to right, at least for English) but in all our experiments we did not observe any significant differences in effectiveness (or at best with low effect sizes) between undirected edges, forward edges (natural flow of the text – an edge $term_1 \rightarrow term_2$ meaning that $term_1$ precedes $term_2$ in a sliding window) and backward edges (the opposite), especially across tasks. In the literature, the first representations had no direction (Ohsawa et al., 1998; Ferrer i Cancho and Solé, 2001) then Schenker (2003) and Mihalcea and Tarau (2004) introduced forward edges and later Litvak and Last (2008) considered backward edges. This lack of a dominant choice for edge direction makes us recommend undirected edges for ease of implementation and also because it follows our initial idea of using graphs to capture sets of words rather than (sub)sequences of words; the exception being when the actual application requires some natural language generation, e. g., summarization from path extraction (Filippova, 2010).

EDGE WEIGHT Naturally, we would weight the edges according to the strength of the relationship. For a statistical graph-of-words, this means taking into account the number of co-occurrences between its two endpoints, optionally boosting the weight with the inverse distance in the sliding window (Gamon, 2006). Actually, the weight can end up being the inverse of the co-occurrence frequency if the subsequent graph mining process is based on path lengths for

information retrieval is the activity of obtaining textual documents relevant to an information need from a collection of textual documents. in ad hoc information retrieval, the information need is conveyed through an ad hoc textual user query and processed by a search engine to retrieve information according to some relevance. retrieval models assign to documents relevance scores with regard to user queries based on query term frequencies in the document and the collection. being able to retrieving information is crucial for users when searching for answers.

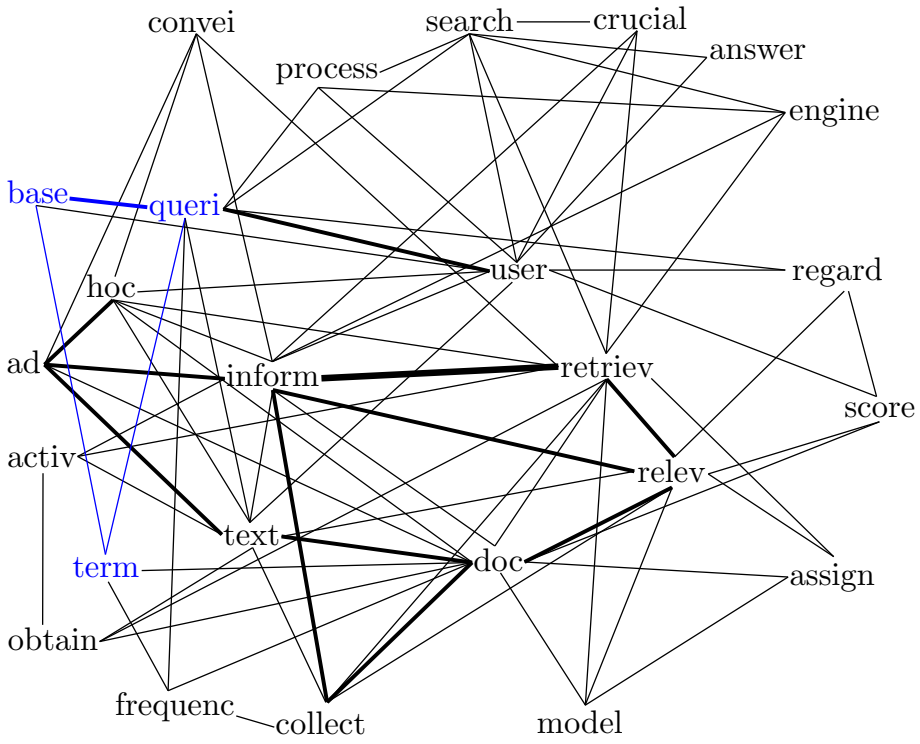


Figure 3.1 – Illustration of the graph-of-words representation of a document (text in upper box). Nodes correspond to unique terms of the document after stop word removal and stemming. Edges represent co-occurrences within a window of size 3 over the processed text. Edge thickness indicate its weight (1, 2 and 6), i. e. the number of co-occurrences – no edge direction, i. e. "information retrieval" and "retrieve information" are both mapped to inform—retriev. In blue, we highlight one particular subgraph.

instance (Abilhoa and de Castro, 2014). In any case, as often when considering unbounded feature values, the use of edge weights will depend eventually on the application. Indeed, if we consider documents separately of one another, for instance in single-document KwE, then encoding the number of co-occurrences can only be beneficial. But when considering multiple documents of heterogeneous length simultaneously, some normalizations need to be applied so as not to favor longer documents that will result on average in higher edge weights.

PRE-PROCESSING STEPS Prior to the graph construction, the standard optional NLP pre-processing steps can be applied to reduce the number of terms, i. e. the number of nodes: stop word removal [2], stemming [3] or lemmatization [4] and POS-tag [5] filtering, e. g., keeping only the nouns and adjectives for keyword extraction, i. e. NN, NNS, NNP, NNPS, and JJ using the Penn Treebank Project notations (Santorini, 1990). There is no consensus in the literature on which set(s) of pre-processing steps to use – this is actually not specific to research on graph-of-words but also true for IR and NLP. Generally speaking, dimensionality reduction techniques will increase the "academic" performances but might not be suitable for more practical applications where users might not be interested in stemmed keywords for instance!

SLIDING WINDOW The size of the sliding window, which represents the maximum scope for what we consider a phrase, varies from 2 to 10 in general. Some works did consider more words, up to 40 for Blanco and Lioma (2007) but only to recommend 6 in the end. The choice of a fixed vs. a parameterized sliding window seems to be more in the favor of the former, at least out of the 50 works or so we found on the subject and this is the approach we adopted. However, we want to highlight three subtleties that are rarely mentioned in papers but that can make a difference when trying to reproduce the experiments: (1) is the sliding window over the original text or its processed version; (2) does the sliding window span sentences; and (3) how to deal with multiple occurrences of the same word in a window? In our case, in all our works, we ended up using a sliding window of size 4 over the processed text, i. e. after stop word removal in particular, ignoring sentence boundaries (we experimented with the Apache OpenNLP Sentence Detector [5]) and shortening the window when finding the same term twice (including a stemmed version) to prevent self-loops.

3.2.3 *Subgraph-of-words*

We highlighted in blue on Figure 3.1 one particular subgraph extracted initially from the trigram "based on query term". The same subgraph would be found in other graph-of-words representations of documents containing

2. http://jmlr.org/papers/volume5/lewis04a/a11-smart-stop-list/english.stop
3. http://tartarus.org/~martin/PorterStemmer
4. https://wordnet.princeton.edu/man/morphy.7WN.html
5. http://opennlp.apache.org/documentation/1.5.3/manual/opennlp.html

one of the following n-grams: "query-based term", "query based on selected terms", "terms based on a user query", etc. Actually, the subgraph does not have to be present in the form of one single sequence of terms – documents containing "query-based" and "query terms" separately would also possess the same subgraph in their graph-of-words representation. This is why we think of subgraphs of graph-of-words as **long-distance n-grams** (Rousseau and Vazirgiannis, 2015) by analogy with the long-distance bigrams from Bassiou and Kotropoulos (2010).

Note that we make a distinction here between a subgraph that can be any combination of nodes and edges from the original graph and an *induced* subgraph where given a set of nodes, all the edges between these vertices present in the original graph appear in the subgraph.

3.2.4 *Graph properties*

Ferrer i Cancho and Solé (2001) and Matsuo et al. (2001a) reported quasi-simultaneously that syntactic graph-of-words were **small worlds** (Milgram, 1967), i. e. networks whose nodes are highly clustered (compared to a random graph) yet the path length between them is small (close to that of a random graph), typically in the order of the logarithm of the number of nodes at most (Watts and Strogatz, 1998). Senellart (2001) reported the same for his dictionary graph. Additionally, Ferrer i Cancho and Solé (2001) also noticed a **scale-free** distribution of node degrees, i. e. the number of nodes of degree δ in a graph is inversely proportional to δ, optionally with a power constant.

4

RETRIEVING INFORMATION FROM GRAPH-OF-WORDS

In this chapter, we retrieve information in text from the graph-of-words representation of documents. We first review how the seminal TF×IDF term weighting scheme capitalizes on the raw term and document frequencies to favor the most important terms of a document through various successive normalizations. We then present an alternative term weight based on graph-of-words to retrieve more relevant documents in ad hoc Information Retrieval (IR).

4.1 INTERPRETING TF×IDF AS COMPOSING NORMALIZATIONS

As introduced in Section 2.1.3, the concepts of Term Frequency (TF) and Inverse Document Frequency (IDF) are the building blocks of the earliest scoring functions used in ad hoc IR and more generally of the term weighting schemes in Text Mining (TM). These scoring functions capitalized on the TF×IDF general weighting scheme to favor query terms that are frequent in a document while rare in the collection – we make here a distinction between TF×IDF as an abstract model and TF-IDF, the specific IR retrieval model with *pivoted document length normalization* defined in (Singhal, Choi, et al., 1999) (cf. Section 4.1.3 for the formula). Nevertheless, the research community proposed and used successive normalizations on the raw term and document frequencies to boost the performances and explained them in terms of satisfying some heuristic retrieval constraints as mentioned in Section 2.2.1.2.

UNFAIR BASELINE Too often in TM and IR, the term "tf-idf" is used without mentioning which specific normalizations were applied and in which composition order, which is crucial for getting the best and fairest results and also for reproducibility. Indeed, taking raw term and document frequencies as a baseline is unfair because normalizations are essential and can *greatly* increase the performances in ad hoc IR as verified in our own experiments (cf. Table 4.2, difference in results between TF and TF_x). It is less significant in other TM tasks because for instance in TC we are generally not trying to categorize documents based on the number of co-occurrences of a given query term but rather cluster together documents sharing the same terms. Actually, opinion mining (i. e. predicting the sentiment polarity of a document, typically a user review) is known to be a subtask where binary features (i. e. presence/absence of a term, regardless of its term frequency) have been claimed to yield better prediction results (Pang, Lilian Lee, and Vaithyanathan, 2002).

OUR MOTIVATION It is in this spirit that we carefully explored in (Rousseau and Vazirgiannis, 2013a) how TF×IDF is built through various successive normalizations on the term and document frequencies so as to better capitalize later on an alternative term weight (in our case extracted from a graph representation of the document). In this work, we also proposed a clear way for researchers to indicate the specific model they considered by indicating as subscripts the normalizations successively applied and the composition order.

EQUATION A TF×IDF scoring function $\hat{r}$ between a document $\boldsymbol{d}$ from collection $\mathcal{D}$ and a query $\boldsymbol{q}$ is composed of three components supposedly independent of one another: one at the *query level* (QF), one at the *document level* (TF) and one at the *collection level* (IDF):

$$\hat{r}(\boldsymbol{d}, \boldsymbol{q}) = \sum_{t \in \boldsymbol{q}} \text{QF}(tf_{\boldsymbol{q}}(t)) \times \text{TF}(tf_{\boldsymbol{d}}(t)) \times \text{IDF}(df_{\mathcal{D}}(t)) \tag{4.1}$$

where $tf_{\boldsymbol{q}}(\cdot)$ is the term frequency in the query, $tf_{\boldsymbol{d}}(\cdot)$ in the document and $df_{\mathcal{D}}(\cdot)$ the document frequency in the collection. Query Frequency (QF) and Term Frequency (TF) have similar roles, the former normalizing the term frequency in the query and the latter in the document. Actually, QF only makes sense in the context of ad hoc IR where there is a query and with some term repetitions, which is unlikely to happen in Web queries for instance because of their short length. Therefore it is common to omit it in practice, be it in IR or more generally in TM. Most of the research work on the subject, including our own, has been devoted to the TF and the IDF components. BM25 (Robertson, Walker, et al., 1994) might be the only state-of-the-art retrieval model to explicitly take it into account (probably because it is one of the oldest and dates back from when queries used to be longer and not Web-related).

4.1.1 *Term frequency normalizations*

Since the early work of Luhn (1957), the research community has believed that the term frequency plays an important role for retrieval and is therefore at the center of all IR retrieval models. Intuitively, the more times a document contains a query term, the more relevant this document is w. r. t. the query. However, the raw term frequency (tf, i. e. the number of times a term occurs in a document) proved to be non-optimal in ad hoc IR and researchers started normalizing it considering multiple criteria that were later explained as functions satisfying heuristic retrieval constraints (Fang et al., 2004, 2011). Still, it is commonly accepted that the overall scoring function must be a monotonically increasing function of the term frequency and thus this needs to be also true for the TF component and the successive normalizations that compose it.

4.1.1.1 *Decreasing marginal gain in relevance*

The marginal gain of seeing an additional occurrence of a term inside a document should be decreasing. Indeed, the change in the score caused by increasing

tf from 1 to 2 should be much larger than the one caused by increasing tf from 100 to 101 (and by default it is not since we sum each term's contribution in Equation 4.1). Mathematically, this corresponds to applying a **concave transformation** on the raw tf. We prefer the term *concave* like in (Clinchant and Gaussier, 2010) to *sub-linear* like in (Manning, Raghavan, et al., 2008, p. 126; Lv and Zhai, 2011a) since the *positive homogeneity* property is rarely respected (and actually not welcomed) and the *sub-additivity* one, even though desirable, not sufficient enough to ensure a decreasing marginal gain. Note that the transformation still needs to be monotonically increasing in the term frequency tf.

There are mainly two concave transformations used in the literature: the one in TF-IDF and the one in BM25 that we respectively called **log-concavity** (TF_l) and **k-concavity** (TF_k) defined as follows for a term t and a document $\boldsymbol{d}$:

$$TF_l(t, \boldsymbol{d}) = 1 + \ln\left(1 + \ln\left(tf_{\boldsymbol{d}}(t)\right)\right) \tag{4.2}$$

$$TF_k(t, \boldsymbol{d}) = \frac{(k_1 + 1) \times tf_{\boldsymbol{d}}(t)}{k_1 + tf_{\boldsymbol{d}}(t)} \tag{4.3}$$

where k_1 is a constant set by default to 1.2 that corresponds to the asymptotical maximal gain achievable by multiple occurrences compared to a single occurrence ($TF_k(t, \boldsymbol{d}) = 1$ when $tf_{\boldsymbol{d}}(t) = 1$ and $TF_k(t, \boldsymbol{d}) = k_1 + 1$ when $tf_{\boldsymbol{d}}(t) \to \infty$).

4.1.1.2 *Diversity in query terms*

If two documents have the same total number of occurrences of all query terms, a higher score should be given to the document covering more distinct query terms. Mathematically, this corresponds to applying a **sub-additive transformation** on the raw tf: $f(tf_1 + tf_2) \leq f(tf_1) + f(tf_2)$. Logarithm is the typical example of a sub-additive function. As noted by Fang et al. (2004), TF_l and TF_k already satisfy that property so no additional normalization is actually needed. More generally, any concave function $f : [0, +\infty) \to [0, +\infty)$ with $f(0) = 0$ is also sub-additive so all monotonically increasing concave functions satisfying $f(0) = 0$ are good candidates for ensuring both a decreasing marginal gain and diversity. Note that these two notions are different from the ones discussed in Section 2.2.1.1 where it was question of overlapping results on the SERP and diminishing return in relevance between documents and not between terms.

4.1.1.3 *Document length penalization*

When collections consist of documents of varying lengths (e. g., Web pages), longer documents will – as a result of containing more terms – have higher tf values without necessary containing more information. For instance, a document twice as long and containing twice as more times each term should not get a score twice as large but rather a very similar score. As a consequence, it is commonly accepted that the overall scoring function should be an inverse function of the document length to compensate that effect.

L^p NORMALIZATION Early works on the Vector Space Model (VSM) (Salton and Buckley, 1988) suggested to normalize the score by the norm of the vector, be it the L^0 norm (number of unique terms in the document), the L^1 norm (length of the document), the L^2 norm (Euclidian length of the document) or the L^∞ norm (maximum tf value in the document). However, these norms still mask some subtleties about longer documents. Indeed, it turned out that in practice longer documents have a higher probability of relevance (computed using ground truth data) and therefore should have a higher probability of retrieval – Figure 4.1 illustrates this (only looking at the black curve for now, other curves corresponding to other models later introduced as we consider more normalizations). Robertson and Spärck Jones (1994) explained it either by: (1) the *scope hypothesis*, the likelihood of a document's relevance increases with length due to the increase in material covered; or (2) the *verbosity hypothesis*, where a longer document may cover a similar scope than shorter documents but simply uses more words and phrases.

PIVOTED NORMALIZATION As a result, the research community has been using a more complex transformation known as **pivoted document length normalization** and defined as follows for a term t and a document $\boldsymbol{d}$:

$$TF_p(t, \boldsymbol{d}) = \frac{tf_{\boldsymbol{d}}(t)}{1 - b + b \times {|\boldsymbol{d}|}/{avdl}} \tag{4.4}$$

where $b \in [0, 1]$ is the slope parameter, $|\boldsymbol{d}|$ the document length and $avdl$ the average document length across the collection of documents as defined in (Singhal, Buckley, et al., 1996). Basically, b controls how much penalization the document should get based on its length: 0 none, 1 all documents have the same probability of retrieval regardless of their document length and in between the amount of tilting around the pivot value $avdl$ (cf. Figure 4.1).

RECENT WORK Note that more recently, Losada et al. (2008) contested this popular belief arguing that the positive relationship between relevance and document length observed empirically was an artefact of the test collections due to incompleteness in the relevance judgments (not all relevant documents are marked as such by human annotators since they would need to go through thousands to billions of documents for each query). However, until the benchmark datasets used by the research community include this finding, taking into account the document length results in better results (in terms of standard evaluation metrics). Additionally, Na (2015) argued very recently that the document length penalization should be in two stages to explicitly take into account both the verbosity and the scope hypotheses.

4.1.1.4 *Composition of normalizations*

With multiple functions satisfying different heuristic retrieval constraints, it seems rather natural to compose them successively. For instance, here are the two compositions behind TF-IDF and BM25, respectively $\text{TF}_{p\circ l}$ (= $\text{TF}_p \circ \text{TF}_l$) and $\text{TF}_{k\circ p}$ (= $\text{TF}_k \circ \text{TF}_p$) for a term t and a document $\boldsymbol{d}$:

$$TF_{p\circ l}(t, \boldsymbol{d}) = \frac{1 + \ln\left(1 + \ln\left(tf_{\boldsymbol{d}}(t)\right)\right)}{1 - b + b \times \frac{|\boldsymbol{d}|}{avdl}} \tag{4.5}$$

$$\begin{aligned} TF_{k\circ p}(t, \boldsymbol{d}) &= \frac{(k_1 + 1) \times \frac{tf_{\boldsymbol{d}}(t)}{1-b+b\times\frac{|\boldsymbol{d}|}{avdl}}}{k_1 + \frac{tf_{\boldsymbol{d}}(t)}{1-b+b\times\frac{|\boldsymbol{d}|}{avdl}}} \\ &= \frac{(k_1 + 1) \times tf_{\boldsymbol{d}}(t)}{k_1 \times \left(1 - b + b \times \frac{|\boldsymbol{d}|}{avdl}\right) + tf_{\boldsymbol{d}}(t)} \\ &= \frac{(k_1 + 1) \times tf_{\boldsymbol{d}}(t)}{K + tf_{\boldsymbol{d}}(t)} \end{aligned} \tag{4.6}$$

where $K = k_1 \times (1 - b + b \times \frac{|\boldsymbol{d}|}{avdl})$ as defined in (Robertson, Walker, et al., 1994). Note that under this last form, it is not obvious that $\text{TF}_{k\circ p}$ is really a composition of two functions with the same properties as the one in $\text{TF}_{p\circ l}$. We believe this is the main reason why composition has never been considered before explicitly. By doing so, we provided not only a way to fully explain BM25 as a TF×IDF scoring function but also a way to easily consider variants of a term weighting scheme by simply changing the order of composition. Actually, this led us to a new TF×IDF scoring function that consistently outperforms BM25 on several standard datasets.

SMART NOTATIONS Salton and Buckley (1988) proposed along with their SMART retrieval system a set of notations for specifying which normalizations are applied but they make a clear distinction between the transformations applied on the raw tf and the vector normalization applied at the document level to account for document length. Therefore, it is then hard to fully fit BM25 in the TF×IDF weighting scheme since the document length penalization is applied prior to the concave transformation. In our case, we do not make this distinction and we allow for as many transformations as necessary.

4.1.1.5 *Lower-bounding regularization*

Lv and Zhai (2011b) introduced two new constraints to the work of Fang et al. (2004) to lower-bound the TF component. In particular, they stated that there should be a sufficiently large gap in the score between the presence and absence of a query term even for very long documents where TF_p tends to 0 and a fortiori the overall score too. Indeed, there is no reason for preserving some sort of continuity between the presence and absence of a query term. Mathematically, this corresponds to be composing with a third function TF_δ

that is always composed after TF_p since it compensates the potential null limit introduced by TF_p and it is defined as follows for a term t and a document $\boldsymbol{d}$:

$$TF_\delta(t, \boldsymbol{d}) = \begin{cases} tf_{\boldsymbol{d}}(t) + \delta & \text{if } tf_{\boldsymbol{d}}(t) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

where δ is the gap, set to 0.5 if TF_δ is composed immediately after TF_p and 1 if concavity is applied in-between. These are the two default values defined in the original papers (Lv and Zhai, 2011b,c) and we just interpreted their context of use in terms of order of composition.

4.1.2 *Document frequency normalizations*

While the higher the frequency of a term in a document is, the more salient this term is supposed to be, this is no longer true at the collection level. Actually, this is quite the inverse since these terms have presumably a lower discrimination power. Indeed, a query term that appears in a lot of documents might not be the best one in helping assess the relevance of documents w. r. t. the query. However, these frequent terms are still somewhat common in queries, and simply removing them, as advocated by Svenonius (1972), could have a damaging effect on retrieval performances. Therefore all terms should be allowed to match but the value of matches on frequent terms should be lower than that for non-frequent terms.

TERM SPECIFICITY Spärck Jones (1972) proposed to capture this **term specificity** through the inverse of the document frequency, defining the Inverse Document Frequency (IDF) of a term t as follows:

$$IDF(t) = \log \frac{N}{df(t)} \tag{4.8}$$

which already normalizes the document frequency with the total number of documents N and a concave transformation. It appeared as such in (Robertson and Spärck Jones, 1976; Salton and Buckley, 1988; Church and Gale, 1995; Singhal, Salton, et al., 1995; Manning, Raghavan, et al., 2008). Later on (Robertson and Spärck Jones, 1994; Singhal, Choi, et al., 1999) and from there (Fang et al., 2004; Lv and Zhai, 2011b), it was changed to a slightly different version:

$$IDF(t) = \log \frac{N+1}{df(t)} \tag{4.9}$$

supposedly to prevent a null IDF value when a term appears in all documents ($df(t) = N$) but no justification nor any explanation were found in the literature to explain this sudden change.

4.1.2.1 *Probabilistic IDF*

In parallel, Robertson and Spärck Jones (1976) developed a relevance term weighting scheme from a probabilistic perspective, which in the absence of relevance judgment and relevance feedback can be interpreted as the *logit* (a. k. a. log-odds) of the document frequency of a term t:

$$IDF(t) = \log \frac{N - df(t)}{df(t)} \tag{4.10}$$

However, it was noted in (Robertson and Walker, 1997) that for terms that occur in more than half the documents of the collection ($df(t) > N/2$), this would result in negative values, i. e. a score less than with the absence of the term. This is obviously problematic for ad hoc IR but can be good for other applications, e. g., novelty detection in our case (Karkali, Rousseau, et al., 2013) where penalizing previously seen terms is of interest.

4.1.2.2 *Add-1/2 smoothed IDF*

LIDSTONE SMOOTHING As explained by Manning, Raghavan, et al. (2008, p. 226), for trials with categorical outcomes (such as noting the presence or absence of a term in a document), one way to estimate the probability of an event from data is simply to count the number of times an event occurred ($df(t)$) divided by the total number of trials (N) – this is referred to as the *relative frequency* of the event ($df(t)/N$). Estimating the probability as the relative frequency corresponds to the Maximum Likelihood Estimate (MLE), because this value makes the observed data maximally likely. However, if we simply use the MLE like in the original IDF formula (Equation 4.8), then the probability given to observed events is usually overestimated and for unseen events underestimated (actually null: $df(t) = 0$ for query terms that do not appear in the collection, leading in our case to infinite idf values, which is unlikely but undesirable).

Simultaneously decreasing the estimated probability of seen events and increasing the probability of unseen events is referred to as **smoothing**. One simple way of smoothing is to add a number α to each of the observed counts (and α times the number of possible outcomes to the total number of trials), which is known as (add-α) **Lidstone smoothing** (W. E. Johnson, 1932; Jeffreys, 1948, sec. 3.23). In the case of $\alpha = 1$ and binary outcomes, this is also known as Laplace smoothing by analogy with Laplace's rule of succession for estimating the probability that the sun will rise tomorrow given that it has risen every day for the past 5,000 years (Laplace, 1814, p. 23). These *pseudo-counts* correspond to the use of a uniform distribution over the vocabulary as a *Bayesian prior* (hence why we take into account the number of possible outcomes). We initially assume a uniform distribution over events, where the magnitude of α denotes the strength of our belief in uniformity, and we then update the probability based on observed events. This is a form of **maximum a posteriori** estimation, where we choose the most likely point value for probabilities based on the prior (α) and the observed evidence ($df(t)$).

FINAL IDF VERSION Early on, Robertson and Spärck Jones (1976) and Croft and Harper (1979) proposed to add Lidstone smoothing ($\alpha = 1/2$) to the probabilistic IDF under the following form (when again no relevance judgment nor relevance feedback is available):

$$IDF(t) = \log \frac{N - df(t) + 0.5}{df(t) + 0.5} \tag{4.11}$$

Similarly, for the standard IDF, the following smoothed form appeared in (Amati and van Rijsbergen, 2002; Lv and Zhai, 2011a,c, 2012):

$$IDF(t) = \log \frac{N + 1}{df(t) + 0.5} \tag{4.12}$$

The 1 (=2α) in the numerator comes from the fact that there are two possible outcomes: presence or absence of a term in each document. We think this last version is the one that makes the most sense because it does not suffer from any of the shortcomings aforementioned (null, infinite nor negative values) while fixing them in a principled way using pseudo-counts.

4.1.2.3 *Theoretical justifications*

Initially, the idea of IDF and its associated formula were merely a heuristic backed by empirical proof. In the 1990s-2000s, there has been a fair number of works including (Church and Gale, 1995; Robertson, 2004; Joho and Sanderson, 2007; Lillian Lee, 2007; Metzler, 2008) that tried to explain it in a more theoretical manner. Papineni (2001) showed that IDF is the optimal scoring function for document self-retrieval, which confirms our intuition that IDF can be seen as a cheap estimator of the likelihood that the unigram language model (built from the previously seen documents) generated the incoming document in a stream of documents for novelty detection (Karkali, Rousseau, et al., 2013).

4.1.3 *TF-IDF vs. BM25*

By TF-IDF, we refer hereinafter to the TF×IDF retrieval model defined in (Singhal, Choi, et al., 1999), often called **pivoted normalization weighting**. The retrieval model corresponds for a term t and a document $\boldsymbol{d}$ to TF_{pol}×IDF:

$$TF\text{-}IDF(t, \boldsymbol{d}) = \frac{1 + \ln\left(1 + \ln\left(tf_{\boldsymbol{d}}(t)\right)\right)}{1 - b + b \times \frac{|\boldsymbol{d}|}{avdl}} \times \log \frac{N + 1}{df(t) + 0.5} \tag{4.13}$$

By BM25 (BM stands for Best Match), we refer hereinafter to the scoring function, often called **Okapi weighting**, defined in (Robertson, Walker, et al., 1994) for a term t and a document $\boldsymbol{d}$ as follows:

$$BM25(t, \boldsymbol{d}) = \frac{(k_3 + 1) \times tf_{\boldsymbol{q}}(t)}{k_3 + tf_{\boldsymbol{q}}(t)} \times \frac{(k_1 + 1) \times tf_{\boldsymbol{d}}(t)}{K + tf_{\boldsymbol{d}}(t)} \times \log \frac{N + 1}{df(t) + 0.5} \tag{4.14}$$

It corresponds to $\text{TF}_{k\circ p}\times$IDF when we omit QF (k-concavity of parameter $k_3 = 1000$ for $tf(t, \boldsymbol{q})$). Thus, it has an inverse order of composition between the concavity and the document length penalization compared to TF-IDF but they are more similar than what most people think.

PIV+, BM25+ AND BM25L The weighting models Piv+ and BM25+ defined by Lv and Zhai (2011b) correspond respectively to $\text{TF}_{\delta\circ p\circ l}\times$IDF (= $\text{TF}_\delta \circ \text{TF}_p \circ \text{TF}_l \times$IDF) and $\text{TF}_{\delta\circ k\circ p}\times$IDF (= $\text{TF}_\delta \circ \text{TF}_k \circ \text{TF}_p\times$IDF) while BM25L defined by Lv and Zhai (2011c) corresponds to the other order of composition $\text{TF}_{k\circ\delta\circ p}\times$IDF (= $\text{TF}_k \circ \text{TF}_\delta \circ \text{TF}_p\times$IDF). We clearly see that the only difference between BM25+ and BM25L is the order of composition: this is one of the advantages of our framework – easily represent and compute multiple variants of a same general weighting model. In our experiments reported in (Rousseau and Vazirgiannis, 2013a), we considered all the possible orders of composition between TF_k or TF_l, TF_p and TF_δ with the condition that TF_p always precedes TF_δ as explained before. For instance, we considered a novel scoring function $\text{TF}_{l\circ\delta\circ p}\times$IDF with $\text{TF}_{l\circ\delta\circ p}$ defined a term t and a document $\boldsymbol{d}$ as follows:

$$TF_{l\circ\delta\circ p}(t, \boldsymbol{d}) = 1 + \ln\left(1 + \ln\left(\frac{tf_{\boldsymbol{d}}(t)}{1 - b + b \times {|\boldsymbol{d}|}/{avdl}} + \delta\right)\right) \tag{4.15}$$

where b is set to 0.20 and δ to 0.5. This model consistently outperformed the others on several standard TREC datasets.

4.1.4 *Which normalizations and in which order?*

While the proposed framework allows researchers to clearly specify which TF×IDF variant they use, it does not give the optimal set of transformations to apply along with its optimal order of composition. Indeed, we have seen that TF-IDF and BM25 rely on a different concave transformation, resulting in an inverse order of composition. With only three successive transformations to apply, the best way remains to test all possibilities on a training set. Even with more normalizations like for the SMART notations, Zobel and Moffat (1998) did not find any principle way of combining them. In the future, with more heuristic retrieval constraints to satisfy and additional transformations to meet them, there could be some research to be done on automatically **learning to compose** these normalizations.

4.2 TW-IDF, GOING BEYOND THE TERM FREQUENCY

Behind the bag-of-words representation widely adopted in ad hoc IR, there is this assumption that terms are independent of one another. The result is at least two-fold: (1) the relevance of a document w. r. t. a query is estimated by summing up each query term's individual contribution (cf. Equation 4.1) and (2) these individual contributions are estimated independently of one another - the term frequencies solely depend on the occurrences of the particular term regardless of its neighboring terms in the document. This is also due to practical reasons in terms of indexing and querying (cf. Section 2.2.1.3). Note that some works tried to consider the query as a long-distance n-gram in the retrieved documents, e. g., Clarke, Cormack, et al. (1995) experimented with using the distance between query terms in retrieved documents to re-score retrieval results.

Initially, we wanted to challenge both aspects of the term independence assumption by representing the query as a graph-of-words and then consider the task as a graph matching/retrieval problem – basically retrieve documents containing the query in a graphical setting. However, this would mean building a graph index that would potentially store millions to billions of graph-of-words while allowing for fast querying based on a subgraph (the query). Considering that people choose for efficiency reasons to store raw tf values in the inverted index rather than the final normalized version used at query time even if it means computing it on the fly every single time, we discarded this research direction that would require a very efficient solution to even begin to compete with modern search engines and we explored instead replacing the stored term frequency (tf) with an alternative graph-based term weight (tw), hopefully more meaningful.

4.2.1 *Model definition*

In this section, we define the document representation, the term weighting scheme and the scoring function TW-IDF we came up with for ad hoc IR, which we introduced in (Rousseau and Vazirgiannis, 2013b).

4.2.1.1 *Related work*

To the best of our knowledge, Blanco and Lioma (2007) were the first to explore this approach. They proposed to apply PageRank (cf. Section 5.1.2) on each graph-of-words to extract "random walk term weights" (rw) and then replaced the term frequencies by them. It is unclear in the paper but from the experimental results, it seems that no concave transformation was applied on the raw term frequency, which lowers the baseline, and as for the pivoted document length normalization, the value of the slope parameter b was not discussed, neither for the baseline nor their model. They rather focused on the sliding window W for the graph-of-words construction, trying values from 2 to 40. They later extended their work (Blanco and Lioma, 2012) by including

in the scoring function some global graph properties such as average degree, average path length and clustering coefficient interpreting them in the context of a graph-of-words as respectively the cohesiveness of the document, "how tightly knit the discourse is" and its "topical clustering". However, this time, they used a logarithmic concave transformation on the *rw* values but no document length penalization at all, again focusing on the size of the sliding window.

4.2.1.2 *The proposed model*

For ad hoc IR, our experiments led us to represent each document as an **unweighted undirected graph-of-words** with a fixed sliding window of size 4 (cf. Figure 3.1 for illustration) where the node degree is the term weight used for retrieval.

EDGE DIRECTION Initially, we tried to direct the edges according to the natural flow of a text (from left to right, at least in English) for marginal yet significant improvement then according to some grammatical hierarchy, e. g., based on POS-tagging (adjectives pointing to nouns pointing to verbs) but that last approach proved to be empirically more expensive while not more effective. Therefore, we do not recommend any edge direction for ad hoc IR.

EDGE WEIGHT Initially, we considered weighted edges in the graph. It seemed natural to weight the edges by the number of co-occurrences of the two terms in the text. We even tested with boosted weights that depend on the distance between the two terms within the considered sliding window assuming that the closer two terms are, the stronger their relationship should be. Indeed, in Figure 3.1, considering the window "query term frequencies", the additional weight of the association query–term should be a priori more important than the one for query–frequenc. However, in practice, the use of unweighted edges consistently led to better results. We will propose a possible explanation in Section 4.2.3.1 but we first need to introduce the term weighting and the scoring function since it inherently depends on these choices for the implication on the retrieval effectiveness.

TERM WEIGHT Following the intuition behind the term frequency that the most frequent terms of a document should get a higher score because more representative of the document, the weights given to terms of the graph-of-words should express how important the nodes are in the network and one measure of importance is how central a vertex is (Freeman, 1979). Unlike Blanco and Lioma (2007) though, we chose to keep the vertex score definition simple – its degree – and more precisely its **indegree** in case of a directed network, which, in the context of an unweighted graph-of-words, corresponds to the number of different contexts of co-occurrence of a term inside a document, i. e. how many different terms surround it regardless of how many times it occurs with each term.

CENTRALITY MEASURES We did explore other vertex centrality measures in our early experiments. These measures fall into four categories: the ones based on (1) the *degree* (connections between a vertex and its neighbors); (2) the *closeness* (lengths of the shortest paths between a vertex and the other vertices); (3) the *betweenness* (frequency of the vertex on the shortest paths between other vertices); and (4) the *eigenvectors* (spectral decomposition of the adjacency matrix). We refer to Section 5.1.2 for the definitions. Note that measures such as PageRank, that belongs to the last category, assign a floating-point weight rather than a small integer and that requires some quantization for efficient compression and storage (Anh et al., 2001) while our degree-based model does not suffer from this shortcoming. This was another (practical) reason for us to keep the vertex weight definition simple. There was no mention of this issue in Blanco and Lioma (2007, 2012) but we think it does matter. Moreover, the use of these more complex centrality measures for term weighting did not lead to significantly better results than the degree-based ones, finding actually not specific to ad hoc IR but also observed in text categorization (Valle and Öztürk, 2011) and keyword extraction (Boudin, 2013).

SCORING FUNCTION Following the TF-IDF and BM25 scoring functions presented in Section 4.1.3, we defined our final model TW-IDF for a term t and a document $\boldsymbol{d}$ as follows:

$$TW\text{-}IDF(t, \boldsymbol{d}) = \frac{tw_{\boldsymbol{d}}(t)}{1 - b + b \times \frac{|\boldsymbol{d}|}{avdl}} \times \log \frac{N + 1}{df(t) + 0.5} \tag{4.16}$$

where $tw_{\boldsymbol{d}}(t)$ is the weight of the vertex (i. e. its indegree in practice) associated with the term t in the graph-of-words representation of the document $\boldsymbol{d}$. It corresponds to $\text{TW}_p \times \text{IDF}$ when defining TW_p by analogy with TF_p. b is set to 0.003 and does not require tuning – this constant value consistently produced good results across various collections. Its two orders of magnitude less than TF-IDF (0.20) and BM25 (0.75) can be explained by the structure of the graph-of-words. Since there is no weight on the edges, the indegree of a vertex does not increase linearly with the document length like the term frequency does: it is incremented only when a new context of occurrence appears. Thus, this requires less tilting (but still some pivoting as observed empirically, cf. Section 4.2.2.3 and Figure 4.1).

CONCAVE TRANSFORMATION Note that the model does not explicitly include a concave transformation (TW_k or TW_l defined by analogy with TF_k and TF_l). In fact, applying such a function worsens sometimes the results. This can be explained by the use of unweighted edges in the graph representation. Recall that the raw term frequency was historically dampened so that the difference between 1 and 2 and 100 and 101 in terms of tf values is much more important for the former (cf. Section 4.1.1.1). Here, in the context of a graph-of-words, an additional edge is added to the graph only if the context of occurrence for the term is new. This holds the same amount of information whatever the tw value already is, hence, the absence of such penalization for effectiveness reasons.

LOWER-BOUNDING REGULARIZATION Similarly, there is no explicit lower-bounding regularization as well. This can be explained by the two orders of magnitude less for b. In the original paper that introduced this normalization (Lv and Zhai, 2011b), it was noted that the pivoted document length normalization could result in null scores for very long documents (as $|d|$ becomes much larger than $avdl$, cf. Section 4.1.1.5). Here, this would happen in TW-IDF for documents a hundred times longer than for TF-IDF or BM25 and there is none in today's collections (in recent datasets, the size of the collection has been growing but not the document length). Anyway, if this were the case (e. g., books), then it would only require an additional composition with the function TW_δ that we omit for now in the implementation for efficiency reasons.

4.2.2 *Experiments*

In this section, we present the experiments we carried out to validate our proposed retrieval model TW-IDF. We first describe the datasets, the evaluation metrics, the platform and the considered models. We then report the results we obtained and discuss their interpretations.

4.2.2.1 *Datasets, evaluation metrics, platform and models*

DATASETS We used four standard TREC collections of documents: (1) `Disks 1&2`; (2) `Disks 4&5` (minus the Congressional Record); (3) `WT10G`; and (4) `.GOV2`. `Disks 1&2` includes 741,856 news articles from the Wall Street Journal (1987-1992), the Federal Register (1988-1989), the Associated Press (1988-1989) and the Information from the Computer Select disks (1989-1990) [1]. `Disks 4&5` contains 528,155 news releases from the Federal Register (1994), the Financial Times (1991-1994), the Foreign Broadcast Information Service (1996) and the Los Angeles Times (1989-1990)[1]. `WT10G` consists of 1,692,096 crawled pages from a snapshot of the Web in 1997 [2]. `.GOV2` corresponds to a crawl of 25,205,179 .gov sites in early 2004 [3]. Table 4.1 presents some basic statistics on the datasets – the document length corresponds to the number of terms in a document while the number of unique terms to the number of vertices in its graph-of-words representation (we give the average values per document over the entire collection).

EVALUATION METRICS We evaluated the retrieval models over these collections in terms of Mean Average Precision (MAP) and Precision at 10 (P@10) considering only the top-ranked 1000 documents for each run (cf. Section 2.3.3.1). Our goal was to propose a novel scoring function that improved both metrics. The statistical significance of improvement was assessed using the paired Student's t-test considering two-sided p-values less than 0.05 to reject the null hypothesis (cf. Section 2.3.4).

1. http://trec.nist.gov/data/docs_eng.html
2. http://ir.dcs.gla.ac.uk/test_collections/wt10g.html
3. http://ir.dcs.gla.ac.uk/test_collections/gov2-summary.htm

Statistic \ Dataset	Disks 1&2	Disks 4&5	WT10G	.GOV2
# documents	741,856	528,155	1,692,096	25,205,179
# unique terms	535,001	520,423	3,135,780	15,324,292
average # terms	237	272	398	645
average # vertices	125	157	165	185
average # edges	608	734	901	1185

Table 4.1 – Collection and document statistics on the four TREC datasets used in our experiments – the average values are given for a document.

RELEVANCE JUDGMENTS For each collection, we used a set of TREC topics (title only to mimic Web queries) and their associated relevance judgments[4]: (1) 51-200 for `Disks 1&2` (TREC1-3 Ad Hoc Tasks); (2) 301-450 and 601-700 for `Disks 4&5` (TREC 2004 Robust Track); (3) 451-550 for `WT10G` (TREC9-10 Web Tracks); and (4) 751-850 for `.GOV2` (TREC 2004-2006 Terabyte Tracks).

PLATFORM We used `Terrier` version 3.5 to index, retrieve and evaluate the retrieval models over the TREC collections. We extended the framework to accommodate our graph-based approach (mainly graph creation, indexing of term weight instead of term frequency and novel weighting models such as TW-IDF). We chose the Java `JUNG` library for graph representation and computation. We used `Hadoop` 1.0.3 to handle the distributed indexing of the `.GOV2` collection, which takes 426 GB of space in its original text form (compressed). For all the datasets, the preprocessing steps involved `Terrier`'s built-in stop word removal and Porter's stemming.

MODELS We considered four state-of-the-art scoring functions to compare our model with: TF-IDF, BM25, Piv+ and BM25+ (cf. Section 4.1.3). They all use pivoted document length normalization with a slope parameter b set by default to 0.20 for TF-IDF and its extension Piv+ and 0.75 for B25 and its extension BM25+. We will present results with default value for b and tuned one using 2-fold cross-validation (up to 4 decimals, odd vs. even topic ids, MAP maximization). For TW-IDF, b does not require any tuning and a default value of 0.003 was consistently giving good results across all collections. Regarding the parameter δ defined by Lv and Zhai (2011b) and used in the lower-bounding regularization, we did not tune it and used the default value (1.0) suggested in the original paper for both Piv+ and BM25+.

4.2.2.2 *Retrieval results*

Table 4.2 and Table 4.4 present the results we obtained for each of the four tasks. We separate results with untuned slope parameter b (Table 4.2) and with

4. http://trec.nist.gov/data/qrels_eng/index.html

Model	b	TREC1-3 Ad Hoc		TREC 2004 Robust	
		MAP	P@10	MAP	P@10
TF	none	0.0150	0.0273	0.0480	0.0867
IDF	none	0.0576	0.0960	0.1396	0.2040
TF_p	0.20	0.0482	0.1067	0.0596	0.1193
TF_p	0.75	0.0539	0.1807	0.0640	0.1289
TF_l	none	0.0583	0.0940	0.1591	0.3141
TF_k	none	0.0599	0.0913	0.1768	0.3269
$TF_{p \circ l}$	0.20	0.1471	0.3960	0.1797	0.3647
$TF_{k \circ p}$	0.75	0.1346	0.3533	0.2045	0.3863
$TF_{\delta \circ p \circ l}$	0.20	0.1470	0.3820	0.2002	0.3876
$TF_{\delta \circ k \circ p}$	0.75	0.1272	0.3240	0.2165	0.3956
TW	none	0.1502	0.3662	0.1809	0.3273
TW_p	0.003	**0.1576***	**0.4040***	**0.2190***	**0.4133***
TF-IDF	0.20	0.1832	0.4107	0.2132	0.4064
Piv+	0.20	0.1825	0.3813	0.2368	0.4157
BM25	0.75	0.1660	0.3700	0.2368	0.4161
BM25+	0.75	0.1558	0.3207	**0.2466**	0.4145
TW-IDF	0.003	**0.1973***	**0.4148***	0.2403	**0.4180***

Model	b	TREC9-10 Web		TREC 2004-2006 Terabyte	
		MAP	P@10	MAP	P@10
TF	none	0.0376	0.0833	0.0126	0.0617
IDF	none	0.0539	0.0729	0.0478	0.0651
TF_p	0.20	0.0531	0.1021	0.0228	0.0631
TF_p	0.75	0.0473	0.1000	0.0262	0.0550
TF_l	none	0.1329	0.2063	0.1412	0.4215
TF_k	none	0.1522	0.2104	0.1569	0.4188
$TF_{p \circ l}$	0.20	0.1260	0.1875	0.1853	0.4913
$TF_{k \circ p}$	0.75	0.1702	0.2208	0.2527	0.5342
$TF_{\delta \circ p \circ l}$	0.20	0.1436	0.2021	0.2055	0.5081
$TF_{\delta \circ k \circ p}$	0.75	0.1835	0.2354	0.2654	0.5369
TW	none	0.1430	0.1979	0.2081	0.5021
TW_p	0.003	**0.1946***	**0.2479***	**0.2828***	**0.5407***
TF-IDF	0.20	0.1430	0.2271	0.2068	0.4973
Piv+	0.20	0.1643	0.2438	0.2293	0.5047
BM25	0.75	0.1870	0.2479	0.2738	0.5383
BM25+	0.75	0.2026	0.2521	0.2830	0.5383
TW-IDF	0.003	**0.2125***	**0.2917***	**0.3063***	**0.5633***

Table 4.2 – Effectiveness results (MAP and P@10) of TW over TF and TF+ with untuned slope parameter b. Bold font marks the best performance in a column. * indicates statistical significance at $p < 0.05$ using the Student's t-test with regard to the baseline ($TF_{\delta \circ k \circ p}$ or BM25+) of the same column.

Model	TREC1-3 Ad Hoc	TREC 2004 Robust	TREC9-10 Web	TREC 04-06 Terabyte
TF_{pol}	0.1100	0.0635	0.0310	0.0340
TF-IDF	0.1000	0.0780	0.0340	0.0350
TF_{kop}	0.4180	0.3719	0.1740	0.3910
BM25	0.3600	0.3444	0.2505	0.3900
$TF_{\delta op ol}$	0.1548	0.0919	0.0407	0.0430
Piv+	0.1510	0.0863	0.0340	0.0400
$TF_{\delta ok op}$	0.5170	0.3780	0.2720	0.4120
BM25+	0.5510	0.3760	0.2345	0.4150

Table 4.3 – Values of slope parameter b tuned using cross-validation.

tuned one using cross-validation (Table 4.4). Table 4.3 indicates the tuned value for b that was learnt for each model on each task. We also indicate each time the results for the TF/TW component only and then for the full model (that takes into account IDF). That way, we can see the raw impact of TW as an alternative to TF. Statistical significance was computed with regard to the baseline model: $TF_{\delta ok op}$ for TW_p and BM25+ for TW-IDF (with tuning in Table 4.4). Note that the results for BM25 in Table 4.4 match the ones on the official Terrier website[5], assuring some reproducibility of the experiments.

TF VS. TF_x In the first two rows of Table 4.2, we present the results when using the raw term frequency (TF) or the normalized inverse document frequency (IDF) alone. We witness the popular belief that IDF is "superior" to TF in terms of contribution to the relevance. But in the next rows, we present the results for the normalized versions of TF, which we refer to as TF_x and we see in particular that the concave transformation (TF_l or TF_k) is crucial. Moreover, the effect of the document length penalization, even though milder, still adds up when combining the two, much more than when combining TF and IDF (still positive in all cases). This was a consistent finding that is rarely highlighted in our opinion and that really shows the importance of the TF normalizations and their composition.

TW VS. TF Table 4.2 also reports results for TW, which corresponds to a raw graph-based term weight without document length penalization. Surprisingly, it is already outperforming TF_{pol}, the TF component of TF-IDF. This was one of our early models and this finding encouraged us to pursue further and develop TW-IDF. When taking into account the document length with a small tilting, i. e. TW_p, we obtain a model that is significantly better than its counterparts, even with lower-bounding regularization.

5. http://terrier.org/docs/v3.5/trec_examples.html#paramsettings

Model	b	TREC1-3 Ad Hoc		TREC 2004 Robust	
		MAP	P@10	MAP	P@10
$TF_{p \circ l}$	tuned	0.1555	**0.4167**	0.1946	0.3867
$TF_{k \circ p}$	tuned	0.1521	0.3767	0.2176	0.4145
$TF_{\delta \circ p \circ l}$	tuned	0.1481	0.3873	0.2082	0.4036
$TF_{\delta \circ k \circ p}$	tuned	0.1319	0.3240	**0.2246**	**0.4185**
TW	none	0.1502	0.3662	0.1809	0.3273
TW_p	0.003	**0.1576***	0.4040	0.2190	0.4133
TF-IDF	tuned	0.1936	**0.4340**	0.2261	0.4193
Piv+	tuned	0.1841	0.3840	0.2436	0.4229
BM25	tuned	0.1893	0.4080	0.2502	**0.4382**
BM25+	tuned	0.1603	0.3340	**0.2547**	0.4349
TW-IDF	0.003	**0.1973***	0.4148	0.2403	0.4180

Model	b	TREC9-10 Web		TREC 2004-2006 Terabyte	
		MAP	P@10	MAP	P@10
$TF_{p \circ l}$	tuned	0.1847	**0.2729**	0.2336	0.5611
$TF_{k \circ p}$	tuned	0.1915	0.2583	0.2807	**0.5785**
$TF_{\delta \circ p \circ l}$	tuned	0.1949	**0.2729**	0.2463	0.5658
$TF_{\delta \circ k \circ p}$	tuned	**0.1997**	0.2708	**0.2894**	0.5758
TW	none	0.1430	0.1979	0.2081	0.5021
TW_p	0.003	0.1946	0.2479	0.2828	0.5407
TF-IDF	tuned	0.2031	0.2854	0.2589	0.5732
Piv+	tuned	0.2117	0.2875	0.2667	0.5725
BM25	tuned	0.2104	**0.3210**	0.3046	**0.5899**
BM25+	tuned	**0.2169**	0.2771	**0.3085**	**0.5906**
TW-IDF	0.003	0.2125	0.2917	0.3063	0.5633

Table 4.4 – Effectiveness results (MAP and P@10) of TW over TF and TF+ with tuned slope parameter b. Bold font marks the best performance in a column. * indicates statistical significance at $p < 0.05$ using the Student's t-test with regard to the baseline ($TF_{\delta \circ k \circ p}$ or BM25+) of the same column.

TW-IDF VS. BM25 Table 4.2 clearly establishes the significant performance of TW-IDF over BM25 (and a fortiori TF-IDF). With tuning of the slope parameter b (Table 4.4), TW-IDF still outperforms BM25 on the Web datasets in terms of MAP. Moreover, tuning of parameters is known to be costly and requires a set of queries with associated relevance judgments. For a new collection of documents without such training set (like most of real-world datasets), TW-IDF appears more robust and should produce better results than BM25.

TW-IDF VS. BM25+ Table 4.2 compares our novel models to extensions of TF-IDF and BM25 recently proposed by Lv and Zhai (2011b). Again, without tuning, TW_p and TW-IDF significantly outperform the other models. This shows how well the graph-of-words encompasses concavity, document length penalization and lower-bounding regularization compared to the traditional bag-of-words. This allows our model to require less parameterization and to show more robustness across collections.

TUNED SLOPE PARAMETER In all fairness, we also reported in Table 4.4 results for tuned Piv+ and BM25+. These are the only cases where TW-IDF performed comparably. One has to take into consideration that we are challenging a well-established model with a novel approach and in this last case, tuned state-of-the-art scoring functions. We shall see in Section 4.2.3.2 our thoughts on how people could further improve TW-IDF.

4.2.2.3 *Likelihood of relevance and likelihood of retrieval*

We mentioned in Section 4.2.1.2 that, *in principle*, TW-IDF should include an explicit penalization over the document length like most of the traditional scoring functions such as TF-IDF or BM25. We then turned to seeking empirical evidence to see if this was the case in *practice*. We already witnessed in the experiments that TW_p was giving much better results than the raw TW in terms of MAP and P@10.

PLOT Following Singhal et al.'s finding that a good scoring function should retrieve documents of all lengths with similar chances to their probability of relevance (Singhal, Buckley, et al., 1996), we compared the retrieval pattern of TW-IDF (with and without regularization) against the relevance pattern extracted from the golden judgments. We followed the binning analysis strategy proposed by Singhal, Buckley, et al. (1996) and plot in Figure 4.1 the three patterns against all document lengths on the `WT10G` collection. We set the bin size to 5,000 and display the x-axis with logarithmic scale following Lv and Zhai (2011b). We also plot the retrieval patterns of TF-IDF, Piv+, BM25 and BM25+ (with the tuned slope parameter b from Table 4.3). The regression curves were obtained using Locally Weighted Scatterplot Smoothing (LOWESS) (Cleveland, 1979) through the `geom_smooth(method="loess")` ggplot2 geometric object[6].

6. http://docs.ggplot2.org/0.9.3.1/geom_smooth.html

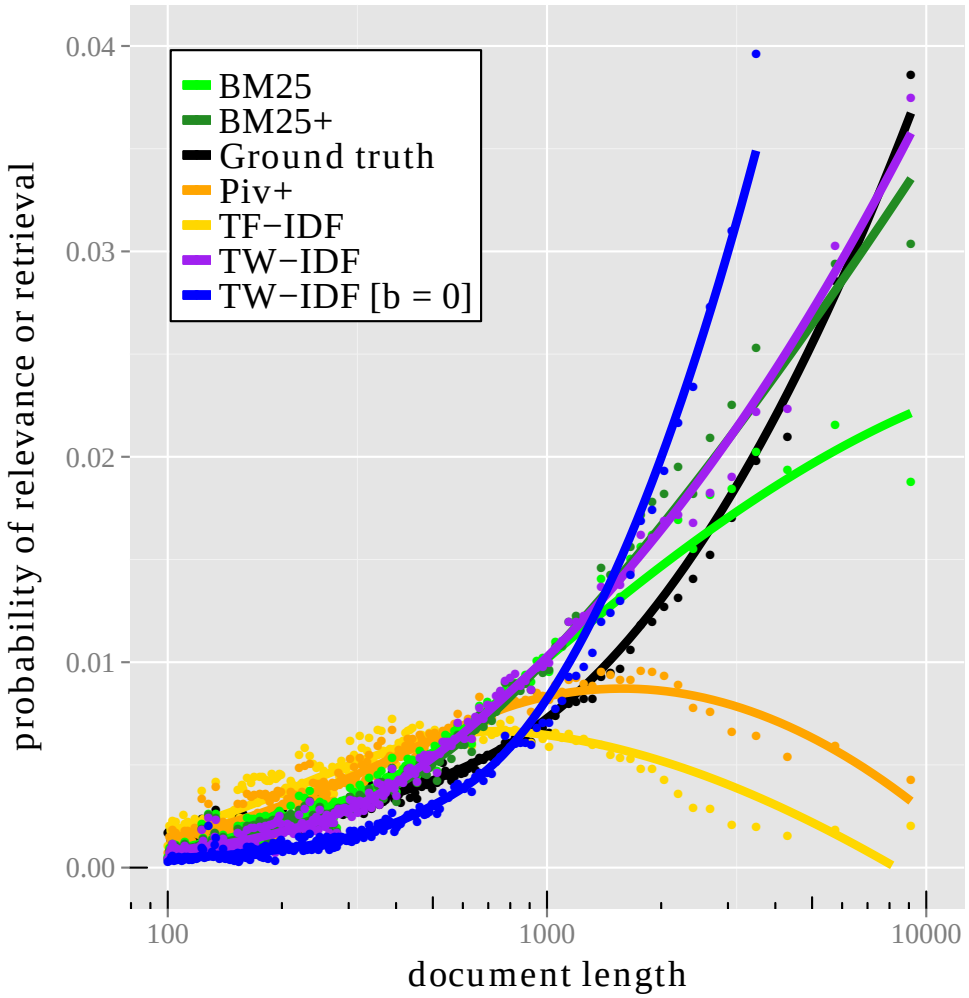


Figure 4.1 – Probability of relevance and probability of retrieval vs. document length on the WT10G collection.

INTERPRETATION The plot shows that TW-IDF without document length penalization ($b = 0$, blue curve) favors more than it should longer documents and less than it should shorter documents, thus requiring pivoting and tilting ($b = 0.003$, purple curve). This empirically confirms our previous analysis that TW-IDF needs document length penalization, even if it is at a smaller scale than TF-IDF and BM25, which was not explored by Blanco and Lioma (2007, 2012). The plot also shows clearly that TW-IDF with pivoted document length normalization matches better the likelihood of relevance (black curve) than any other retrieval functions, even BM25+ (dark green curve) that was specifically designed to overcome the over-penalization for very long documents compared to BM25 (green curve) as introduced by Lv and Zhai (2011b). This is yet another advantage of our scoring function TW-IDF in terms of **robustness against varying document lengths**. TF-IDF (gold curve) and Piv+ (orange curve) are definitely not tailored for varying document lengths, which suggests that the concave transformation TF_k might be more suited than TF_l for longer documents.

4.2.3 *Highlights, current limitations and future work*

By representing documents as unweighted undirected graph-of-words, we are able to extract more meaningful terms weights than traditional term frequencies, which results in a better scoring function (TW-IDF) compared to the classical TF-IDF and BM25. In particular, the unweighted indegree-based term weight proved to be a valid alternative to the dampened term frequency by only taking into account the number of different contexts of co-occurrences regardless of the frequency of each context. Moreover, the constant and two order of magnitude smaller value for the slope parameter b (0.003) alleviates the costly parameter tuning and the additional lower-bounding regularization. Finally, our proposed model boasted a better robustness against varying document lengths. We describe next the current limitations of our scoring function due mostly to the scale of the task at hand and propose research directions for future work.

4.2.3.1 *Normalized weighted node degree*

Experiments showed that an unweighted graph-of-words representation resulted in better retrieval performances. But intuitively, if we were to weight the edges with the number of times the two terms co-occur in the document, we would capture the strength of these relationships. However, taking the raw weighted node degree as term weight would be equivalent to considering the tf value up to the constant multiplier W (except for edge cases like starting and ending terms and self-loop avoidance). And subsequent normalizations at query time would only results in models equivalent to Piv+ and BM25+. Nevertheless, we could in theory include some normalizations when computing the weighted node degree by penalizing each edge weight with some concave transformation or taking into account the document length or related global graph properties. But this second layer of normalizations, at the node level, should occur before the tw value is computed and thus prior to the indexing. Therefore, a parameterized normalization would involve the tuning at query time of a parameter set at indexing time, which turned out to be infeasible for us in practice as the indexing of one collection for one set of parameter values was already taking hours to days, even on a cluster of distributed machines. And we would need to try hundreds of different values for a given parameter with no guarantee that the optimization problem is convex (just like for tuning the slope parameter b) nor that the parameters are independent.

4.2.3.2 *TW-IDW, challenging the document independence assumption*

Through the graph-of-words representation, we challenged the *term independence assumption* made behind the bag-of-words model and proposed TW as an alternative to TF. In ad hoc IR and more generally in TM, there is actually another assumption made, this time at the collection level: the **document independence assumption**. Indeed, we commonly assume that each document in a collection is independent of one another. In particular, when computing the

document frequency of a term for the IDF component, we consider the collection as a **bag-of-documents**. In the context of a search engine, taking into account relations between documents could improve search and user experience by providing more diversity among the top results for instance. Additionally, instead of just counting the number of documents in which a term appears to assess its specificity, we might want to consider whether a term appears in a set of related documents or not by representing the collection as a **graph-of-documents** and explore its communities. Defining the *document weight* of a term in the context of a graph representation of a collection to challenge TF$\times$IDF with TW$\times$IDW is a research issue in itself and is beyond the scope of this dissertation, especially if we want to weight the edges between documents according to some similarity measures based on their graph-of-words representation through the definition of relevant graph kernels for instance.

5

EXTRACTING KEYWORDS FROM GRAPH-OF-WORDS

In this chapter, we extract keywords from graph-of-words representations of documents. The idea in itself is not novel as other researchers have wondered in the past what it means for a node of a graph-of-words to be a keyword. Therefore, we first review the main unsupervised approach in Keyword Extraction (KwE), which states that central nodes make good keywords. We then pursue with our proposed approach, which extracts not only central but also densely connected sets of nodes that we call "corewords" as we have observed in practice that this technique leads to the selection of keywords closer to what authors would choose since they prefer keyphrases over keywords, i. e. subgraphs of graph-of-words over isolated nodes.

5.1 CENTRAL NODES MAKE GOOD KEYWORDS

Many works in the past decade have reported that nodes with high centrality in graph-of-words representations of documents usually correspond to the keywords that a human would pick for the documents. Therefore, we first need to define the various vertex centrality measures from graph theory before providing an overview of the publications on the subject.

5.1.1 *Preliminary graph definitions*

GRAPH Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a **graph** (a. k. a. a network), $\mathcal{V}$ its set of **vertices** (a. k. a. nodes) and $\mathcal{E}$ its set of **edges** (a. k. a. arcs or links). Edges can be *directed*, in which case networks are referred to as digraphs, and/or *weighted* as seen in Section 3.2.2 and also *labeled*, e. g., covalent bonds in chemical compounds. By abusing the notation, when considering nodes from a graph, we will write $v \in \mathcal{G}$ rather than $v \in \mathcal{V}$. We denote by n the number of vertices ($n = |\mathcal{V}|$) and m the number of edges ($m = |\mathcal{E}|$). A graph can represent anything, from a social network to a power grid or in our case a textual document.

WALK VS. PATH A **walk** is a sequence of vertices and edges, where each edge's endpoints are the preceding and following vertices in the sequence. It can be directed and/or weighted just like the edges. A **random walk** is a walk where each next edge in the sequence is chosen at random from the last endpoint, according to the outgoing edges' weights in a weighted network (Pearson, 1905). A **path** is a walk in which all the vertices are distinct (and a fortiori the edges as well). The **length** of a walk/path is the number of edges between the starting

and ending vertices in the unweighted case and the sum of the edge weights in the weighted case.

CONNECTIVITY A graph is **connected** if there are paths between every pair of vertices, otherwise it is said to be disconnected. A digraph is **strongly connected** if there are directed paths between every pair of vertices, weakly connected if there are only undirected paths between every pair of vertices and disconnected otherwise. A **connected component** of a graph $\mathcal{G}$ is a *maximal* connected subgraph of $\mathcal{G}$, i. e. it cannot be augmented (by adding new edges or nodes) without losing this property.

SHORTEST PATH The **shortest path** between two vertices u and v is the path of minimum length. Its length $d(u, v)$ is called the *geodesic distance* from vertex u to v. Note that the shortest path may not be unique, may not exist because of a disconnected network or a non-strongly connected directed network or may only be one-way in case of a digraph. Also, for weighted graphs, depending on the meaning of the edge weights (e. g., number of co-occurrences in the case of a graph-of-words), one might need to take the inverse of the weights when computing the length of the paths so as to favor the strongest path(s).

COMPLETENESS A (di)graph is **complete** if there are (directed) edges between every pair of vertices. An induced subgraph, i. e. a subgraph of the original network containing all the original edges between a given set of nodes, is a **clique** if it is complete, i. e. all the vertices of the subgraph share an edge with each other.

5.1.2 *Vertex centrality measures*

DEGREE The **degree** $\deg_{\mathcal{G}}(v)$ of a node v is defined as the sum of the weights of its *incoming* edges in the general case of a directed weighted network. For undirected unweighted graphs, this simply corresponds to the number of adjacent neighbors (i. e. the number of distinct contexts of co-occurrence for a graph-of-words). For undirected weighted ones, this also takes into account the strength of the relationship (e. g., the number of co-occurrences between two terms for a graph-of-words). Note that we restrict the definition to in-links for the directed case to keep the definition of a k-core unidimensional (cf. Section 5.2.1) and we refer to the work of Giatsidis et al. (2011a) for the bidimensional case. Since we can choose between forward and backward edges in our graph-of-words, we can alternatively consider in-links and out-links, just not both at the same time. We also denote by $\Delta(\mathcal{G})$ the maximum degree of $\mathcal{G}$, i. e. $\Delta(\mathcal{G}) = \max_{v \in \mathcal{V}}(\deg_{\mathcal{G}}(v))$.

CLOSENESS The **closeness** $C_C(v)$ of a node v is defined as the inverse of the total distances from every other node to v in the graph:

$$C_C(v) = \frac{1}{\sum_{u \neq v} d(u, v)} \tag{5.1}$$

In the literature and in graph libraries, we have found definitions for digraphs with distances from v and to v. The original paper from Bavelas (1950) defined it with distances *from* v but it was later referenced by Beauchamp (1965) with distances *to* v. In our opinion, this mostly depends on the application. For instance, in the case of the Web, pages have less control over their in-links and therefore it might make more sense to consider distances to v in order to measure how central a Web page is, at least in terms of closeness. For graph-of-words, we generally recommend undirected networks anyway and in any case, we can alternatively consider in-links and out-links.

BETWEENNESS The **betweenness** $C_B(v)$ of a node v is defined as the fraction of shortest paths from all vertices (except v) to all others (except v) that pass through v:

$$C_B(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{5.2}$$

where σ_{st} is the total number of shortest paths from node s to node t and $\sigma_{st}(v)$ is the number of those paths that pass through v. Note that in the general case of a directed network, σ_{st} might be different from σ_{ts}. Essentially, the betweenness quantifies the number of times a node acts as a bridge along the shortest path between two other nodes. The concept was formalized by Freeman (1977).

CLUSTERING COEFFICIENT The **clustering coefficient** $CC(v)$ of a node v is defined as the fraction of neighbors that are also connected together (at most $\binom{\deg_{\mathcal{G}}(v)}{2}$ for undirected networks), i. e. the fraction of actual triangles the node belongs to in an undirected graph:

$$CC(v) = \frac{2|\{e = (s,t) \in \mathcal{E},\ (s,v) \in \mathcal{E} \wedge (v,t) \in \mathcal{E}\}|}{\deg_{\mathcal{G}}(v)(\deg_{\mathcal{G}}(v) - 1)} \tag{5.3}$$

Essentially, the clustering coefficient quantifies how close the neighbors of a node are to being a *clique*, its "cliquishness" as presented by Watts and Strogatz (1998), which makes it a measure of centrality and *cohesion* (Wasserman and Faust, 1994, page 249) as well.

SMALL-WORLDLINESS As mentioned in Section 3.2.4, statistical graph-of-words have been reported to be small worlds, i. e. networks whose nodes are highly clustered (high *clustering coefficient*) yet the path length between them is small (short *characteristic path length*).

The **characteristic path length** $L(\mathcal{G})$ of a graph $\mathcal{G}$ is defined as the average shortest distance between all pairs of nodes ($\binom{n}{2}$ in total for undirected networks), assuming a (strongly) connected graph:

$$L(\mathcal{G}) = \frac{2}{n(n-1)} \sum_{s \neq t} d(s,t) \tag{5.4}$$

Some nodes contribute more than others to the *small-worldliness* of a graph and one way to measure their local contributions is to compute the two metrics on the original network and on a network without the node of interest (and its adjacent edges). Since the removal of one node has little effect on the overall clustering coefficient (Matsuo et al., 2001b), people have focused in practice on the difference in characteristic path length, defining the **small-worldliness** $C_{SW}(v)$ of a node v as:

$$C_{SW}(v) = L(\mathcal{G} \setminus \{v\}) - L(\mathcal{G}) \tag{5.5}$$

where $\mathcal{G} \setminus \{v\}$ is the graph $\mathcal{G}$ without the node v (and a fortiori without its adjacent edges).

ECCENTRICITY The **eccentricity** $\varepsilon(v)$ of a node v is defined as the length of the longest shortest path(s) from any other nodes to v, i. e. the maximum geodesic distance to v:

$$\varepsilon(v) = \max_{u \neq v} d(u,v) \tag{5.6}$$

Therefore, its inverse is a measure of centrality. The main difference with closeness is that a node with a low eccentricity value is relatively close to *every* other node whereas a node with a high closeness value is on *average* close to all the other nodes. In practice, because of "outliers" that can artificially lower the centrality, e. g., high cost weight on all edges to a particular node, it has been less used than closeness, including in KwE from graph-of-words.

EIGENVECTOR CENTRALITY So far, the definition of a centrality measure for a given node v has been independent of the centrality values of its neighbors, considering basically that each neighbor contributes equally to the centrality of v – apart from the edge weight but we consider for now unweighted networks. However, intuitively, an in-link from a central node carries a stronger endorsement in terms of its own centrality, e. g., connections to influential people might lend a person more influence than connections to less influential people. We thus require knowing the centrality values of the neighbors to compute the centrality value of a node, values that might even depend on the value of interest in case of bidirectional links.

This results in a recursive definition of the **eigenvector centrality** $e(v)$ of a node v:

$$\lambda e(v) = \sum_{u \in \mathcal{N}_I(v)} e(u) = \sum_{u \in \mathcal{G}} A_{uv} e(u) \tag{5.7}$$

where λ is a constant required so that the equations have a non-zero solution (Bonacich, 1987), $\mathcal{N}_I(\cdot)$ is the set of neighbors pointing *to* a node and $\boldsymbol{A}$ is the

(binary) **adjacency matrix** of graph $\mathcal{G}$, i. e. A_{ij} encodes the existence of an edge from node i to node j. Note that in the literature there is a discrepancy between the textual definition that states we should consider nodes pointing to v and the corresponding formula where A_{vu} is employed instead, e. g., see Newman (2003, p. 43). Since the original case was for undirected networks, it did not matter but we think A_{uv} makes more sense and it also aligns with PageRank, a close measure as we will see next and first applied to the Web, a *directed* graph. In matrix notation, the centrality can be expressed as:

$$\lambda \boldsymbol{e} = \boldsymbol{A}^\top \boldsymbol{e} \tag{5.8}$$

where $\boldsymbol{e}$ is the vector of centralities for all nodes and thus corresponds to the eigenvector of $\boldsymbol{A}^\top$ associated to the eigenvalue λ. As reported by Newman (2003, pp. 43-44), provided that the network is connected, the Perron–Frobenius theorem tells us that there is only one eigenvector with all weights non-negative, which is the unique eigenvector corresponding to the largest eigenvalue and it can be found for instance by repeated multiplication of the adjacency matrix (i. e. by *power iteration*) into any initial non-zero vector (in practice a uniform vector).

PAGERANK PageRank is a variant of the eigenvector centrality. It was introduced by Page et al. (1999) and incorporated in the earliest versions of Google's search engine (Brin and Page, 1998), which made this centrality measure extremely popular, both in industry and academia. It was used to model the navigation of a "random surfer" on the Web as a sequence of visited pages, i. e. a random walk in the corresponding graph.

The **PageRank** score $PR(v)$ of a node v is defined as:

$$PR(v) = \frac{1-d}{n} + d \times \sum_{u \in \mathcal{N}_I(v)} \frac{PR(u)}{|\mathcal{N}_O(u)|} \quad \text{subject to } \|PR\|_1 = 1 \tag{5.9}$$

where d is a damping factor ($\in [0, 1]$, typically 0.85) and $\mathcal{N}_O(\cdot)$ is the set of neighbors pointed *by* a node.

The three differences with the original eigenvector centrality are (1) the additional term controlled by the damping factor, interpreted as the probability $(1 - d)$ of the surfer to jump to any random page at any given moment (with probability $1/n$); (2) the normalization of the adjacency matrix terms by the number of outgoing links; and (3) the normalization of the PageRank vector at each step to make it a probability distribution, which results in the PageRank score being not only the centrality of the node but also the probability of visiting the given Web page at random.

HITS In parallel of the development of PageRank, Kleinberg (1999) proposed Hyperlink-Induced Topic Search (HITS) as a measure of the influence of a node in a network. Contrary to previous centrality measures, he made a distinction between nodes pointed by a lot of other nodes, the so-called **authorities**, and nodes pointing to a lot of other nodes, the **hubs**. For instance, in the case of

a citation network of publications, seminal papers would be authorities, Ph.D. dissertations hubs and seminal surveys both. Therefore, while still based on a recursive definition, each node has two scores: one for authority that depends on the hub scores of its incoming neighbors and one for hubness that depends on the authority scores of its outgoing neighbors. Indeed, a seminal paper should be cited by a lot of other papers, including surveys, and a survey should reference a lot of papers, including seminal papers.

The **HITS** authority and hub scores $a(v)$ and $h(v)$ of a node v are defined as:

$$a(v) = \sum_{u \in \mathcal{N}_I(v)} h(u) \tag{5.10}$$

$$h(v) = \sum_{u \in \mathcal{N}_O(v)} a(u) \tag{5.11}$$

In matrix notation, this gives $\boldsymbol{a} = \boldsymbol{A}^\top \boldsymbol{h}$ and $\boldsymbol{h} = \boldsymbol{A}\boldsymbol{a}$ and by combining the two equations, this results in $\boldsymbol{a} = \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{a}$ and $\boldsymbol{h} = \boldsymbol{A}\boldsymbol{A}^\top \boldsymbol{h}$. Therefore, $\boldsymbol{a}$ is by definition the right singular vector of the adjacency matrix $\boldsymbol{A}$ and the eigenvector of the covariance matrix $\boldsymbol{A}^\top \boldsymbol{A}$ while $\boldsymbol{h}$ is the left singular vector of the adjacency matrix $\boldsymbol{A}$ and the eigenvector of the Gram matrix $\boldsymbol{A}\boldsymbol{A}^\top$.

COMPLEXITY In all cases, we need $\mathcal{O}(n+m)$ space to store the graph. Brandes (2001) proposed an algorithm for computing the betweenness of all vertices, which also covers closeness, small-worldliness and eccentricity since they all involve shortest paths, that requires $\mathcal{O}(nm)$ time for unweighted graphs and $\mathcal{O}(nm + n^2 \log n)$ time for weighted ones. Computing the clustering coefficients of all vertices requires $\mathcal{O}(n\Delta(\mathcal{G})^2)$ time when intersecting adjacency lists (for each node, we need to test for the existence of at most $\binom{\Delta(\mathcal{G})}{2}$ edges). Computing the eigenvector centrality, PageRank and HITS scores of all vertices iteratively requires $\mathcal{O}(m)$ time at each iteration.

5.1.3 *Literature review*

In the literature, there has been a few works on extracting keywords from graph-of-words representations of document and in particular on defining what makes a node a good keyword – high centrality being the most adopted approach. The earliest work we found was the one from Ohsawa et al. (1998) – the one that actually pioneered the graph-of-words representation to the best of our knowledge – where keywords are defined as nodes that tie and hold **clusters** together. Then, Matsuo et al. (2001b) proposed to extract the nodes that contribute the most to the **small-worldliness** of the network by computing the difference in **characteristic path length** between the graphs with and without the node, idea further explored by Zhu et al. (2003) as well. Mihalcea and Tarau (2004) came up with the idea of using **PageRank** to score each node's contribution in the context of a graph-of-words, popularizing the graph representations of text in TM, and subsequently revisited in several other papers (Wan and Xiao, 2008; Weiming Liang et al., 2009; Wan and Xiao, 2010;

R. Wang et al., 2015). Palshikar (2007) argued in favor of the inverse of the **eccentricity** to score terms, comparing it with **closeness** and **betweenness** just like Abilhoa and de Castro (2014) who also simply considered the node **degree**. Litvak and Last (2008) explored **HITS** as opposed to PageRank to try to identify a difference between authoritative and hub terms in a document. Boudin (2013) and Lahiri, S. R. Choudhury, et al. (2014) both wrote a survey of the various centrality measures to be used in the context of KwE. Schluter (2014) recently tried to explain what make central nodes good keywords. Additionally, on *semantic* graph-of-words, Grineva et al. (2009) defined a score based on its **community's density** and its community's informativeness while Tsatsaronis et al. (2010) stuck to more classical eigenvector centrality measures, namely PageRank and HITS.

BASELINES In our research, we focused on PageRank and HITS as baselines for various reasons. PageRank has been repeatedly reported as working best at extracting keywords from graph-of-words and HITS is an interesting variant where there could be a difference between authoritative and hub terms in a document. They both take into account the centrality values of the neighbors when computing the centrality of a node, which can explain why they have performed better in a wide variety of tasks and they are also cheaper to run – even if our graph-of-words have only a few thousands nodes at most, we potentially apply the extraction procedure on thousands to millions of documents.

5.2 COMMUNITIES OF CENTRAL NODES MAKE BETTER KEYWORDS

In this section, we first present the concept of graph degeneracy that takes into account not only how central nodes are but also how cohesive their neighborhoods are. We then capitalize on that notion to propose a novel method to extract the most representative words of a document – its *corewords* – and compare it with baseline approaches based solely on centrality measures. Finally, we conclude with current limitations and potential future work extensions.

5.2.1 *Graph degeneracy*

The idea of a *k-degenerate* graph comes from the work of Bollobás (1978, page 222) that was further extended by Seidman (1983) into the notion of a *k-core*, which explains the use of **degeneracy** as an alternative denomination for **k-core** in the literature. Henceforth, we will be using the two terms interchangeably.

DEFINITION A subgraph $\mathcal{H}_k = (\mathcal{V}', \mathcal{E}')$, induced by the subset of vertices $\mathcal{V}' \subseteq \mathcal{V}$ (and a fortiori by the subset of edges $\mathcal{E}' \subseteq \mathcal{E}$), is called a **k-core** or a *core of order k* iff $\forall v \in \mathcal{V}'$, $\deg_{\mathcal{H}_k}(v) \geq k$ and $\mathcal{H}_k$ is the maximal subgraph with this property, i.e. it cannot be augmented without losing this property. In other words, the k-core of a graph corresponds to the set of maximal connected

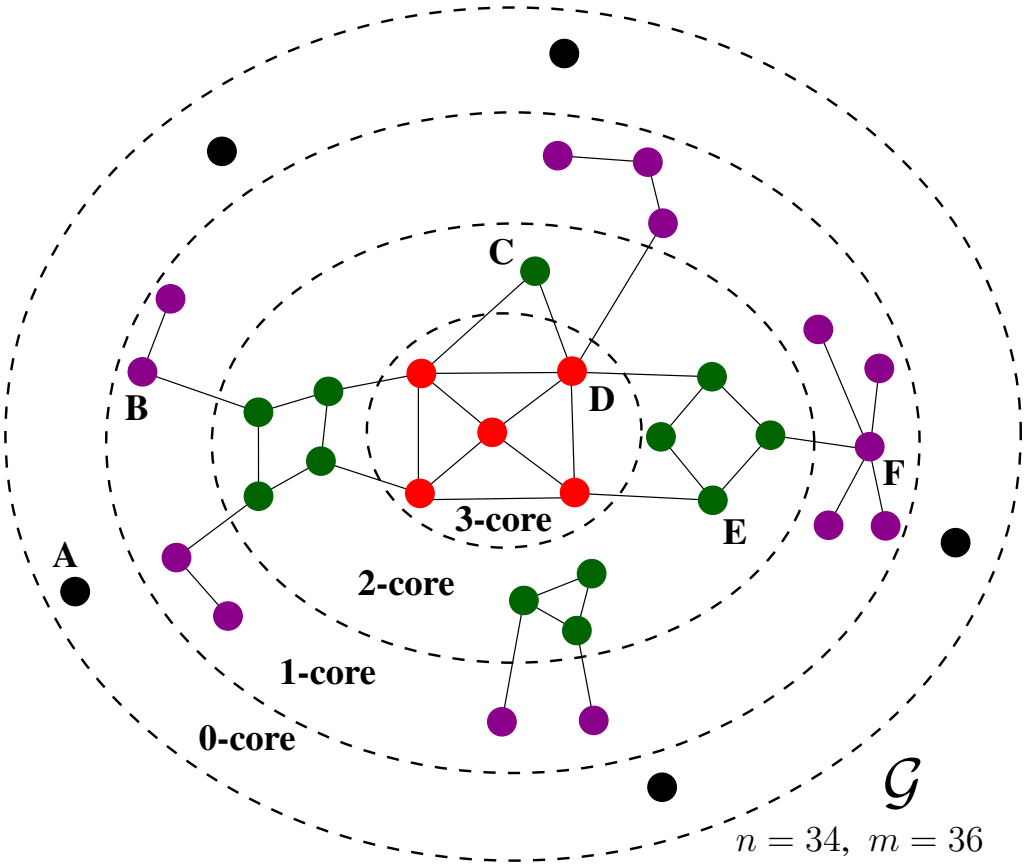


Figure 5.1 – Illustration of the decomposition of a given undirected unweighted graph into nested subgraphs of increasing core order. Node color indicates the highest core a vertex belongs to: black for the 0-core, purple for the 1-core, green for the 2-core and red for the 3-core (main core).

subgraphs whose vertices are at least of degree k within the subgraph they belong to. The **core number** $\text{core}(v)$ of a vertex v is the highest order of a core that contains this vertex. The core of maximum order is called the **main core** and its core number is denoted by $\text{core}(\mathcal{G})$. The set of all the k-cores of a graph (from the 0-core to the main core) forms what is called the **k-core decomposition** of a graph, a sequence of nested subgraphs of increasing core order k.

ILLUSTRATION Figure 5.1 illustrates the decomposition of a given undirected unweighted graph $\mathcal{G}$ of 34 vertices and 36 edges into nested cores of order 0, 1, 2 and 3. In this example, $\text{core}(A) = \deg_{\mathcal{G}}(A) = 0$, $\text{core}(B) = 1$, $\deg_{\mathcal{G}}(B) = 2$, $\text{core}(F) = 1$, $\deg_{\mathcal{G}}(F) = 5$, $\text{core}(D) = 3$, $\deg_{\mathcal{G}}(D) = 6$, $\text{core}(\mathcal{G}) = 3$ and $\Delta(\mathcal{G}) = 6$. We note in particular that (a) the cores are nested, i. e. $\forall j > i,\ \mathcal{H}_j \subseteq \mathcal{H}_i$ and (b) they do not need to form a single connected component (e. g., the 2-core in dark green is composed of two disconnected components). This figure has been manually created using the `Ipe` extensible drawing editor.

INTERPRETATION Centrality measures like the node degree or the PageRank score aforementioned tend to favor vertices with a lot of neighbors (at least in the unweighted case) but it does not imply that their neighbors need to be connected with each other as well (classically a vertex at the center of a *star* or *wheel* pattern in a graph). A good example of that difference is the node F in Figure 5.1 of relatively high degree (5) but of lower core number (1) in comparison with other vertices of lower degree like C (degree of 2, core number of 2) or of similar degree like D (degree of 6, core number of 3). Degeneracy captures not only how central a node is but also how connected its neighborhood is, property known as the **cohesion** of a graph (Wasserman and Faust, 1994, page 249).

COMPLEXITY Batagelj and Zaveršnik (2003) proposed a **linear** algorithm ($\mathcal{O}(n+m)$ time, $\mathcal{O}(n)$ space) for the k-core decomposition of an *unweighted* graph. The main idea is to remove the vertex of lowest degree (in the remaining subgraph) at each step and decrease the degree of its adjacent neighbors by one. The vertices are initially sorted in linear time using *bin sort* since there are at most $\Delta(\mathcal{G})+1$ distinct values for the degrees and $\Delta(\mathcal{G}) < n$. Because the algorithm only visits each edge twice and updating the degree can be achieved in constant time (by moving the node to the adjacent bin), the complexity is linear in the number of nodes and edges.

However, the algorithm no longer works for weighted edges since updating the degree of a neighbor when removing the vertex of lowest degree might change its degree by more than one. Thus, keeping the vertices in increasing order of degree requires a more complex data structure than the binning strategy used in the unweighted case (and also because the values of the degrees can be potentially any real number, which does not scale well with bin sort). Batagelj and Zaveršnik (2002) proposed the use of a *min-oriented binary heap* to retrieve the vertex of lowest degree in logarithmic time ($\mathcal{O}(\log n)$) at each step. We note that building the heap from an existing array of elements can be done in linear time (Cormen et al., 2009, page 159) and not trivially in linearithmic time as counted by Batagelj and Zaveršnik. Thus, the overall complexity is $\mathcal{O}(n + m \log n)$ in time and $\mathcal{O}(n)$ in space. Also note that this heap needs to support the *decrease-key* operation in order to update the priority of a node as its degree changes. This can be achieved in logarithmic time as well ($\mathcal{O}(\log n)$) providing a hash table for fast access to any node in the heap ($\mathcal{O}(n)$ additional space). Indeed, with constant time access ($\mathcal{O}(1)$) and logarithmic time call to the internal *bubble-up* routine ($\mathcal{O}(\log n)$), any node can be moved in the heap to its rightful position whenever its priority is decreased in a min-oriented heap. This is the data structure that we implemented in our experiments. Batagelj and Zaveršnik also mentioned the use of a Fibonacci heap (Fredman and Tarjan, 1987) that guarantees linear time construction ($\mathcal{O}(n)$) and amortized constant time update ($\mathcal{O}(1)$) for an overall complexity of $\mathcal{O}(n \log n + m)$ in time but the structure is more complex to maintain for little to no gain in practice for us (graphs with a number of vertices and edges in the order of a thousand at most).

APPLICATIONS Graph degeneracy has been used for (1) *visualization* (Alvarez-Hamelin et al., 2005; Ahmed et al., 2007); (2) *generation* (Baur et al., 2007); (3) *influential spreaders identification* (Kitsak et al., 2010); (4) *community detection* (Giatsidis et al., 2011b); (5) *modeling* of engagement dynamics (Malliaros and Vazirgiannis, 2013); (6) community-preserving *anonymization* (Carpineto and Romano, 2013; Assam et al., 2014; Rousseau, Casas-Roma, et al., 2015) and *generalization* (Casas-Roma and Rousseau, 2015) of networks and applied in addition to (1) *dynamic* (Miorandi and De Pellegrini, 2010; R.-H. Li et al., 2014); (2) *directed* (Giatsidis et al., 2011a); (3) *weighted* (Garas et al., 2012; Eidsaa and Almaas, 2013); (4) *signed* (Giatsidis, Vazirgiannis, et al., 2014); and (5) *uncertain* (Bonchi et al., 2014) graphs. In all works, researchers capitalize on its efficient way to capture cohesive sets of nodes, i. e. **communities** of similar entities or at least its core members compared to maybe more effective but far more expensive techniques such as clustering in polynomial time (Girvan and Newman, 2002; Frey and Dueck, 2007) or finding maximal cliques in exponential time (Bron and Kerbosch, 1973).

5.2.2 *Coreword extraction: retaining the nodes from the main core*

In this section, we define the document representation and the keyword extraction technique we came up with for KwE, which we introduced in (Rousseau and Vazirgiannis, 2015).

5.2.2.1 *Model definition*

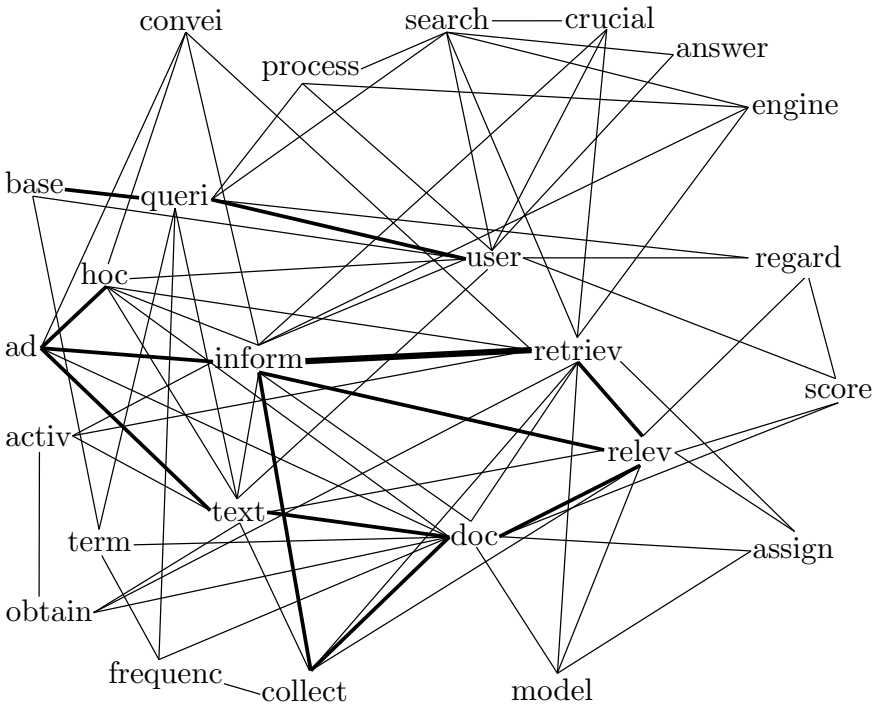
For KwE, we represented each document as a **weighted (un)directed graph-of-words** with a fixed sliding window of size 4 (cf. Figure 3.1 for illustration). We took into consideration edge weights since, in the context of *single-document* KwE, the task is done independently of the other documents and therefore we do not have the normalization issues we had for ad hoc IR (cf. Section 4.2.3.1), resulting actually in better performances.

Our idea was to consider the vertices of the main core as the set of keywords to extract from the document, the main core being obtained by running either the unweighted or the weighted version of the k-core algorithm on the graph-of-words. By analogy with the **keyworlds** from Matsuo et al. (2001b) that corresponded to the nodes contributing the most to the small-worldliness of the network, we called ours the **corewords** of the document. Indeed, the main core corresponds to the most cohesive connected component(s) of the graph and thus its vertices are intuitively good candidates for representing the entire graph-of-words. Additionally, assuming a set of golden keywords to compare with, we thought of considering more and more cores in the k-core decomposition and expected an increase in the recall of the extracted keywords without a too important decrease in precision or at least at a lower rate than for existing methods.

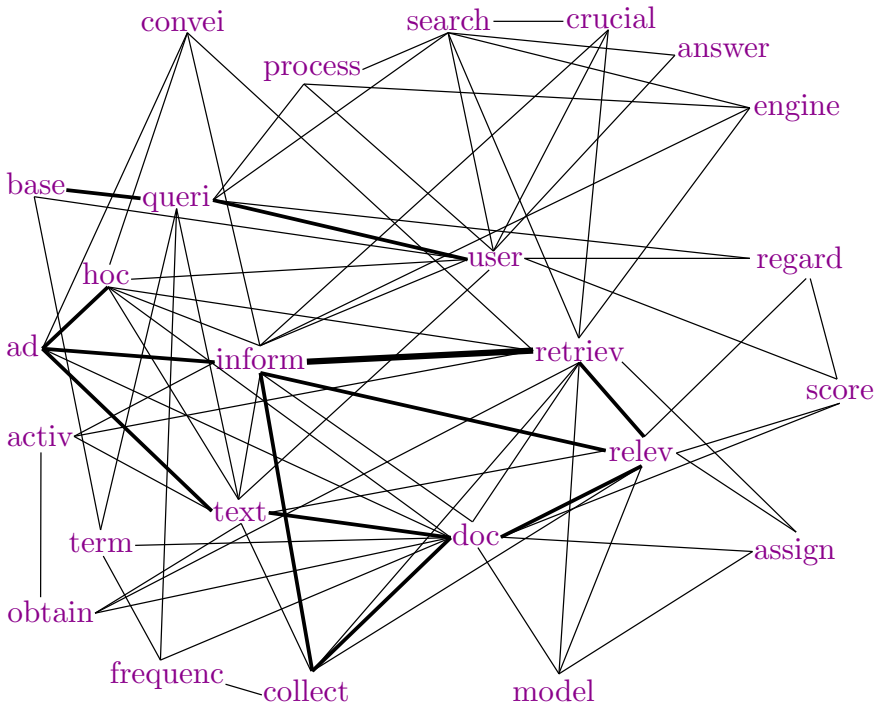
ILLUSTRATION Figure 5.2 illustrates the iterative procedure of decomposing a graph-of-words into nested k-cores of increasing cohesion. We kept the same node color code as in Figure 5.1 and grayed out the nodes not belonging to the current core as we go deeper in the decomposition. Figure 5.2d corresponds to the last core in the decomposition, i. e. the main core, and its terms highlighted in red form the **corewords** of the documents, i. e. the keywords selected by our method to represent the document. In particular, we see that the node 'queri' has a degree of 10 in the 4-core but because of its neighbors it does not belong to the 5-core while the node 'hoc' has a degree of only 6 but makes it to the main core. This figure has been manually created using the `Ipe` extensible drawing editor.

INTUITION Table 5.1 shows the ranked list of the unique terms from the document from Figure 3.1 according to their core numbers (first two columns) or eigenvector centrality (PageRank or HITS, last two columns) in the corresponding graph-of-words. Since our toy example is undirected, HITS authority and hub scores are the same. We can see the difference between weighting the edges or not for graph degeneracy, in particular in terms of the size of the main core. Figure 5.3 illustrates the distributions of eigenvector centralities and both display a S-shape resulting in three types of nodes: high, average and low centrality values.

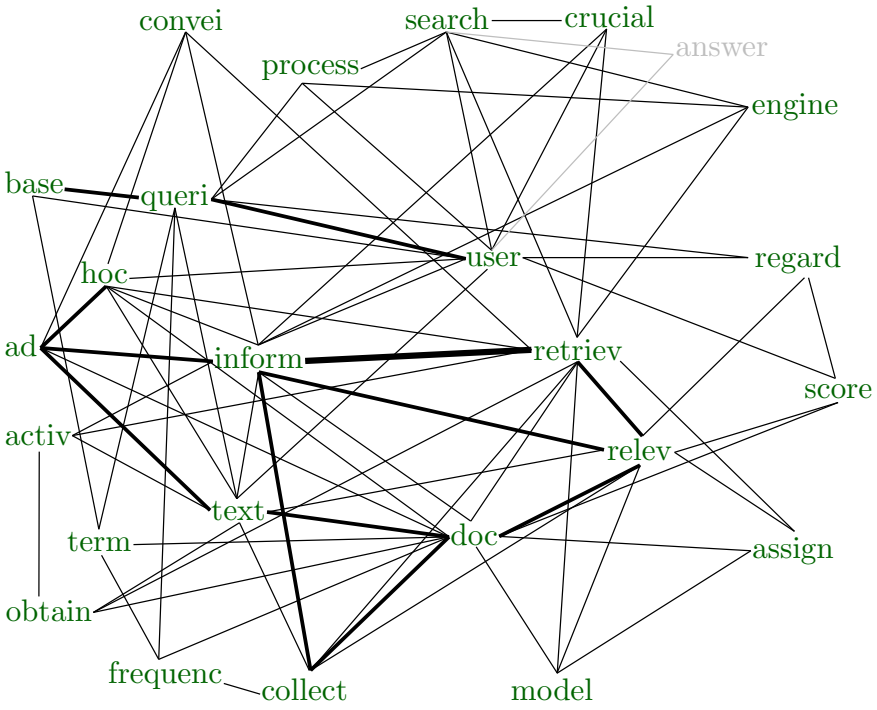
If we assume that the keyphrases a human annotator would choose for the toy document are the following: 'ad hoc information retrieval', 'relevance' and 'collection of textual documents' then we have a set of **golden keywords** to obtain, highlighted in bold in the table. Each method also needs to define a **cut-off** on the score above which it considers a term a keyword. For our proposed approach, we defined it as the main core, whatever the core number is or the number of nodes in it – note that they may both differ between unweighted and weighted versions of a network as observed in the table. For PageRank and HITS, Mihalcea and Tarau suggested extracting the top third of the vertices (top 33%), relative numbers helping accounting for documents of varying length. Here, for PageRank, following Figure 5.3, the cut-off should probably rather be between 'collect' and 'ad' or 'ad' and 'term'. Since this is just a toy example to get an intuition of what our method brings, we are not interested in finding if there is a better cut-off for the baselines, we will come back to this crucial point later on. We note that PageRank and HITS give relatively high scores to 'user', 'queri' and 'search', which are not golden keywords, because of their centrality but not degeneracy because of their neighbors. Again, the main core corresponds to a cohesive set of vertices (or potentially several sets) in which they all contribute equally to the subgraph they belong to – removing any node would collapse the entire subgraph through the cascading effect implied by the k-core condition. PageRank and HITS, on the other hand, provide scores for each vertex based on its centrality yet it does not require its neighbors to be central as well, providing enough neighbors to endorse it. But keywords happen to be keyphrases most of the time and thus capturing sets of co-occurring words is well-suited for the task.



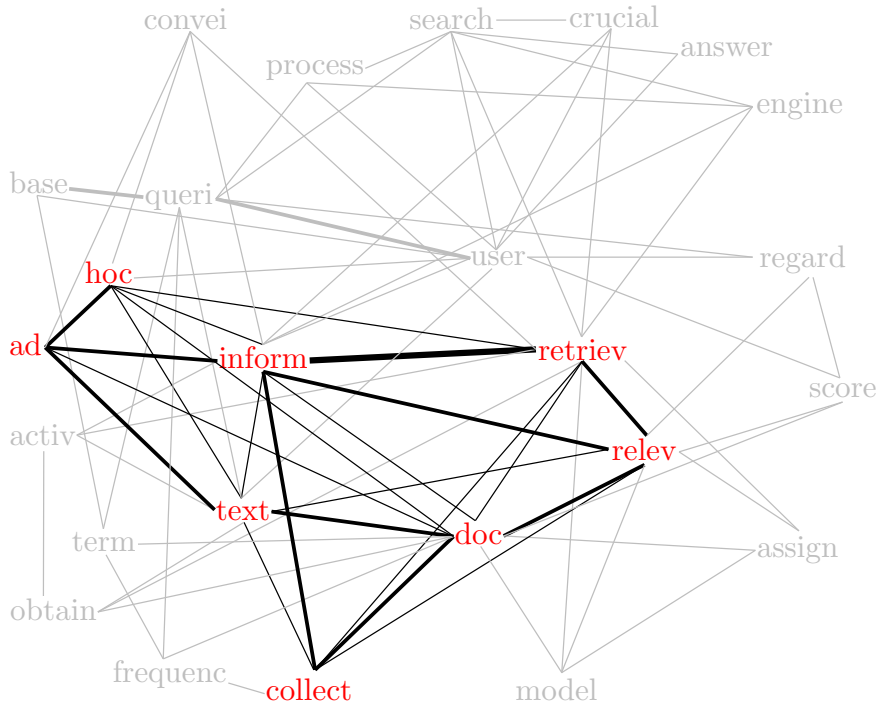
(a) 0-core, the initial graph from Figure 3.1



(b) 2-core, all the nodes survived



(c) 4-core, 'answer' is dropped since its degree is 2 in the 4-core

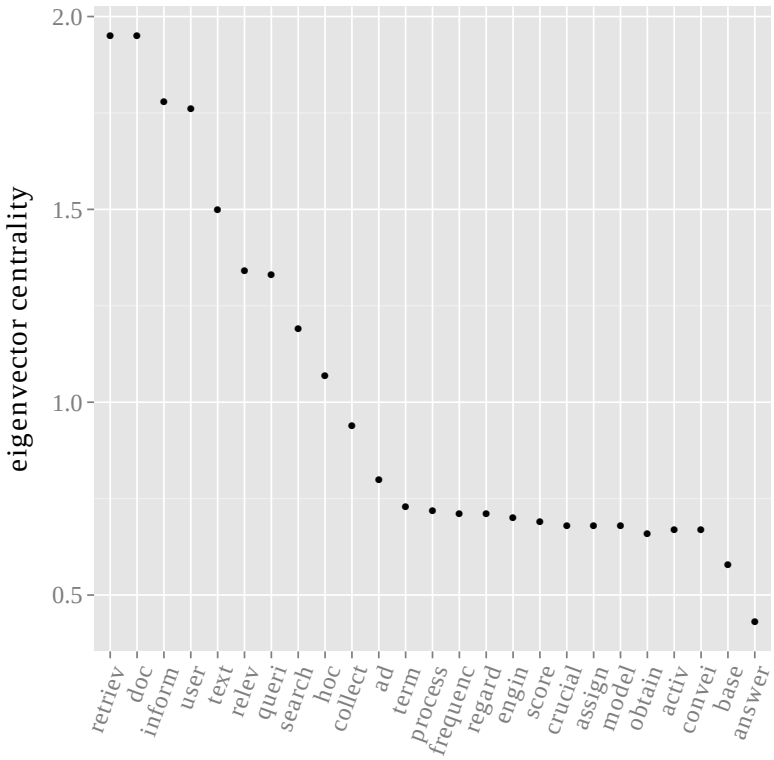


(d) 6-core, the main core – the remaining terms form the *corewords*

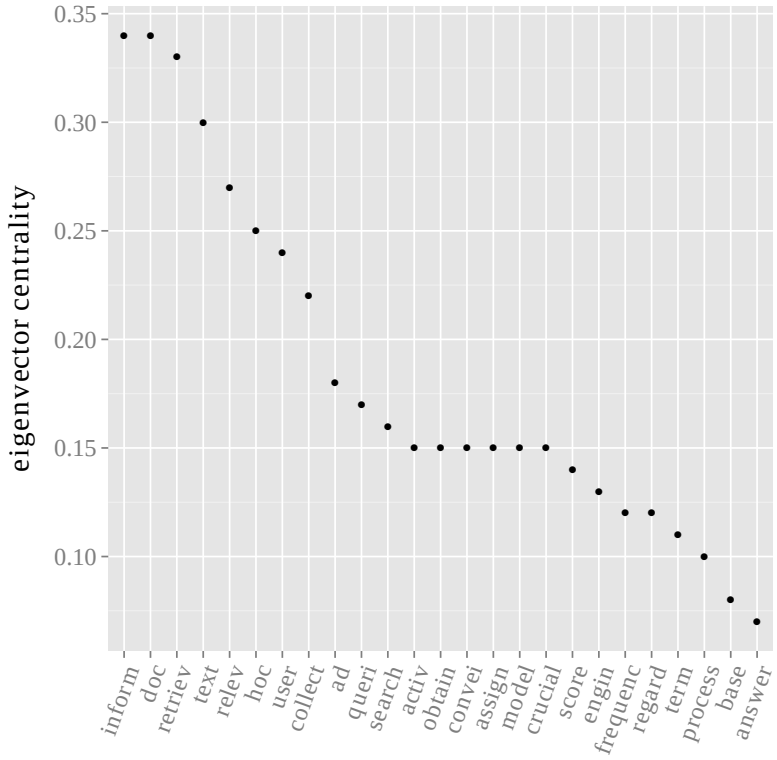
Figure 5.2 – Illustration of the k-core decomposition of the graph-of-words from Figure 3.1. The terms in red, belonging to the last and main core, form the corewords of the document.

k-core		k-core (weighted edges)		PageRank		HITS	
ad	4	**ad**	6	**retriev**	1.95	**inform**	0.34
hoc	4	**hoc**	6	**doc**	1.95	**doc**	0.34
inform	4	**inform**	6	**inform**	1.78	**retriev**	0.33
retriev	4	**retriev**	6	user	1.76	**text**	0.30
relev	4	**relev**	6	**text**	1.50	**relev**	0.27
collect	4	**collect**	6	**relev**	1.34	**hoc**	0.25
text	4	**text**	6	queri	1.33	user	0.24
doc	4	**doc**	6	search	1.19	**collect**	0.22
model	4	model	4	**hoc**	1.07	**ad**	0.18
assign	4	assign	4	**collect**	0.94	queri	0.17
convei	4	convei	4	**ad**	0.80	search	0.16
crucial	4	crucial	4	term	0.73	activ	0.15
engin	4	engin	4	process	0.72	obtain	0.15
score	4	score	4	frequenc	0.71	convei	0.15
regard	4	regard	4	regard	0.71	assign	0.15
activ	4	activ	4	engin	0.70	model	0.15
process	4	process	4	score	0.69	crucial	0.15
queri	4	queri	4	crucial	0.68	score	0.14
obtain	4	obtain	4	assign	0.68	engin	0.13
user	4	user	4	model	0.68	frequenc	0.12
search	4	search	4	obtain	0.66	regard	0.12
term	3	term	4	activ	0.67	term	0.11
frequenc	3	frequenc	4	convei	0.67	process	0.10
base	3	base	4	base	0.58	base	0.08
answer	2	answer	2	answer	0.43	answer	0.07

Table 5.1 – Ranked list of the unique terms from the document from Figure 3.1 according to their core numbers or eigenvector centrality (PageRank or HITS) in the corresponding graph-of-words. Bold font indicates the golden keywords and the dashed lines the cutoff for each method.



(a) PageRank



(b) HITS

Figure 5.3 – Ranked list of the unique terms from the document from Figure 3.1 according to their eigenvector centrality (PageRank or HITS) in the corresponding graph-of-words.

5.2.2.2 *Keywords are keyphrases*

For the 500 abstracts from the *Inspec* database that we used in our experiments, only 662 out of the 4,913 keyphrases manually assigned by human annotators are keywords (13%). The rest of them range from bigrams (2,587 – 52%) to 7-grams (5). Similar statistics were observed on other datasets. Therefore, we make the general claim that human annotators tend to select *keyphrases*. Thus, to improve the performances of an automated system, one needs to capture the interactions between keywords in the first place. This is the main reason why we explored the graph-of-words representation to challenge the traditional unigram bag-of-words and why we considered graph degeneracy to extract cohesive sets of keywords.

Even if nodes correspond to unigrams because of the way the graph-of-words is constructed, the edges do represent co-occurrences within a sliding window. And for small-enough sizes (which is typically the case in practice), we can consider that two linked vertices represent a long-distance bigram (Bassiou and Kotropoulos, 2010), if not a bigram (cf. Section 3.2.3). Hence, by considering cohesive subgraphs, we make sure to extract unigrams that co-occur together and thus are bigrams, if not higher order n-grams. On the contrary, methods based solely on centrality measures may extract unigrams that are central because they co-occur with a lot of other words but these words may not be extracted as well because of a lower weight. Extracting salient bigrams would require to include bigrams as vertices but the number of nodes increases linearly with the order of the n-gram.

5.2.2.3 *Corewords are adaptive*

Most current techniques in KwE assign a score to each term of a document and then select the top ones as keywords. For a given collection of homogeneous documents in length or because of specific constraints, an **absolute** number may make sense. For example, Turney (1999) limited to the top 5 keyphrases while Witten et al. (1999) to the top 15, which is inline with what is imposed by some publishers for academic papers for instance. Mihalcea and Tarau (2004) argued that a **relative** number should be used for documents of varying length or when no prior is known, i. e. top X% instead of top X. We go further and claim that the numbers of retrieved keywords should be decided at the document level and not at the collection level. For instance, for two documents, even of equal length, one of them might require more keywords to express the gist of its content (because it deals with more topics for example). Therefore, adapting the number of extracted keywords to the document should yield results closer to the human ones.

Grineva et al. (2009) suggested to use the **elbow method**, an unsupervised empirical method originally developed for selecting the number of clusters in k-means (Thorndike, 1953). Indeed, as mentioned in Section 5.1.2, most centrality measures will follow a power law distribution because of the scale-free property observed in graph-of-words (cf. Section 3.2.4). Consequently,

the authors proposed to select the keywords with a score in the "upper arm". However, it was noted in Lahiri, S. R. Choudhury, et al. (2014) that the elbow is not always clear, especially for short text (in their case for closeness), and more complex to automate. On our toy example, we see in Figure 5.3 that for both PageRank and HITS, it is not straightforward to choose the cut-off because of the plateau in the middle and therefore, in our experiments, we opted for relative and absolute numbers.

In our case, the size of each k-core, i. e. the number of vertices in the subgraph, depends on the structure of the graph. In the unweighted case, it is lower-bounded by $k+1$ since each vertex has at least k neighbors but can potentially be up to n in the case of a complete graph. Hence, we think that degeneracy can capture this variability in the number of extracted keywords, in particular for a fixed document length (the other methods will still extract more and more keywords as the document length increases when using relative numbers). In Section 5.2.3.4, we will show distributions of extracted keywords per document length for all models and from human annotators to support our claim.

5.2.3 *Experiments*

In this section, we present the experiments we carried out to validate our proposed keyword extraction technique. We first describe the datasets, the evaluation metrics and the considered models. We then report the results we obtained and discuss their interpretations.

5.2.3.1 *Datasets, evaluation metrics and models*

DATASETS We used two standard collections of documents publicly available [1]: (1) `Hulth2003` and (2) `Krapivin2009`. `Hulth2003` consists of 500 abstracts from the *Inspec* database introduced by Hulth (2003) and used in many works since then (Mihalcea and Tarau, 2004; Z. Liu et al., 2009; Tsatsaronis et al., 2010; Boudin, 2013; Lahiri, S. R. Choudhury, et al., 2014). In her experiments, Hulth is using a total of 2,000 abstracts, divided into 1,000 for training, 500 for validation and 500 for test. Since our approach is unsupervised and because this is also the choice made by all the others, we only used the 500 test documents in order to get comparable results. Additionally, we used the "uncontrolled" golden keywords since we do not want to restrict the keywords to a given thesaurus. `Krapivin2009` consists of 2,304 ACM full papers in CS (references and captions excluded) introduced by Krapivin et al. (2009) – the golden keywords are the ones chosen by the authors of each ACM paper. Since we are interested in *single-document* KwE, the scalability of the methods are measured with regards to the document length, not the collection size (that only needs to be large enough to measure the statistical significance of the improvements). This is the main reason why we considered `Krapivin2009` as second dataset: to see how these methods scale for longer documents and larger graphs.

1. https://github.com/snkim/AutomaticKeyphraseExtraction

EVALUATION METRICS For each document, we have a set of golden keywords manually assigned by human annotators and a set of extracted keywords produced by each method (one of the baselines or of our approaches). We can thus naturally compute **precision**, **recall** and **F1-score** per document and per method. Following the suggestion of Hulth (2003), we consider the proportion of correctly extracted keywords (precision) equally as important as the amount of terms assigned by a professional indexer (recall) hence the choice of a balanced F-measure (i.e. F1-score, cf. Section 2.3.2.3). We then **macro-average** these metrics to get results at the collection level (cf. Section 2.3.3.2) – we prefer to macro-average (arithmetic mean over all documents of the metrics per document) rather than micro-average (metric computed over all documents) in order not to bias towards longer documents (with more keywords). Note that it is not always clear in the related works which average the authors took. We believe the historical supervised approaches (Turney, 1999; Witten et al., 1999; Hulth, 2003) considered micro-averaged results since the classification examples were the actual words, regardless of which documents they belong to, and therefore they naturally pooled the per-example decisions (cf. Section 2.2.3). However, in the unsupervised graph-based approaches, we consider documents separately, which is better represented by macro-averaged results. The statistical significance of improvement was assessed using the paired Student's t-test considering two-sided p-values less than 0.05 to reject the null hypothesis (cf. Section 2.3.4).

KEYWORD VS. KEYPHRASE In practice, humans tend to select key-phrases to represent documents (cf. Section 5.2.2.2) but the graph-based methods select nodes, i.e. unigrams. Therefore, we converted the golden keyphrases into unigrams applying the same preprocessing steps (e.g., stop word removal or POS-tag filtering) beforehand. That way, it is straightforward to compute precision and recall between this set of golden unigrams and the set of extracted unigrams. In their survey, Hasan and V. Ng (2010) suggested to only consider the actual keywords (i.e. discard the other keyphrases) but this does not hold for the datasets we used (cf. Section 5.2.2.2). It does not penalize that much models that would be designed to extract higher order n-grams since they would still get higher precision and recall. Otherwise, it becomes hard to define hits and misses when a model might extract two unigrams and the overall bigram is expected. Indeed, there is no standard definition on how to penalize a method that, given a golden bigram to extract, would return part of it (unigram) or more than it (trigram). Mihalcea and Tarau (2004) suggested "reconciling" the n-grams as a post-processing step by looking in the original text for adjacent unigrams but then questions arise such as whether we should keep the original unigrams in the final set, impacting the precision and recall. Similarly, Z. Liu et al. (2009) proposed to merge keywords that occur together in the original text as a noun phrase, i.e. matching the following regular expression: $(JJ)^*$ $(NN|NNS|NNP)^+$. It is not clear how the other related works did it and if one method is preferable to the others. Therefore, we consider this reconciliation step as part of the application that would use any keyword extraction method but not part of the technique itself, hence an evaluation based on unigrams.

MODELS For the graph-of-words representation, we experimented with undirected edges, forward edges (natural flow of the text – an edge $term_1 \rightarrow term_2$ meaning that $term_1$ precedes $term_2$ in a sliding window) and backward edges (the opposite). In terms of keyword-extracting methods, we considered (1) PageRank, (2) HITS, (3) k-core on an unweighted network and (4) k-core on a weighted one (the edge weight being the number of co-occurrences). Note that for HITS, we only report the results for the authority scores since the hub scores are the same in the undirected case and symmetric in the directed case (the hub score for forward edges corresponds to the authority score for backward edges and vice versa).

NUMBER OF KEYWORDS We extracted the top third keywords (top 33%) on `Hulth2003` and the top 15 keywords on `Krapivin2009` for the baselines and the main core for our approaches (the highest k value differs from document to document). The choice between relative (top X%) and absolute numbers (top X) comes from the fact that for relatively short documents such as abstracts, the longer the document, the more keyword human annotators tend to select while past a certain length (10-page long for ACM full papers), the numbers vary far less. Hence, in all fairness to the baselines, we selected the top 33% on the abstracts like in the original papers and the top 15 for the full papers (15 being the average number of unigrams selected as keywords by the papers' authors on the dataset).

5.2.3.2 *Macro-averaged results*

Table 5.2 show the macro-average precision, recall and F1-score for all models (rows) for the different variants of graph-of-words considered (multirows) on the `Hult2003` dataset; same for Table 5.3 on the `Krapivin2009` dataset. Overall, PageRank and HITS have similar results, with a precision higher than the recall as reported in previous works. It is the opposite for k-core with unweighted edges, which tends to extract a main core with a lot of vertices since the k-core condition can be interpreted as a set of keywords that co-occur with at least k other keywords. For the weighted case, it corresponds to a set of keywords that co-occur at least k times in total with other keywords leading to cores with fewer vertices but with stronger links and the extraction of important bigrams, hence the increase in precision (at the cost of a decrease in recall) and an overall better F1-score.

Edge direction has an impact but not necessarily a significant one and is different across methods. This disparity in the results and the lack of a dominant choice for edge direction is consistent with the relevant literature. Mihalcea and Tarau (2004) and Blanco and Lioma (2012) used undirected edges, Litvak and Last (2008) backward edges and Rousseau and Vazirgiannis (2013b) forward edges. Hence, we recommend the use of undirected edges for ease of implementation but other techniques that would try to extract paths from the graph-of-words for instance might need the edge direction to follow the natural flow of the text such as for summarization (Filippova, 2010).

Graph	Method	Precision	Recall	F1-score
undirected edges	PageRank	0.589	0.422	0.473
	HITS	0.579	0.418	0.466
	k-core	0.465	**0.625***	0.491*
	k-core (weighted)	**0.612***	0.503*	**0.519***
forward edges	PageRank	0.558	0.420	0.457
	HITS	0.548	0.404	0.450
	k-core	0.425	**0.729***	**0.517***
	k-core (weighted)	**0.570***	0.469*	0.506*
backward edges	PageRank	0.593	0.427	0.476
	HITS	0.564	0.407	0.454
	k-core	0.409	**0.706***	0.452
	k-core (weighted)	**0.602***	0.499*	**0.500***

Table 5.2 – Effectiveness results (macro-average precision, recall and F1-score) of coreword extraction over eigenvector centrality-based keyword extraction on the `Hulth2003` dataset. Bold font marks the best performance in a column of a block. * indicates statistical significance at $p < 0.05$ using the Student's t-test with regard to the PageRank baseline of the same column of the same block.

5.2.3.3 *Precision/recall curves*

Additionally, instead of just considering the main core or the top X% vertices, we computed precision and recall at each core and at each percent of the total number of terms to get **precision/recall curves** (cf. Section 2.3.3.1). We used relative numbers because the documents are of varying length so the top 10 keywords for a document of size 10 and 100 do not mean the same while the top 10% might.

We show on Figure 5.4 the resulting curves on the `Hulth2003` dataset, one for each model (100 points per curve, no linear interpolation following Davis and Goadrich (2006)). The final recall for all models is not 100% because human annotators sometimes used keywords that do not appear in the original documents. Since we are dealing with the task of *extractive* KwE (as opposed to abstractive), the methods are only supposed to extract keywords present in the

Graph	Method	Precision	Recall	F1-score
undirected edges	PageRank	0.502	0.488	0.496
	HITS	0.495	0.479	0.480
	k-core	0.405	**0.784***	0.466
	k-core (weighted)	**0.535***	0.502*	**0.508***
forward edges	PageRank	0.478	0.449	0.457
	HITS	0.470	0.442	0.450
	k-core	0.398	**0.791***	0.460
	k-core (weighted)	**0.522***	0.457*	**0.470***
backward edges	PageRank	0.514	0.500	**0.505**
	HITS	0.491	0.470	0.474
	k-core	0.392	**0.776***	0.469
	k-core (weighted)	**0.521***	0.502*	0.504

Table 5.3 – Effectiveness results (macro-average precision, recall and F1-score) of coreword extraction over eigenvector centrality-based keyword extraction on the `Krapivin2009` dataset. Bold font marks the best performance in a column of a block. * indicates statistical significance at $p < 0.05$ using the Student's t-test with regard to the PageRank baseline of the same column of the same block.

documents. We observe that the curve for the k-core with weighted edges (green, solid circle) is systematically above the others, thus showing improvements in Area Under the Curve (AUC) and not just in point estimates such as the F1-score. The curve for k-core (orange, diamond) is overall below the other curves as the algorithm tends to only find a few cores with a lot of vertices, lowering the precision but insuring some minimum recall (its lowest value of recall is greater than for the other curves).

5.2.3.4 *Distribution of the number of keywords*

Human annotators do not assign the same number of keywords to all documents. There is a variability that is partially due to varying document length (the number increases with the length) as observed on the `Hulth2003` but not only. Indeed, when computing a distribution per document length, we can still

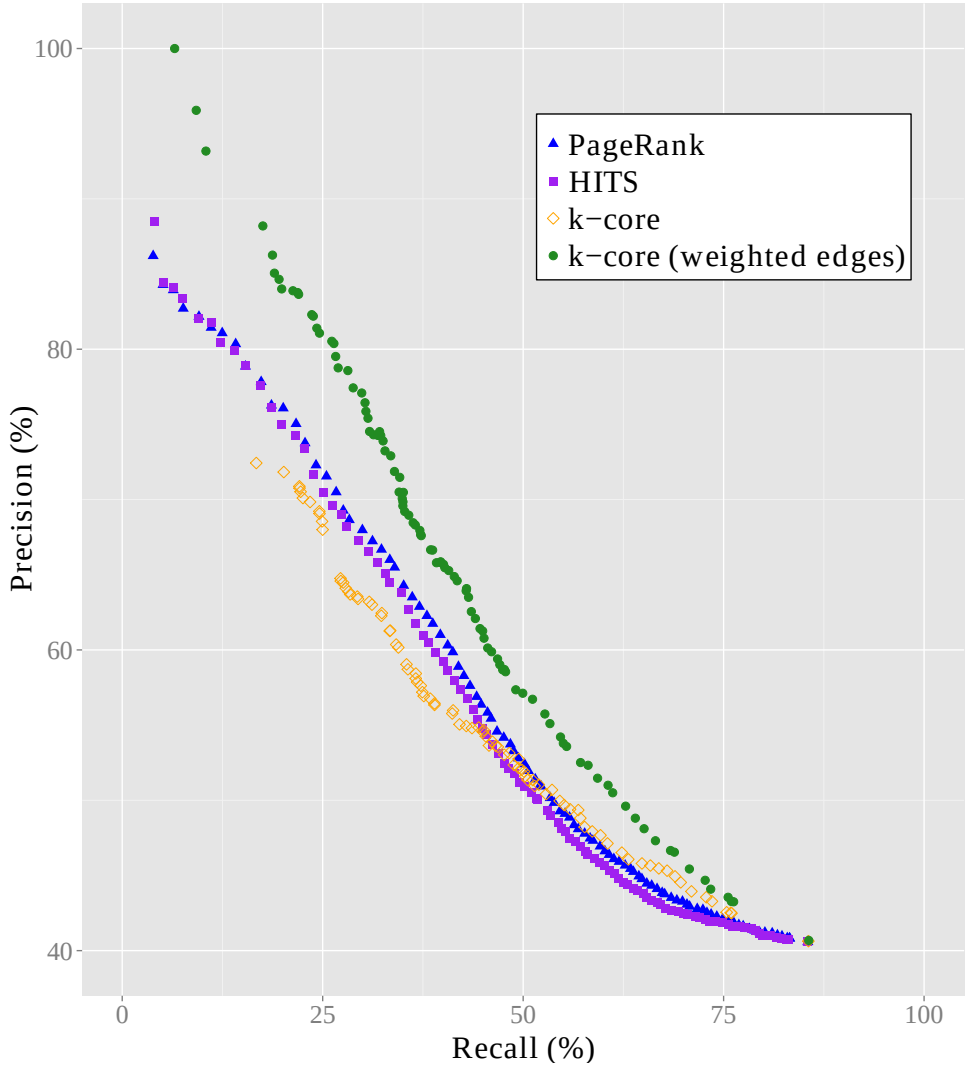


Figure 5.4 – Effectiveness results (macro-average P/R curves) of coreword extraction over eigenvector centrality-based keyword extraction on the `Hulth2003` dataset.

observe some dispersion. With PageRank, by extracting a relative number of unigrams (top 33%), one accounts for varying length but does not fully capture the variability introduced by human annotators while *k*-core does better. Similarly, for the `Krapivin2009` dataset, where documents are of the same length (10-page), some authors may have chosen more keywords for their paper than others because for instance there are alternative denominations for the concept(s) developed in their work and as a result more keywords.

Figure 5.5 shows groups of 3 box plots computed on the `Hulth2003` dataset. In each group, the left one corresponds to PageRank (in blue, similar results for HITS), the middle one to human annotators (white) and the right one to *k*-core with weighted edges (green). We do not show any box plot for *k*-core with unweighted edges since the method tends to overestimate the number of keywords (higher recall, lower precision). For space constraints and also for

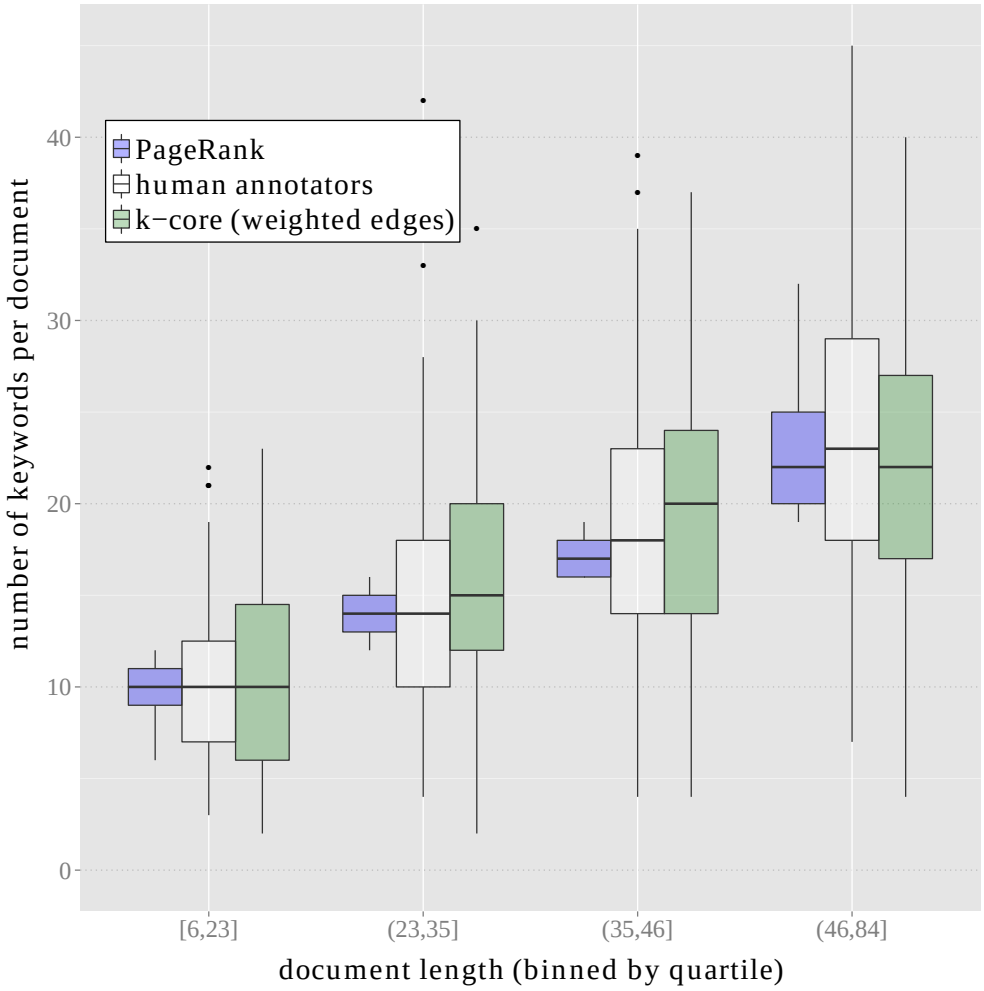


Figure 5.5 – Distribution of the number of keywords assigned by human annotators (middle, white) versus extracted (a) via PageRank (left, blue) and (b) via *k*-core with weighted edges (right, green) for different document lengths (binned by quartile) on the `Hulth2003` dataset.

sparsity reasons, we binned the document lengths by quartile (i. e. the bins are not of equal range but contains the same number of documents – 25% each).

As expected, the number of keywords for a given bin varies little for PageRank – the increase in median value across the bins being due to the baseline taking the top third unigrams, relative number that increases with the document length. For *k*-core with weighted edges, we observe that the variability is taken much more into account: the first, second and third quartiles' values for the number of keywords are much closer to the golden ones. For PageRank and HITS, it would be much harder to learn the number of keywords to extract for each document while it is inherent to graph degeneracy. Alone, these results would not mean much but because higher effectiveness has already been established through consistent and significantly higher macro-average F1-scores, they support our claim that *k*-core is better suited for the task of keyword extraction because of its adaptability to the graph structure and therefore to the document structure.

5.2.4 *Highlights, current limitations and future work*

Similarly to previous approaches, we capitalized on the graph-of-words representation to extract central terms from documents. However, by retaining only the main core of the graph, we were able to capture more cohesive subgraphs of vertices that are not only central but also densely connected. Hence, the extracted corewords are more likely to form keyphrases and their number adapts to the graph structure, as human annotators tend to do when assigning keywords to the corresponding document.

As a final example, here are the stemmed unigrams belonging to the main core of the graph-of-words corresponding to the paper we published on this work (references and captions excluded): {*keyword, extract, graph, represent, text, weight, graph-of-word, k-core, degeneraci, edg, vertic, number, document*}. Using PageRank, "work" appears in the top 5, "term" and "pagerank" in the top 10, and "case" and "order" in the top 15. Existing methods tend to extract central keywords that are not necessarily part of a cohesive subgraph as opposed to our proposed approach, which provides closer results to what humans do on several aspects.

LIMITATIONS Our main regret in this work was that most of the main cores were connected graphs, even on 10-page long ACM papers, i. e. with a single component, while we were expecting to extract sets of keywords, one per (sub)topic ideally. Moreover, as tested in practice when developing a Web browser plugin based on coreword extraction, the end users want unprocessed keyphrases, not a set of stemmed keywords and therefore the reconciliation post-processing step mentioned in Section 5.2.3.1 needs to be solved as well. Content extraction from a Web page turned out to be also a challenge as the core text might be surrounded by other "noisy" textual elements as opposed to abstracts and full papers, but this is another research issue in itself.

FUTURE WORK Possible extension of this work would be the exploration of the clusters of keywords in the top cores (even in a single component) to elect representatives per cluster for topic modeling following the approach from Z. Liu et al. (2009). Also, we could explore alternative definitions of the degree to keep the good recall of the unweighted graph and the precision of the weighted one, capitalizing on the generalized cores of Batagelj and Zaveršnik (2002) that work for additional definitions of the degree.

6

LEARNING PATTERNS FROM GRAPH-OF-WORDS

In this chapter, we learn classification patterns in text from the graph-of-words representation of documents. We first present the state-of-the-art approaches to obtain a classifier from supervised learning on labeled data. We then challenge the traditional feature space used for Text Categorization (TC) by considering subgraphs of graph-of-words as features and the task as a graph classification problem. Additionally, we show how we can capitalize on the coreword extraction technique introduced in Chapter 5 to reduce the dimension of the feature spaces for little to no cost in prediction performances. Finally, we tackle the sparsity issue that arises when classifying short documents or when little training data is available by proposing a convolutional sentence kernel on syntactic graph-of-words based on distances in word embeddings.

6.1 CLASSIFICATION FROM SUPERVISED LEARNING

In this section, we assume a classification task for which we have already-classified examples and the goal is to obtain a model capable of predicting class labels for unseen examples. The task is usually split into two components: (1) **feature extraction**, i. e. choose a set of common features to describe each example; and (2) **model learning**, i. e. produce an automated system that can correctly classify the data we have and make good predictions about the data we do not have. Since we have ground-truth labels for a set of examples, we consider the task as a **supervised** problem, i. e. *learning* a model on a labeled training set, *tuning* its potential parameters on a validation set and *evaluating* it on a test set (in a typical 60-20-20 split or 80-20 with cross-validation on the 80% training+validation set), as opposed to an unsupervised approach like the one we adopted for single-document keyword extraction in Chapter 5.

FEATURE SPACE Throughout Section 6.1, we assume we have already chosen the set of features, i. e. the examples are represented as **feature vectors** $(f_k)_{k\in[\![1,p]\!]}$ in a p-dimensional feature space. Typically, for TC, examples are documents (denoted by $\boldsymbol{d}$ and represented as unigram bag-of-words), $p = |\mathcal{V}|$ and each f_k corresponds to a unique unigram of the vocabulary. In this context, the collection $\mathcal{D}$ can thus be seen as a (sparse) **document-term matrix** of dimension $N \times p$; each feature vector constituting a row of this matrix. The sparsity comes from the fact that a document only contains a few terms of the vocabulary and thus a lot of 0s in its associated feature vector. In the rest of the section, we will be taking TC as our use case for classification since this was the application we considered in our research but most of the content applies to the general case, e. g., examples could be graphs and features subgraphs if we were to classify networks.

6.1.1 *Probabilistic classifiers*

We present in this section classifiers that rely on a probabilistic formulation of the problem and on maximizing the likelihood of the parameters given the observed data.

6.1.1.1 *Bayesian formulation*

Given a document $\boldsymbol{d}$ and a set of categories $\boldsymbol{\mathcal{C}}$, for *single-label* classification, we want to predict its most likely class label $c^* \in \boldsymbol{\mathcal{C}}$, i. e. the category with *maximum a posteriori* probability $\mathbb{P}(c|\boldsymbol{d})$ – we will use the caret symbol to signify that we only have access to an estimate $\hat{\mathbb{P}}$ of the true probability $\mathbb{P}$, estimate computed on a training set typically. In other words, we want to find the category c^* such that:

$$\begin{aligned} c^* &= \arg\max_{c \in \boldsymbol{\mathcal{C}}} \left(\hat{\mathbb{P}}(c|\boldsymbol{d})\right) \\ &= \arg\max_{c \in \boldsymbol{\mathcal{C}}} \left(\frac{\hat{\mathbb{P}}(c) \times \hat{\mathbb{P}}(\boldsymbol{d}|c)}{\hat{\mathbb{P}}(\boldsymbol{d})}\right) && \text{(Bayes' rule)} \\ &= \arg\max_{c \in \boldsymbol{\mathcal{C}}} \left(\hat{\mathbb{P}}(c) \times \hat{\mathbb{P}}(\boldsymbol{d}|c)\right) && (\hat{\mathbb{P}}(\boldsymbol{d}) \text{ same for all } c) \end{aligned} \tag{6.1}$$

Bayes' rule encompasses the fundamental idea behind supervised Machine Learning (ML). In the end (e. g., on a test set), we want a model that makes predictions given the data (from $\mathbb{P}(c|\boldsymbol{d})$, the *posterior probability*). And to learn this model, we consider the data given the predictions that we have (from $\mathbb{P}(\boldsymbol{d}|c)$, the *likelihood*, estimated typically on a training set); hence the swap between $c|\boldsymbol{d}$ and $\boldsymbol{d}|c$. Equation 6.1 can be interpreted as a **generative process** since we model explicitly the joint probability $\mathbb{P}(\boldsymbol{d}, c)$: to generate a document $\boldsymbol{d}$, we first choose a class label c with probability $\mathbb{P}(c)$ (the *prior*) and then, given this category, we choose a document according to some distribution $\mathbb{P}(\boldsymbol{d}|c)$. Note that in the second equation above, it is fine to drop the denominator $\mathbb{P}(\boldsymbol{d})$ (the *evidence*) because it is the same for all classes and we are only interested in the classification decisions (arg max), not the probability estimates themselves.

Generally speaking, in ML, assuming a p-dimensional binary feature space of (very) high dimension, we are trying to estimate $\mathbb{P}(f_1, \ldots, f_p|c)$, which would require estimating $\mathcal{O}(2^p \times |\boldsymbol{\mathcal{C}}|)$ parameters in practice. This could only be estimated if a very, very large number of training examples were available. One solution is to tackle the problem "naively" by making additional assumptions, hence the common Naive Bayes (NB) denomination described next.

6.1.1.2 *Naive Bayes*

A Naive Bayes model assumes **conditional independence** between the features: each of the attributes it uses are conditionally independent of one another *given* the class label, which means:

$$\mathbb{P}(f_1, \ldots, f_p|c) = \prod_{k=1}^{p} \mathbb{P}(f_k|c) \tag{6.2}$$

The assumption of conditionally independent features is often violated in practice but the model still performs well enough in general. This has been explained by Friedman (1997) and Domingos and Pazzani (1997) by the fact that binary classification estimation is only a function of the sign of the function estimation and therefore the function approximation can still be poor while classification accuracy remains high. Under this assumption, we now only have $\mathcal{O}(p \times |\boldsymbol{\mathcal{C}}|)$ parameters to estimate, which is a lot more practical.

6.1.1.3 *Bernoulli vs. Multinomial Naive Bayes*

There are two standard Naive Bayes processes in TC to model the generation $\mathbb{P}(\boldsymbol{d}|c)$ of a document given a class. Intuitively, because of the choice of document representation (bag-of-words), we are already used to consider terms independently of one another in a document.

BERNOULLI The first process considers $|\boldsymbol{\mathcal{V}}|$ binary random variables E_t indicating presence ($E_t = 1$) or absence ($E_t = 0$) of each vocabulary term t in the document. Assuming that each of these random variables is independent of one another given the class, the Naive Bayes document generation model corresponds to a process following a **multivariate Bernoulli** distribution. Indeed, each variable E_t follows a Bernoulli distribution such that:

$$\mathbb{P}(E_t = e_t|c) = \mathbb{P}(T = t|c)^{e_t} \left(1 - \mathbb{P}(T = t|c)\right)^{1-e_t}, \quad e_t \in \{0, 1\} \tag{6.3}$$

where T is a categorical random variable that can take any of the vocabulary term as value. This results in:

$$\mathbb{P}(\boldsymbol{d}|c) = \prod_{t \in \boldsymbol{\mathcal{V}}} \mathbb{P}(t|c)^{e_t} \left(1 - \mathbb{P}(t|c)\right)^{1-e_t} \qquad \text{(Bernoulli)} \tag{6.4}$$

MULTINOMIAL The second process considers $|\boldsymbol{d}|$ successive draws with replacement of any of the vocabulary term t with probability $\mathbb{P}(T = t|c)$ or $\mathbb{P}(t|c)$ for short. Basically, for every successive token position in the document, we pick a term from the vocabulary. We have to make a first assumption known as **positional independence**. That is, the conditional probabilities of a term given a class are the same, regardless of its position in the document. Indeed, we considered that between each trial the probabilities $\mathbb{P}(t|c)$ do not change based on the position (hence a single categorical random variable T). This is fine because the position of a term in a document does not carry information

about the class by itself since we considered terms independently of each other anyway. This also has practical reasons: in terms of probability estimation, the contrary would pose a problem of **data sparsity** since we would need training data in which every word appears in every position of every document.

Moreover, the generative process needs an additional step that generates a document length according to some distribution $\mathbb{P}(|\boldsymbol{d}||c)$. When this is taken into account, researchers have been assuming that the document length does not depend on the class and thus considered $\mathbb{P}(|\boldsymbol{d}|)$ instead (McCallum and Nigam, 1998; Schneider, 2003; Metsis et al., 2006). This is an additional over-simplistic assumption, which is more questionable in spam filtering. For example, the probability of receiving a very long spam message appears to be smaller than that of receiving an equally long ham message (Metsis et al., 2006).

The $|\boldsymbol{\mathcal{V}}|$ categorical random variables X_t indicate the number of times each possible outcome t is observed over the $|\boldsymbol{d}|$ trials (i. e. the number of times the term t occurs in the document – its term frequency $tf_{\boldsymbol{d}}(t)$). Assuming that each draw is independent of one another given the class, the Naive Bayes document generation model corresponds to a process following a **multinomial** distribution. In NLP, this corresponds to having a *unigram language model* per class for document generation. This results in:

$$\mathbb{P}(\boldsymbol{d}|c) = \mathbb{P}(|\boldsymbol{d}|)\frac{|\boldsymbol{d}|!}{\prod_{t\in\boldsymbol{\mathcal{V}}} tf_{\boldsymbol{d}}(t)!}\prod_{t\in\boldsymbol{\mathcal{V}}}\mathbb{P}(t|c)^{tf_{\boldsymbol{d}}(t)} \quad \text{(Multinomial)} \tag{6.5}$$

INTERPRETATIONS The two models have a very different interpretation in terms of how to generate a document given a class and the random variables considered. In particular, the multinomial model takes into account multiple occurrences of the same term but not the Bernoulli one, which only states that a term appears (or not) in a document regardless of its position(s) and number of occurrences.

On one hand, as noted by McCallum and Nigam (1998), consider, for example, the occurrence of numbers in the Reuters newswire articles (a standard TC dataset) and let's assume that the tokenization process maps all strings of digits to a single token "_NUMBER_" (a common pre-processing in NLP). Since every news article is dated, and thus has a number, the special token in the multivariate Bernoulli event model is uninformative. However, news articles about earnings tend to have a lot of numbers compared to general news articles. Thus, capturing frequency information of this token can help classification.

On the other hand, the Bernoulli model explicitly includes the non-occurrence probability of vocabulary terms that do not appear in the document while the multinomial one does not. In general, the Bernoulli model works better for short documents and low-dimension vocabulary while the multinomial model scales better to longer documents and larger feature space.

MAXIMUM A POSTERIORI Going back to the initial problem of finding the category with maximum a posteriori probability $\mathbb{P}(c|\boldsymbol{d})$, the expression for the multinomial model can be greatly simplified by removing all the terms that do not depend on the class (hence why we do not consider that the document length depends on the category):

$$c^* = \arg\max_{c \in \boldsymbol{\mathcal{C}}} \left(\hat{\mathbb{P}}(c) \times \prod_{t \in \boldsymbol{\mathcal{V}}} \hat{\mathbb{P}}(t|c)^{e_t} \left(1 - \hat{\mathbb{P}}(t|c)\right)^{1-e_t} \right) \qquad \text{(Bernoulli)} \qquad (6.6)$$

$$c^* = \arg\max_{c \in \boldsymbol{\mathcal{C}}} \left(\hat{\mathbb{P}}(c) \times \prod_{t \in \boldsymbol{\mathcal{V}}} \hat{\mathbb{P}}(t|c)^{tf_{\boldsymbol{d}}(t)} \right) \qquad \text{(Multinomial)} \qquad (6.7)$$

OVERSIMPLIFIED MODEL Note that in quite a lot of resources (papers and online material), the expression we see for NB for text categorization is different from either model's. It is usually of the form:

$$c^* = \arg\max_{c \in \boldsymbol{\mathcal{C}}} \left(\hat{\mathbb{P}}(c) \times \prod_{t \in \boldsymbol{d}} \hat{\mathbb{P}}(t|c) \right) \qquad (6.8)$$

Basically, it omits the vocabulary terms that are absent from the document as opposed to the Bernoulli model and the term frequencies as opposed to the multinomial model. It still works in practice because of the high number of terms in the vocabulary that result in practice in very low $\hat{\mathbb{P}}(T = t|c)$ and thus the term $\prod_{t \notin \boldsymbol{d}} \left(1 - \hat{\mathbb{P}}(T = t|c)\right)$ close to 1. Additionally, it limits the number of computations to the document length like for the multinomial model instead of the vocabulary size.

MULTIVARIATE POISSON S.-B. Kim et al. (2003) proposed a Naive Bayes multivariate Poisson model to capture the term frequencies that the Bernoulli model was missing, modeling the number of occurrences of each vocabulary term in a document as following a Poisson distribution:

$$\mathbb{P}(\boldsymbol{d}|c) = \prod_{t \in \boldsymbol{\mathcal{V}}} \frac{\mathbb{P}(t|c)^{tf_{\boldsymbol{d}}(t)}}{tf_{\boldsymbol{d}}(t)!} e^{-\mathbb{P}(T=t|c)} \qquad \text{(Poisson)} \qquad (6.9)$$

Eyheramendy et al. (2003) showed that for classification, this is similar to the multinomial model.

6.1.1.4 *Maximum Likelihood Estimate*

Similarly to IDF, the probabilities $\mathbb{P}(t|c)$ are estimated on a training set using the Maximum Likelihood Estimate (MLE), because these values makes the observed data maximally likely. For the Bernoulli model, the probability should reflect the chance of a term of occurring in a document of class label c regardless of the actual number of times it occurs in documents while for the multinomial model, terms occurring multiples times should be favored: this is exactly the

difference between document frequency and collection frequency as introduced in Section 2.1.3. Therefore, the two quantities can be estimated as follows:

$$\hat{\mathbb{P}}(t|c) = \frac{df(t)}{N_c} \qquad \text{(Bernoulli)} \tag{6.10}$$

$$\hat{\mathbb{P}}(t|c) = \frac{cf(t)}{L_c} \qquad \text{(Multinomial)} \tag{6.11}$$

where N_c is the total number of documents of class label c in the training set and $L_c = \sum_t cf(t) = \sum_{\boldsymbol{d}} |\boldsymbol{d}|$ is the collection length (the length of the collection if interpreted as one mega-document consisting of all the documents of class label c in the training set appended to one another).

ADD-1 SMOOTHED PROBABILITIES Again, similarly to IDF, unseen terms in the training set will results in null estimates for $\hat{\mathbb{P}}(t|c)$ and because of the product, null estimates for $\hat{\mathbb{P}}(\boldsymbol{d}|c)$. Therefore, it is common in practice to smooth the probability estimates using add-one Lidstone smoothing, resulting in:

$$\hat{\mathbb{P}}(t|c) = \frac{df(t)+1}{N_c+2} \qquad \text{(Bernoulli)} \tag{6.12}$$

$$\hat{\mathbb{P}}(t|c) = \frac{cf(t)+1}{L_c+|\boldsymbol{\mathcal{V}}|} \qquad \text{(Multinomial)} \tag{6.13}$$

The difference in value in the denominator reflects the difference in the number of outcomes/types of random variables between the two models. For Bernoulli, a term appears or not in a document, with a uniform probability of $1/2$ (binary random variable E_t). For Multinomial, a term appears or not at a given position, with a uniform probability of $1/|\boldsymbol{\mathcal{V}}|$ (categorical random variable T).

6.1.1.5 *Log-likelihood*

Even with smoothed probability estimates, the values will be very low in practice, especially for large vocabulary. Therefore, multiplying these low values will result in numerical underflow. To prevent this, people have been estimating the **log-likelihood** instead since it does not change the classification decision and computes a sum of negative real-valued numbers that is less likely to overflow than the product to underflow. This results in:

$$c^* = \arg\max_{c \in \boldsymbol{\mathcal{C}}} \left(\log \hat{\mathbb{P}}(c) + \sum_{t \in \boldsymbol{\mathcal{V}}} \log \hat{\mathbb{P}}(t|c)^{e_t} \left(1 - \hat{\mathbb{P}}(t|c)\right)^{1-e_t} \right) \text{(Bernoulli)} \tag{6.14}$$

$$c^* = \arg\max_{c \in \boldsymbol{\mathcal{C}}} \left(\log \hat{\mathbb{P}}(c) + \sum_{t \in \boldsymbol{\mathcal{V}}} tf_{\boldsymbol{d}}(t) \times \log \hat{\mathbb{P}}(t|c) \right) \qquad \text{(Multinomial)} \tag{6.15}$$

6.1.1.6 *Online learning*

Naive Bayes is a simple (as in high bias, low variance) yet effective model in practice. In particular, for large-scale evolving datasets, it is straightforward to

take into account new data by just updating the relevant counts, which makes it suitable for **online learning**. It is also robust to noisy features and time-related changes in the data, e. g., *concept drift* (Forman, 2006) – the gradual change over time of the concept underlying a class – where the probability estimates will re-distribute between themselves the probability mass based on present use of terms (through their frequencies).

6.1.2 *Geometric classifiers*

We present in this section classifiers that rely on a geometric formulation of the problem and on minimizing a regularized loss function.

6.1.2.1 *Geometric formulation*

This time, we consider documents as points living in a p-dimensional vector space $\boldsymbol{\mathcal{X}}$ (i. e. the feature space in ML). If we restrict ourselves to binary classification (class label $y \in \boldsymbol{\mathcal{Y}} = \{-1, +1\}$ associated to each point $\boldsymbol{x} \in \boldsymbol{\mathcal{X}}$), we are trying to find an hyperplane (of equation $\boldsymbol{\theta}^\top \boldsymbol{x} + b = 0$) that separates the two classes as good as possible. The notion of "goodness" differs from model to model: not only we want to make as little classification mistakes as possible on the training set, i. e. maximize the **goodness of fit**, but we also want to make in the future as little prediction mistakes as possible, i. e. minimize the **generalization error**. For instance, if the problem is linearly separable on the training set, there are still an infinite number of hyperplanes that separate perfectly the data and which one to choose should be done according to its generalization power. The p-dimensional vector $\boldsymbol{\theta}$ corresponds to the **feature weight vector** and b is the intercept (sometimes called bias – not to be confused with the bias of the model, as in bias-variance trade-off).

NOTATIONS The ML literature is full of notations and abuse of notations. In particular, $\boldsymbol{\beta}$ and $\boldsymbol{w}$ are often used instead of $\boldsymbol{\theta}$ (but in TM w usually corresponds to a word and $\boldsymbol{w}$ to the corresponding word embedding). The class labels are sometimes in $\{0, 1\}$ but we think $\{-1, 1\}$ simplifies the expressions better. As for the intercept b, it is often integrated into the $\boldsymbol{\theta}^\top \boldsymbol{x}$ expression considering $(p+1)$-dimensional vectors with $\boldsymbol{\theta}_0 = b$ and $\boldsymbol{x}_0 = 1$ but since the intercept is not regularized (as we will see later on in Section 6.1.2.3), you end up having two different feature weights between the loss function and the penalty term. To simplify the expressions, we propose to define instead $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^\top \boldsymbol{x} + b$. For regression, this actually corresponds exactly to the prediction while for classification, its sign corresponds to the prediction (hence $\{-1, 1\}$ for the class labels). Finally, we will be using a lot the expression $y h_{\boldsymbol{\theta}}(\boldsymbol{x})$, which represents the relative distance to the hyperplane – absolute if $\|\boldsymbol{\theta}\|_2 = 1$ and positive (resp. negative) if $\boldsymbol{x}$ is well-classified (resp. mis-classified). It is also known as the functional margin (Manning, Raghavan, et al., 2008, p. 322).

6.1.2.2 *Loss function*

Given a model $h_{\boldsymbol{\theta}}$, a. k. a. **hypothesis** (in our case a set of feature weights $\boldsymbol{\theta}$ defining a specific hyperplane $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = 0$), we want to assess the **risk** $\mathcal{R}$, i. e. the expected classification error, ideally on $\boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{Y}}$:

$$\mathcal{R}(\boldsymbol{\theta}) = \int_{\boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{Y}}} \ell(y, h_{\boldsymbol{\theta}}(\boldsymbol{x})) d\,\mathbb{P}(\boldsymbol{x}, y) \tag{6.16}$$

where $\ell : \boldsymbol{\mathcal{Y}} \times \mathbb{R} \to \mathbb{R}_+$ is a **loss function** that quantifies the discrepancy between the true class label y and the prediction $h_{\boldsymbol{\theta}}(\boldsymbol{x})$. The problem of classification is thus reduced to finding the hypothesis that minimizes the risk. But the joint probability distribution $\mathbb{P}(\boldsymbol{x}, y) = \mathbb{P}(y|\boldsymbol{x})\,\mathbb{P}(\boldsymbol{x})$ is unknown and the only available information is contained in the training set $\{(\boldsymbol{x}^i, y^i)\}_{i=1\ldots N}$. Therefore, in practice, we estimate this risk on the training set, estimate known as the **empirical risk** $\hat{\mathcal{R}}$ (Vapnik, 1991):

$$\hat{\mathcal{R}}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \ell(y^i, h_{\boldsymbol{\theta}}(\boldsymbol{x}^i)) \qquad \text{(Empirical risk)} \tag{6.17}$$

We then try to minimize this risk, e. g., using (stochastic) gradient descent (Perceptron, LLSF, LR, SVM) or coordinate descent (AdaBoost).

SURROGATE LOSS Naturally, we would want to use the **0-1 loss function** for classification error, i. e. simply penalizing by a single unit any mis-classification. However, as shown on Figure 6.1, the 0-1 loss function is non-smooth, non-convex and makes the minimization problem NP-hard. Therefore, in practice, we use (convex) **surrogate loss functions** $\hat{\ell}$. Table 6.1 presents some common surrogate loss functions used in binary classification and their associated classifiers. Figure 6.1 plots their values as a function of $yh_{\boldsymbol{\theta}}(\boldsymbol{x})$. They are all *convex upper bounds* of the 0-1 loss function. Generally speaking, when the training examples are mis-classified ($yh_{\boldsymbol{\theta}}(\boldsymbol{x}) < 0$), the further away from the hyperplane, the higher the loss: for the hinge and log loss, (almost) linear; for the squared loss, quadratic; and for the exponential loss, well, exponential! Conversely, when the training examples are well-classified ($yh_{\boldsymbol{\theta}}(\boldsymbol{x}) > 0$), the loss tends to decrease as the distance to the separating hyperplane increases: the loss is actually null for the hinge loss from $yh_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq 1$ but never for the log and exponential loss – this actually explains one of the differences between LR and SVM: in the former, all the training examples contribute to the decision boundary, even slightly for the outliers, while in the latter, only the training examples for which the loss is non-null, the so-called *support vectors*. Note that the squared loss is the only symmetric loss and penalizes heavily points falling far from the separating hyperplane, regardless of the actual classification, which explain why it is used for linear regression mainly. We refer to (F. Li and Y. Yang, 2003) for a review of the classification methods used in TC and expressed in terms of loss functions, even for Naive Bayes.

Name	Expression	Classifier
0-1 loss	$\mathbb{1}_{yh_\theta(x)<0}$	optimal
exponential loss	$e^{-yh_\theta(x)}$	AdaBoost
hinge loss	$[1-yh_\theta(x)]_+$	SVM
log loss	$\log(1+e^{-yh_\theta(x)})$	LR
squared loss	$[1-yh_\theta(x)]^2$	LLSF

Table 6.1 – Common surrogate loss functions for binary classification.

6.1.2.3 *Regularization*

Only minimizing the empirical risk can lead to **overfitting**, that is, the model no longer learns the underlying pattern we are trying to capture but fits the noise contained in the training data and thus results in poorer generalization (e. g., lower performances on the test set). For instance, along with some feature space transformations to obtain non-linear decision boundaries in the original feature space, one could imagine a decision boundary that follows every quirk of the training data. Additionally, if two hypothesis lead to similar low empirical risks, one should select the "simpler" model for better generalization power, simplicity assessed using some measure of model complexity.

INTERPRETATION The concept of **regularization** encompasses all these ideas. It can be interpreted as (1) taking into account the **model complexity** by penalizing larger feature weights; (2) incorporating **prior knowledge** to help the learning by making prior assumptions on the feature weights and their distribution; and (3) helping compensate an **ill-posed problem** (e. g., typically when $X^\top X$ is not invertible in linear regression but $X^\top X + \lambda I$ is).

LOSS+PENALTY Regularization takes the form of additional *constraints* to the minimization problem, i. e. a budget on the feature weights, which are often relaxed into a **penalty term** controlled via a Lagrange multiplier λ. Therefore, the overall **expected risk** (Vapnik, 1991) is the weighted sum of two components: a loss function and a regularization penalty term, expression referred to as "Loss+Penalty" by Hastie et al. (2009, p. 426).

L1 AND L2 REGULARIZATION The two most used penalty terms are known as L^1-regularization, a. k. a. *Lasso* (Tibshirani, 1996), and L^2-regularization, a. k. a. *ridge* (Hoerl and Kennard, 1970) or *Tikhonov* (Tikhonov and Arsenin, 1977), since they correspond to penalizing the model with respectively the L^1-norm and L^2-norm of the feature weight vector $\boldsymbol{\theta}$:

$$\hat{\mathcal{R}}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \hat{\ell}(y^i, h_{\boldsymbol{\theta}}(\boldsymbol{x}^i)) + \lambda \sum_{j=1}^{p} |\boldsymbol{\theta}_j| \qquad (L^1\text{-regularized empirical risk})$$

$$\hat{\mathcal{R}}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \hat{\ell}(y^i, h_{\boldsymbol{\theta}}(\boldsymbol{x}^i)) + \lambda \sum_{j=1}^{p} \boldsymbol{\theta}_j{}^2 \qquad (L^2\text{-regularized empirical risk})$$

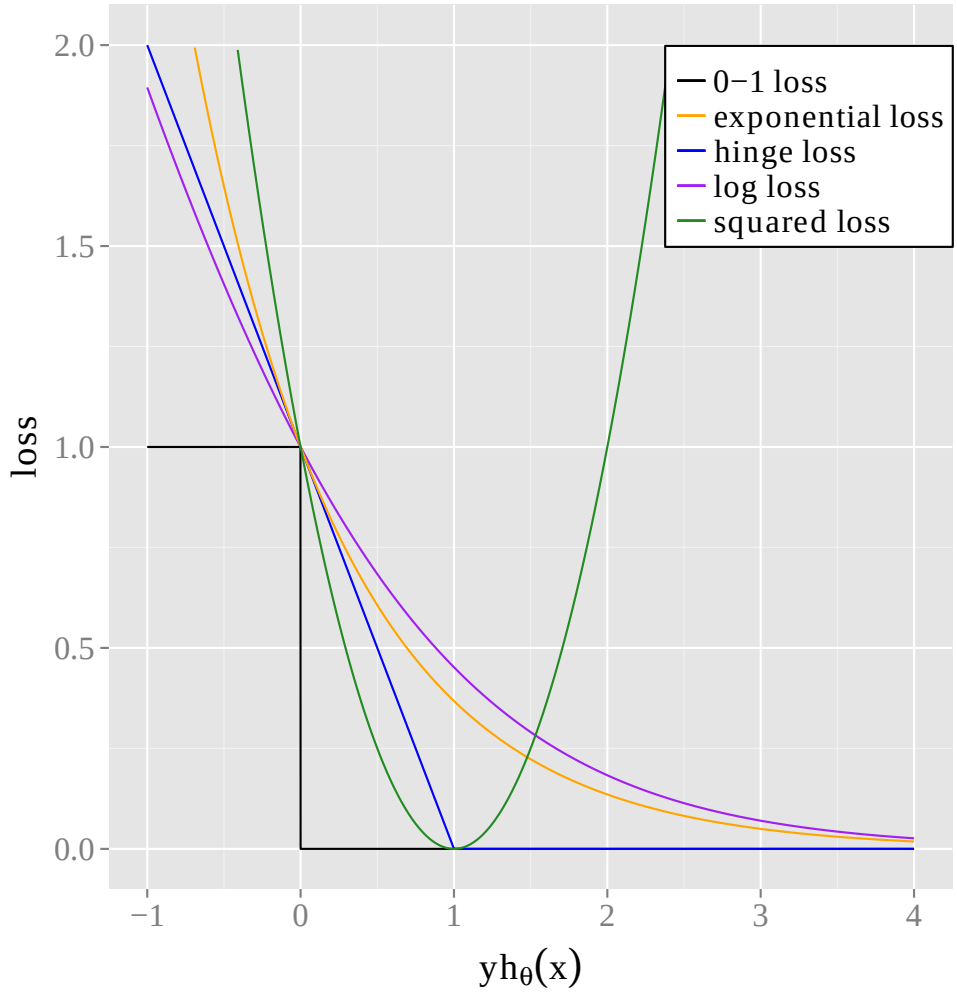


Figure 6.1 – Common surrogate loss functions for binary classification.

As aforementioned, we do not try to control the size of the intercept b via regularization because it describes the overall scale of the data rather than the relationship between $\boldsymbol{x}$ and y (and this is the reason why making $\boldsymbol{\theta}$ a $(p+1)$-dimensional vector is "dangerous").

PRIOR ON THE FEATURE WEIGHTS L^1- (resp. L^2-) regularization can be interpreted as adding a Laplacian (resp. Gaussian) prior on the feature weight vector. Indeed, given the training set, we want to find the most likely hyperplane, i. e. more generally the hypothesis h^* with *maximum a posteriori* probability:

$$\begin{aligned} h^* &= \arg\max_{h \in \mathcal{H}} \Big(\mathbb{P}(h | \{(\boldsymbol{x}^i, y^i)\}_{i=1\dots N}) \Big) \\ &= \arg\max_{h \in \mathcal{H}} \left(\frac{\mathbb{P}(y^1, \dots, y^N | \boldsymbol{x}^1, \dots, \boldsymbol{x}^N, h)\, \mathbb{P}(h | \boldsymbol{x}^1, \dots, \boldsymbol{x}^N)}{\mathbb{P}(y^1, \dots, y^N | \boldsymbol{x}^1, \dots, \boldsymbol{x}^N)} \right) \\ &= \arg\max_{h \in \mathcal{H}} \Big(\mathbb{P}(y^1, \dots, y^N | \boldsymbol{x}^1, \dots, \boldsymbol{x}^N, h)\, \mathbb{P}(h | \boldsymbol{x}^1, \dots, \boldsymbol{x}^N) \Big) \end{aligned}$$

$$= \arg\max_{h \in \mathcal{H}} \left(\mathbb{P}(y^1, \dots, y^N | \boldsymbol{x}^1, \dots, \boldsymbol{x}^N, h) \, \mathbb{P}(h) \right) \tag{6.18}$$

$$= \arg\max_{h \in \mathcal{H}} \left(\prod_{i=1}^{N} \left(\mathbb{P}(y^i | \boldsymbol{x}^i, h) \right) \mathbb{P}(h) \right) \tag{6.19}$$

$$= \arg\max_{h \in \mathcal{H}} \left(\sum_{i=1}^{N} \left(\log \mathbb{P}(y^i | \boldsymbol{x}^i, h) \right) + \log \mathbb{P}(h) \right)$$

$$= \arg\min_{h \in \mathcal{H}} \left(\sum_{i=1}^{N} \left(\underbrace{-\log \mathbb{P}(y^i | \boldsymbol{x}^i, h)}_{\text{loss function}} \right) \underbrace{-\log \mathbb{P}(h)}_{\text{regularization term}} \right) \tag{6.20}$$

We assumed in particular that the hypothesis does not depend on the examples alone (Equation 6.18) and that the N training labeled examples are drawn from an independent and identically distributed (i.i.d.) sample (Equation 6.19). In that last form (Equation 6.20), we see that the loss function can be interpreted as a **negative log-likelihood** and the regularization penalty term as a **negative log-prior** over the hypothesis.

Therefore, if we assume a multivariate Gaussian prior on the feature weight vector of mean vector $\mathbf{0}$ and covariance matrix $\Sigma = \sigma^2 I$ (i.e. independent features of same prior standard deviation σ), we do obtain the L^2-regularization:

$$\begin{aligned} \mathbb{P}(h) &= \frac{1}{\sqrt{(2\pi)^p |\Sigma|}} e^{-\frac{1}{2}\boldsymbol{\theta}^\top \Sigma^{-1} \boldsymbol{\theta}} && \text{(Gaussian prior)} \\ \Rightarrow -\log \mathbb{P}(h) &= \frac{1}{2\sigma^2} \boldsymbol{\theta}^\top I \boldsymbol{\theta} + \frac{p}{2} \log(2\pi\sigma) \\ &\overset{\text{argmax}}{=} \lambda \|\boldsymbol{\theta}\|_2{}^2, \quad \lambda = \frac{1}{2\sigma^2} && (L^2\text{-regularization}) \end{aligned} \tag{6.21}$$

And similarly, if we assume a multivariate Laplacian prior on the feature weight vector (i.e. $\boldsymbol{\theta}_i \sim Laplace(0, \frac{1}{\lambda})$), we obtain L^1-regularization. In practice, in both cases, the priors basically mean that we expect weights around 0 on average. The main difference between L^1- and L^2-regularization is that the Laplacian prior will result in explicitly setting some feature weights to 0 (**feature sparsity**) while the Gaussian prior will only result in reducing their values (**shrinkage**).

L0 REGULARIZATION Note that when defining the L^0-norm of a vector as the number of non-zero values, L^0-regularization corresponds to **best subset selection** (Foster and George, 1994), i.e. limiting the number of features regardless of their weights. However, the resulting minimization problem is *non-convex*, actually NP-hard (Natarajan, 1995), which makes it difficult to solve computationally.

6.1.2.4 *Logistic regression*

Binomial Logistic Regression (LR) is to some extent the equivalent of linear regression for binary classification. Since the outcome y is categorical and not numerical (the actual difference between a regression and a classification

problem), we cannot directly apply standard linear regression techniques such as LLSF to the outcome but instead, we could try to model $\mathbb{P}(y|\boldsymbol{x}, h)$ as a linear combination of the features, i. e. in the form of $h_{\boldsymbol{\theta}}(\boldsymbol{x})$. However, probabilities are bounded and linear functions are not – any increment in any feature value will result in an increment in the outcome. If we consider $\log \mathbb{P}(y|\boldsymbol{x}, h)$ then an increment will result in a multiplication, which is closest to what we would want, but $\log \mathbb{P}(y|\boldsymbol{x}, h) < 0$ and linear functions are not. Finally, if we consider $\log\left(\mathbb{P}(y|\boldsymbol{x},h)/1-\mathbb{P}(y|\boldsymbol{x},h)\right)$, the **logit** (a. k. a. log-odds) of $\mathbb{P}(y|\boldsymbol{x}, h)$, then it can be a linear combination of the features, which results in assuming that:

$$\log\left(\frac{\mathbb{P}(y|\boldsymbol{x}, h)}{1 - \mathbb{P}(y|\boldsymbol{x}, h)}\right) = h_{\boldsymbol{\theta}}(\boldsymbol{x}) \tag{6.22}$$

$$\Rightarrow \mathbb{P}(y|\boldsymbol{x}, h) = \frac{1}{1 + e^{-yh_{\boldsymbol{\theta}}(\boldsymbol{x})}} \tag{6.23}$$

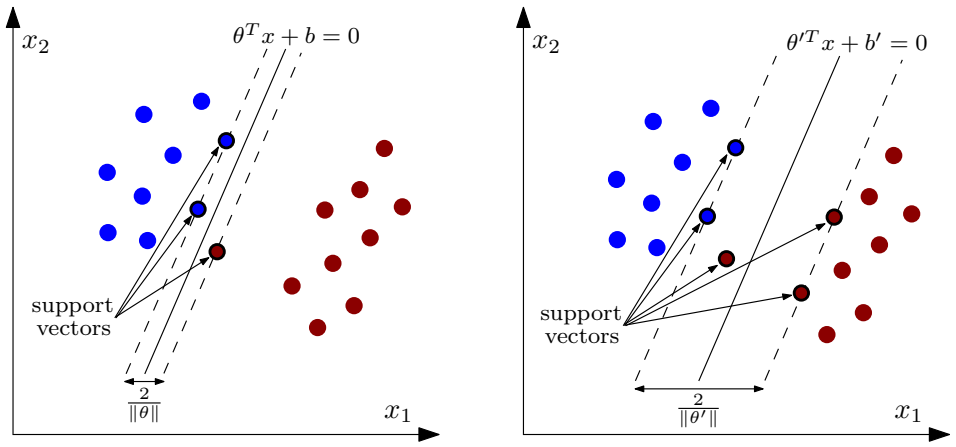
$\pi : s \mapsto \frac{1}{1+e^{-s}}$ is known as the **logistic function**, which is a sigmoid function and the probability can be directly linked to the value of $yh_{\boldsymbol{\theta}}(\boldsymbol{x})$ that corresponds to the relative distance to the hyperplane (cf. Section 6.1.2.1). For instance, for $y = +1$, the larger $h_{\boldsymbol{\theta}}(\boldsymbol{x})$, the further away from the hyperplane in the "right direction" and the higher the probability $\mathbb{P}(y = +1|\boldsymbol{x}, h)$ ($\lim_{s\to+\infty} \pi(s) = 1$). Conversely, the further away from the hyperplane in the "wrong direction" and the lower the probability $\mathbb{P}(y = +1|\boldsymbol{x}, h)$ ($\lim_{s\to-\infty} \pi(s) = 0$).

GENERATIVE VS. DISCRIMINATIVE MODELS As opposed to Naive Bayes (NB) that models the *joint* probability $\mathbb{P}(y, \boldsymbol{x})$ and then picks the most likely label y, LR models directly the *posterior* probability $\mathbb{P}(y|\boldsymbol{x})$. For classification, this makes little difference but if we wanted to generate a new document for a given class, $\mathbb{P}(y|\boldsymbol{x})$ could only tell us how likely a document belong to a class, which is not enough. This is the difference between a *generative* and a *discriminative* classifier. In particular, NB and LR form what is called a "generative-discriminative" pair (A. Y. Ng and Jordan, 2002).

LOG LOSS In terms of loss function, plugging in the expression of $\mathbb{P}(y|\boldsymbol{x}, h)$ from Equation 6.23 into Equation 6.20 immediately gives the log loss presented in Table 6.1. This is the power of expressing a prediction problem as a loss function minimization one – by a simple change of loss function or regularization term, one can consider successive regression or classification models easily.

6.1.2.5 *SVM*

Support Vector Machine (SVM) is a binary classifier that tries to maximize the margin between the two classes around the separating hyperplane as illustrated on Figure 6.2 (point color indicates the class label of the point, blue vs. red). The idea was initiated in (Vapnik and Lerner, 1963) and further extended in (Vapnik, 1982) around the *statistical learning theory*. In its so-called "hard margin" initial formulation, we assume a linearly separable problem (not necessarily in the



(a) Hard margin SVM.

(b) Soft margin SVM.

Figure 6.2 – Illustration of hard and soft margin SVM, mis-classification allowed if it increases the margin (soft) or not allowed at all (hard). We assumed a linearly separable problem in some feature space.

original feature space $\mathcal{X}$). The distance between a point x^0 and the hyperplane $\theta^\top x + b = 0$ corresponds to the projection of $x^0 - x$ onto θ, which is:

$$\text{proj}_\theta (x^0 - x) = \frac{|\theta^\top (x^0 - x)|}{\|\theta\|_2} = \frac{|\theta^\top x^0 - \theta^\top x)|}{\|\theta\|_2} = \frac{|\theta^\top x^0 + b|}{\|\theta\|_2}$$

MAXIMIZING THE MARGIN Since we are trying to maximize the margin around the hyperplane, we can choose θ and b such that the positive examples closest to the hyperplane satisfy $\theta^\top x + b = +1$ and the negative examples closest to the hyperplane satisfy $\theta^\top x + b = -1$. In this scenario, the margin is thus $\frac{2}{\|\theta\|_2}$ and $\forall i \in [\![1, N]\!]$, $y^i(\theta^\top x^i + b) \geq 1$. Maximizing the margin is equivalent to minimizing its inverse or even better its squared inverse. Therefore, the problem can be expressed as finding θ^* such that:

$$\begin{aligned} \theta^* = & \underset{\theta \in \mathbb{R}^p,\ b \in \mathbb{R}}{\arg\min} \ \|\theta\|_2{}^2 \\ & \text{subject to } 1 - y^i(\theta^\top x^i + b) \leq 0 \ \forall i \in [\![1, N]\!] \end{aligned} \tag{6.24}$$

which is a **constrained convex minimization problem**. We refer to the book of Boyd and Vandenberghe (2004) for the theory behind *convex optimization*. We note here that both the function to minimize and the constraints are convex.

LAGRANGIAN If we relax this constrained problem using a vector α of positive Lagrange multipliers, we obtain:

$$\theta^* = \underset{\theta \in \mathbb{R}^p,\ b \in \mathbb{R}}{\arg\min} \ \underset{\alpha \in \mathbb{R}_+^N}{\arg\max} \underbrace{\left(\|\theta\|_2{}^2 + \sum_{i=1}^{N} \alpha_i \left(1 - y^i(\theta^\top x^i + b)\right) \right)}_{\mathcal{L}(\theta, b, \alpha)} \tag{6.25}$$

The expression $\mathcal{L}(\theta, b, \alpha)$ (the Lagrangian) is very similar to the "Loss + Penalty" formulation we have previously seen, with L^2-regularization and **hinge loss**.

This time, we can interpret the L^2-regularization in terms of the inverse of the margin width. Indeed, the wider the margin, the better the generalization we can expect. $\boldsymbol{\alpha}$ weights the loss over the regularization and has an inverse behavior compared to the classic λ used in the "Loss+Penalty" formulation (cf. Section 6.1.2.3).

OPTIMALITY Resolving further the convex optimization problem, the Karush-Kuhn-Tucker (KKT) conditions give us at the optimum:

— *stationarity* $\Rightarrow \nabla_{\boldsymbol{\theta}} \mathcal{L} = 0, \frac{\partial \mathcal{L}}{\partial b} = 0$
$\Rightarrow \boldsymbol{\theta} = \frac{1}{2} \sum_{i=1}^{N} \boldsymbol{\alpha}_i y^i \boldsymbol{x}^i, \sum_{i=1}^{N} \boldsymbol{\alpha}_i y^i = 0$

— *complementary slackness* $\Rightarrow \boldsymbol{\alpha}_i \left(1 - y^i(\boldsymbol{\theta}^\top \boldsymbol{x}^i + b)\right) = 0 \; \forall i \in [\![1, N]\!]$

DUAL FORMULATION On one hand, plugging in the expression for $\boldsymbol{\theta}$ from the stationarity condition in the Lagrangian (Equation 6.25) gives the **dual formulation** of the problem (the Wolfe):

$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha} \in \mathbb{R}_+^N}{\arg\max} \left(\sum_{i=1}^{N} \boldsymbol{\alpha}_i - \frac{1}{4} \sum_{i=1}^{N} \sum_{j=1}^{N} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j y^i y^j \boldsymbol{x}^{i\top} \boldsymbol{x}^j \right) \tag{6.26}$$

We see that the dual maximization problem only depends on $\boldsymbol{x}^{i\top} \boldsymbol{x}^j$ but never on $\boldsymbol{x}^i$ alone. Moreover, if we were to change the feature space using a mapping function $\phi : \boldsymbol{\mathcal{X}} \to \mathbb{H}$, it would still only depends on $\langle \phi(\boldsymbol{x}^i), \phi(\boldsymbol{x}^j) \rangle$ and not on $\phi(\boldsymbol{x}^i)$ alone (where $\langle \cdot, \cdot \rangle$ is the inner product in $\mathbb{H}$). Therefore, we define a **kernel** function $K : \boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{X}} \to \mathbb{R}_+$ between two points such that $K(\boldsymbol{x}^i, \boldsymbol{x}^j) = \langle \phi(\boldsymbol{x}^i), \phi(\boldsymbol{x}^j) \rangle$ without explicitly having to define ϕ. We just need to make sure that it is an inner product in some Hilbert space $\mathbb{H}$, i. e. that K satisfies Mercer's condition (Mercer, 1909). This avoidance is known as the **kernel trick**, which first appeared in (Aizerman et al., 1964) for the *kernel perceptron* and later in (Boser et al., 1992) for the SVM. A kernel function can be interpreted as a similarity metric between two objects (e. g., documents in our case) and the *kernel matrix* corresponds to the symmetric matrix of dimension $N \times N$ filled with the values $K(\boldsymbol{x}^i, \boldsymbol{x}^j)$, basically the pairwise similarities. The choice of solving the problem in the dual rather than in the primal depends on (1) the ratio between N and p: if $N \ll p$ then the kernel matrix might actually requires less memory than the original $\boldsymbol{\mathcal{X}}$ space (depending on its sparsity); and (2) if we want to avoid the explicit feature mapping ϕ at the collection level – in many cases, when defining specific kernels between objects, we still explicitly consider features when comparing two objects but we do not need to list them all beforehand. For instance, when measuring the similarity between two graphs based on the number of shared random walks, we want to avoid to have to list all the random walks of all the graphs in the training set at once.

SUPPORT VECTORS On the other hand, the complementary slackness condition gives birth to the **support vectors**. For any vector not on the margin (i. e. $1 - y^i(\boldsymbol{\theta}^\top \boldsymbol{x}^i + b) < 0$), the corresponding $\boldsymbol{\alpha}_i$ must be equal to 0 and thus

the vector has no impact on the value of $\boldsymbol{\theta}$. Therefore, $\boldsymbol{\theta}$ and the hyperplane it defines only depends on the vectors on the margin, the ones that "support" it. Moreover, on the validation and test set, we actually only need these vectors to make predictions. As a rule of thumb, the ratio between the number of support vectors and the total number of vectors in training gives an upper bound on the generalization error (Vapnik, 1995). Indeed, intuitively, a model that needs only a few support vectors to describe it can be expected to perform better on unseen data.

HARD VS. SOFT MARGIN So far, we have been considering a linearly separable problem in some feature space, not necessarily the original one thanks to the kernel trick. Cortes and Vapnik (1995) proposed a "soft margin" formulation of the problem, allowing for some mis-classifications on the training set controlled via some slack variables. In this configuration, any vector on the margin or inside it or on the wrong side is a support vector. Actually, for better generalization power, even when the problem is linearly separable, it might better to mis-classify or at least have an example within the margin if it increases greatly the margin width as illustrated on Figure 6.2b.

MULTI-CLASS CLASSIFICATION SVM is designed for binary classification since its hypothesis corresponds to a separating hyperplane. As aforementioned, TC is in general a multi-class classification problem with $|\mathcal{C}|$ categories. Rather than extending the definition of the 0-1 loss and its convex upper bounds for the multi-class problem, people have been training instead multiple binary classifiers, either $\binom{|\mathcal{C}|}{2}$ classifiers in a **one-vs-one** scenario, i. e. the machine learns a model for each pair of categories, (Friedman, 1996) or $|\mathcal{C}|$ classifiers in a **one-vs-all** (a. k. a. one-vs-the-rest) scenario, i. e. the machine learns a model for each category considering the data for all the other categories as just negative examples. The former requires more classifiers but they are trained on less data each. Between all the classifiers' outputs, we then select the category with highest probability, probability obtained from the prediction $h_{\boldsymbol{\theta}}(\boldsymbol{x})$ transformed using the logistic function from Section 6.1.2.4 (Platt, 1999). Note that the kernel matrix only needs to be computed once in any case.

6.2 TEXT CATEGORIZATION AS A GRAPH CLASSIFICATION PROBLEM

So far, we have been reviewing the different approaches to learn a classification model from labeled data. We took as supporting example the task of Text Categorization (TC) where features were terms of the vocabulary and documents feature vectors. In this section, we consider the task of TC as a graph classification problem. By representing documents as graph-of-words instead of historical n-gram bag-of-words, we extract subgraphs as features, which turned out empirically to be more discriminative than standard n-grams. The learning part is the same but the feature space is different. This work was presented in (Rousseau, Kiagias, et al., 2015).

TW-IDF While still considering bag-of-words document representation and unigram features, the first idea that comes to mind is to use TW-IDF instead of TF×IDF as a term weighting scheme, i. e. graph-based feature values extracted from graph-of-words document representations like we did for ad hoc IR in Chapter 4. This has already been explored by Hassan et al. (2007), Valle and Öztürk (2011), Amancio et al. (2011), and Malliaros and Skianis (2015) with various centrality measures (cf. Section 5.1.2). Therefore, we decided to go beyond and consider subgraphs as features instead and better capitalize on the graph representation.

6.2.1 *Model definition*

For text categorization, we represented each document as an **unweighted undirected graph-of-words** with a fixed sliding window of size 4 (cf. Figure 3.1 for illustration). We did not consider edge weight so as to simplify the definition of subgraph matching between documents. Indeed, in graph classification, especially when classifying chemical compounds, researchers have been making a distinction between edge labels, e. g., between covalent bonds *C*–*C* and *C*=*C*, but in our case the weights are numerical and not categorical, which would need to be taken into account. For instance, if two edges between the same terms have different weights, can we still consider it a match? If so, does it depend on the difference between the weights? Questions that are not straightforward and therefore left for further research. As for edge directionality, since the goal is to relax the definition of *n*-grams and capture for instance word inversion, we did not consider it either on purpose. We present next an overview of the literature on graph classification and then the procedure we followed to extract the most frequent subgraphs of a collection of graphs that will serve as features and to control the minimum document frequency.

6.2.1.1 *Literature review on graph classification*

Graph classification corresponds to the task of automatically predicting the class label of a given graph. The learning part in itself does not differ from other supervised learning problems and most proposed methods deal with the feature extraction part. They fall into two main categories: approaches that consider **subgraphs as features** and **graph kernels**.

SUBGRAPHS AS FEATURES The main idea is to mine frequent subgraphs and use them as features for classification, be it with Adaboost (Kudo, Maeda, et al., 2004) or a linear SVM (Deshpande et al., 2005). Indeed, most datasets that were used in the associated experiments correspond to chemical compounds where repeating substructure patterns are good indicators of belonging to one particular class. Some popular graph pattern mining algorithms are gSpan (Yan and Han, 2002), FFSM (Huan et al., 2003) and Gaston (Nijssen and Kok, 2004). The number of frequent subgraphs can be enormous, especially for large

graph collections, and handling such a feature set can be very expensive. To overcome this issue, recent works have proposed to retain or even only mine the discriminative subgraphs, i. e. features that contribute to the classification decision, in particular gBoost (Saigo et al., 2009), CORK (Thoma et al., 2009) and GAIA (Jin et al., 2010). However, when experimenting, gBoost did not converge on our larger datasets while GAIA and CORK consider subgraphs of node size at least 2, which exclude unigrams in the case of graph-of-words, resulting in poorer performances. Moreover, all these approaches have been developed for binary classification, which meant mining features as many times as the number of classes instead of just once (one-vs-all learning strategy). In our research, we tackled the **scalability issue** differently through an unsupervised feature selection approach to reduce the size of the graphs and a fortiori the number of frequent subgraphs (cf. Section 6.3).

GRAPH KERNELS Gärtner et al. (2003a) proposed the first kernels *between* graphs (as opposed to previous kernels *on* graphs, i. e. between nodes) based on shared *random walks* to tackle the problem of classification between graphs. Other sub-structures were proposed later on, including *subtrees* (Ramon and Gärtner, 2003), *cycles* (Horváth et al., 2004) and *shortest paths* (Borgwardt and Kriegel, 2005). In parallel, the idea of marginalized kernels was extended to graphs by Kashima et al. (2003) and by Mahé et al. (2004). We refer to the survey by Vishwanathan et al. (2010) for an in-depth review of the topic and in particular its limitations in terms of number of unique node labels, which make them unsuitable for our problem as tested in practice (limited to a few tens of unique labels compared to hundreds of thousands for us). Nevertheless, with some colleagues, we recently started to explore shortest-path kernels between graph-of-words with promising results in a wide range of applications based on document similarity, leading to a paper submission (Meladianos et al., 2015b).

APPLICATIONS TO NLP Kudo and Y. Matsumoto (2004) and S. Matsumoto et al. (2005) used as features frequent subtrees extracted from dependency and syntactic parse tree representations, i. e. syntactic graph-of-words. Similarly, Jiang et al. (2010) and Arora, Mayfield, et al. (2010) mined frequent subgraphs from their annotation graphs that are very close to respectively parse and dependency trees, i. e. still syntactic graph-of-words. The work of Markov et al. (2007) is perhaps the only one really close to what we are trying to achieve here. They capitalized on the statistical graph-of-words definition from Schenker (2003) to extract frequent subgraphs using their own supervised frequent subgraph miner, features subsequently fed to NB and C4.5 (a type of tree-based classifiers). In all cases though, the choice of the support value, which controls the total number of features and can potentially lead to millions of subgraphs in the case of text categorization datasets, was never properly discussed and left to the user. Moreover, subgraphs of syntactic graph-of-words have a different interpretation than the one we made in Section 3.2.3 and therefore it is harder to compare them directly with the standard n-gram features.

6.2.1.2 *Unsupervised subgraph feature mining using gSpan*

We considered the task of TC as a graph classification problem by representing textual documents as graph-of-words and then extracting subgraph features to train a graph classifier. Each document is a separate graph-of-words and the collection of documents thus corresponds to a set of graphs. Therefore, for larger datasets, the total number of graphs increases but not the average graph size (the average number of unique terms in a text), assuming homogeneous datasets.

NOTE ON KERNELS Because the total number of unique node labels corresponds to the number of unique terms in the collection in our case, graph kernels are not suitable for us as verified in practice using the MATLAB code made available by Shervashidze (2009). We therefore only explored the methods that consider subgraphs as features.

INTUITION Repeating substructure patterns between graphs are intuitively good candidates for classification since, at least for chemical compounds, shared subparts of molecules are good indicators of belonging to one particular class, e. g., predicting carcinogenicity in molecules (Helma et al., 2001). We assumed it would the same for text. Indeed, subgraphs of graph-of-words correspond to sets of words co-occurring together, just not necessarily always as the same sequence like for n-grams – it can be seen as a relaxed definition of a n-gram to capture additional variants because of *word inversion* and *subset matching* for instance (cf. Section 3.1.1).

SUBGRAPH MATCHING In graph classification, it is common to introduce a node labeling function μ to map a node id to its label. For instance, consider the case of chemical compounds (e. g., the benzene C_6H_6). Then in its graph representation (its "structural formula"), it is crucial to differentiate between the multiple nodes labeled the same (e. g., C or H). In the case of statistical graph-of-words, node labels are unique inside a graph since they represent unique terms of the document and we can therefore omit these functions since they are injective in our case and we can substitute node ids for node labels. In particular, the general problem of **subgraph matching**, which defines an isomorphism between a graph and a subgraph and is NP-complete (Garey and D. S. Johnson, 1990), can be reduced to a polynomial problem when node labels are unique. In our experiments, we used the standard algorithm VF2 developed by Cordella et al. (2001).

GSPAN We used **gSpan**, graph-based Substructure pattern (Yan and Han, 2002), as frequent subgraph miner like Jiang et al. (2010) and Arora, Mayfield, et al. (2010) mostly because of its fast available C++ implementation from gBoost (Saigo et al., 2009). Briefly, the key idea behind gSpan is that instead of enumerating all the subgraphs and testing for isomorphism throughout the

collection, it first builds for each graph a lexicographic order of all the edges using Depth First Search (DFS) traversal and assigns to it a unique minimum DFS code. Based on all these DFS codes, a hierarchical search tree is constructed at the collection-level. By pre-order traversal of this tree, gSpan discovers all frequent subgraphs with required support.

SUPPORT Consider the set of all subgraphs in the collection of graphs, which corresponds to the set of all potential features. Note that there may be overlapping (subgraphs sharing nodes/edges) and redundant (subgraphs included in others) features since we do not restrict their definition to induced subgraphs (cf. Section 3.2.3). Because the size of this set is exponential in the number of edges (just like the number of n-grams is exponential in n), it is common to only retain or even mine the most frequent subgraphs (again, just like for n-grams with a minimum document frequency (Fürnkranz, 1998; Joachims, 1998)). This is controlled via a parameter known as the **support**, which sets the minimum number of graphs in which a given subgraph has to appear to be considered as a feature, i. e. the number of subgraph matches in the collection. Here, since node labels are unique inside a graph-of-words, we do not have to consider multiple occurrences of the same subgraph in a given graph. The lower the support, the more features selected/considered but the more expensive the mining and the training (not only in time spent for the learning but also for the feature vector generation).

6.2.1.3 *Unsupervised support selection*

The optimal value for the support can be learned through cross-validation so as to maximize the prediction accuracy of the subsequent classifier, making the whole feature mining process *supervised*. But if we consider that the classifier can only improve its goodness of fit with more features (the sets of features being nested as the support varies), it is likely that the lowest support will lead to the best test accuracy; assuming subsequent regularization to prevent overfitting (cf. Section 6.1.2.3). However, this will come at the cost of an exponential number of features as observed in practice. Indeed, as the support decreases, the number of features increases slightly up until a point where it increases exponentially, which makes both the feature vector generation and the learning expensive, especially with multiple classes. Moreover, we observed that the prediction performances did not benefit that much from using all the possible features (support of 1) as opposed to a more manageable number of features corresponding to a higher support. Therefore, we proposed to select the support using the so-called *elbow method*, an unsupervised empirical selection method previously introduced (cf. Section 5.2.2.3). Figure 6.3 (upper plots) in the experiments section will illustrate this process.

MULTICLASS SCENARIO In standard binary graph classification, e. g., predicting chemical compounds' carcinogenicity as either positive or negative (Helma et al., 2001), feature mining is performed on the whole graph collection as we

expect the mined features to be able to discriminate between the two classes (thus producing a good classifier). However, for the task of TC, there are usually more than two classes (e. g., 118 categories of news articles for the Reuters-21578 dataset) and with a skewed class distribution (e. g., a lot more news related to "acquisition" than to "grain"). Therefore, a single support value might lead to some classes generating a tremendous number of features (e. g., hundreds of thousands of frequent subgraphs) and some others only a few (e. g., a few hundreds subgraphs) resulting in a skewed and non-discriminative feature set. To include discriminative features for these *minority* classes, we would need an extremely low support resulting in an exponential number of features because of the *majority* classes. For these reasons, we decided to mine frequent subgraphs per class using the same relative support (%) and then aggregating each feature set into a global one at the cost of a *supervised* process (but which still avoids cross-validated parameter tuning). This was not needed for the tasks of spam detection and opinion mining since the corresponding datasets consist of only two balanced classes.

6.2.2 *Experiments*

In this section, we present the experiments we carried out to validate our proposed novel features. We first describe the datasets, the evaluation metrics, the platform and the considered models. We then report the results we obtained and discuss their interpretations.

6.2.2.1 *Datasets, evaluation metrics, platform and models*

DATASETS We used four standard text datasets: two for multi-class document categorization (`WebKB` and `R8`), one for spam detection (`LingSpam`) and one for opinion mining (`Amazon`) so as to cover all the main subtasks of TC. `WebKB` consists of the 4 most frequent categories ("project", "course", "faculty" and "student") among labeled webpages from various CS departments – split into 2,803 for training and 1,396 for test (Cardoso-Cachopo, 2007, p. 39–41). `R8` consists of the 8 most frequent categories ("acq", "crude", "arn", "grain" "interest", "money-fx", "ship" and "trade") of Reuters-21578, a set of labeled news articles from the 1987 Reuters newswire – split into 5,485 for training and 2,189 for test (Debole and Sebastiani, 2005). `LingSpam` consists of 2,893 emails classified as spam or legitimate messages – split into 10 sets for 10-fold cross validation (Androutsopoulos et al., 2000). `Amazon` consists of 8,000 product reviews over four different sub-collections (books, DVDs, electronics and kitchen appliances) classified as positive or negative – split into 1,600 for training and 400 for test each (Blitzer et al., 2007).

EVALUATION METRICS To evaluate the performance of our proposed approaches over standard baselines, we computed on the test set both the micro- and macro-average F1-score (cf. Section 2.3.3.2). Because we are dealing with

single-label classification, the micro-average F1-score corresponds to the accuracy and is a measure of the overall prediction effectiveness (cf. Section 2.3.3.2). Conversely, the macro-average F1-score takes into account the skewed class label distributions by weighting each class uniformly. The statistical significance of improvement in accuracy over the n-gram SVM baseline was assessed using the micro sign test considering one-sided p-values less than 0.05 to reject the null hypothesis (cf. Section 2.3.4). For the Amazon dataset, we report the average of each metric over the four sub-collections.

PLATFORM We developed our approaches mostly in Python using the `igraph` library for the graph representation. For unsupervised subgraph feature mining, we used the C++ implementation of gSpan from gBoost (Saigo et al., 2009). Finally for classification and standard n-gram TC, we used `scikit-learn`, a standard Python ML library. For all the datasets, the preprocessing steps involved stop word removal and Porter's stemming.

MODELS In text categorization, standard baseline classifiers include k-Nearest Neighbors (kNN) (Larkey and Croft, 1996), Naive Bayes (NB) (McCallum and Nigam, 1998) and linear Support Vector Machine (SVM) (Joachims, 1998) with the latter performing the best on n-gram features as verified in our experiments. We used binary features on all datasets since TF×IDF weighting schemes are less impactful in TC than in ad hoc IR and which set of normalizations to choose is not consistent across datasets. Our interpretation is that in ad hoc IR people are trying to rank the documents all relevant to a query, so somewhat similar and therefore the frequency of a term is much more important than when trying to cluster similar documents in the first place. Since our subgraph features correspond to "long-distance n-grams" (cf. Section 3.2.3), we used linear SVMs as our classifiers in all our experiments – the goal of our work being to explore and propose better features rather than a different classifier.

6.2.2.2 *Classification results*

The two tables from Table 6.2 show the results on the four considered datasets. The first three rows correspond to the baselines: unsupervised n-gram feature extraction and then supervised learning using kNN, NB (Multinomial but Bernoulli yields similar results) and linear SVM. The last row "gSpan + SVM" corresponds to our approach: (un)supervised subgraph-of-words feature extraction using gSpan and then supervised learning using linear SVM. gSpan mining support values are 1.6% (`WebKB|`, 7% (`R8|`, 4% (`LingSpam`) and 0.5% (`Amazon`). We see that our model is consistently better than the baselines on all datasets, both in terms of accuracy and F1-score.

Method \ Dataset	WebKB		R8	
	Accuracy	**F1-score**	**Accuracy**	**F1-score**
kNN (k=5)	0.679	0.617	0.894	0.705
NB (Multinomial)	0.866	0.861	0.934	0.839
linear SVM	0.889	0.871	0.947	0.858
gSpan + SVM	**0.912***	**0.882**	**0.955***	**0.864**

Method \ Dataset	LingSpam		Amazon	
	Accuracy	**F1-score**	**Accuracy**	**F1-score**
kNN (k=5)	0.910	0.774	0.512	0.644
NB (Multinomial)	0.990	0.971	0.768	0.767
linear SVM	**0.991**	**0.973**	0.792	0.790
gSpan + SVM	**0.991**	0.972	**0.798***	**0.795**

Table 6.2 – Effectiveness results (accuracy and macro-average F1-score in test) of long-distance n-grams over standard ones. Bold font marks the best performance in a column. * indicates statistical significance at $p < 0.05$ using the micro sign test against the SVM baseline of the same column.

6.2.2.3 *Unsupervised support selection*

Figure 6.3 illustrates the unsupervised heuristic (elbow method) we used to select the support value, which corresponds to the minimum number of graphs in which a subgraph has to appear to be considered frequent. We noticed that as the support decreases, the number of features increases slightly up until a point where it increases exponentially. This support value, highlighted in black square on the figures and chosen before taking into account the class label, is the value we used in our experiments and for which we report the results in Table 6.2. The lower plots provide evidence that the elbow method helps selecting in an unsupervised manner a support that leads to the best or close to the best accuracy (the dashed red line indicates the accuracy in test for the SVM baseline).

6.2.2.4 *Distribution of mined n-grams, standard and long-distance ones*

In order to gain more insights on why the long-distance n-grams mined with gSpan result in better classification performances than the baseline n-grams, we computed the distribution of the number of unigrams, bigrams, etc. up to 6-grams in the traditional feature set and ours (Figure 6.4a) as well as in the top 5% features that contribute the most to the classification decision of the trained SVM (Figure 6.4b). Again, a long-distance n-gram corresponds to a subgraph of size n in a graph-of-words and can be seen as a relaxed definition of the traditional n-gram, one that takes into account word inversion and subset matching for

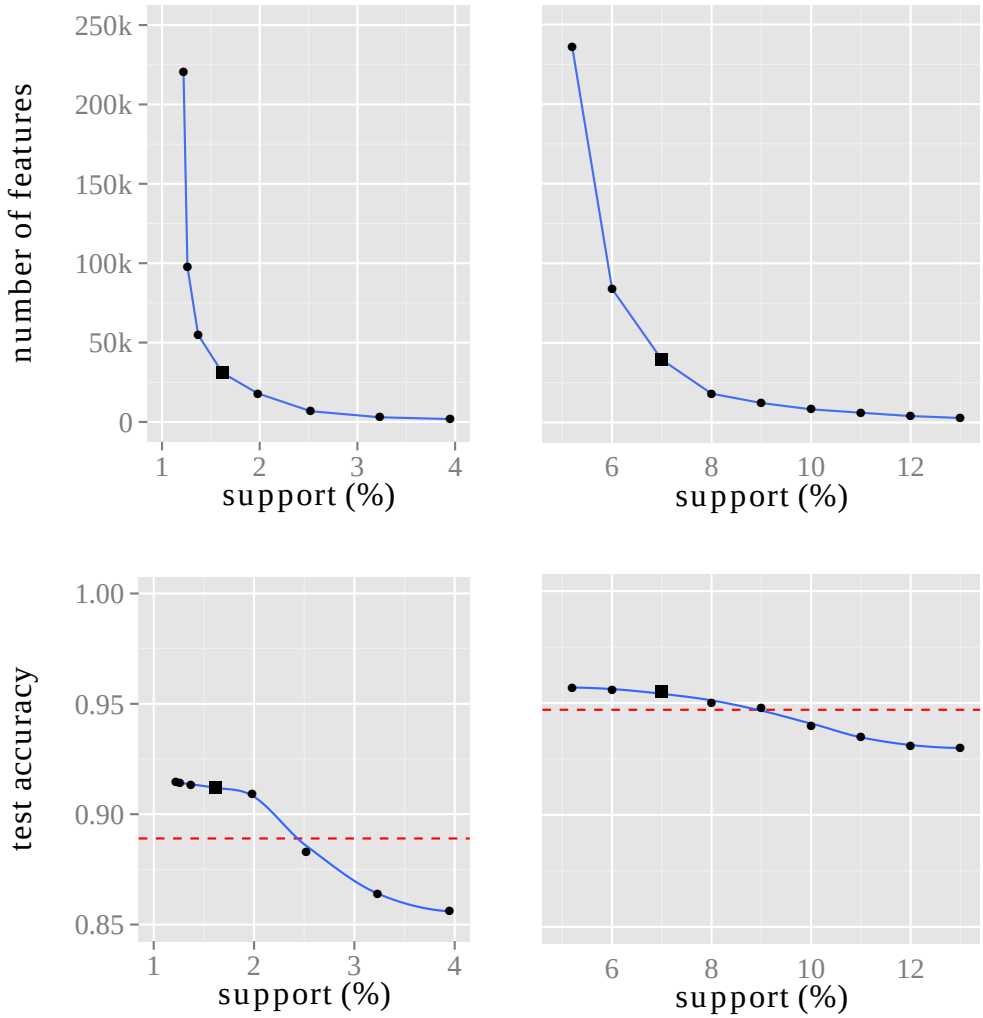
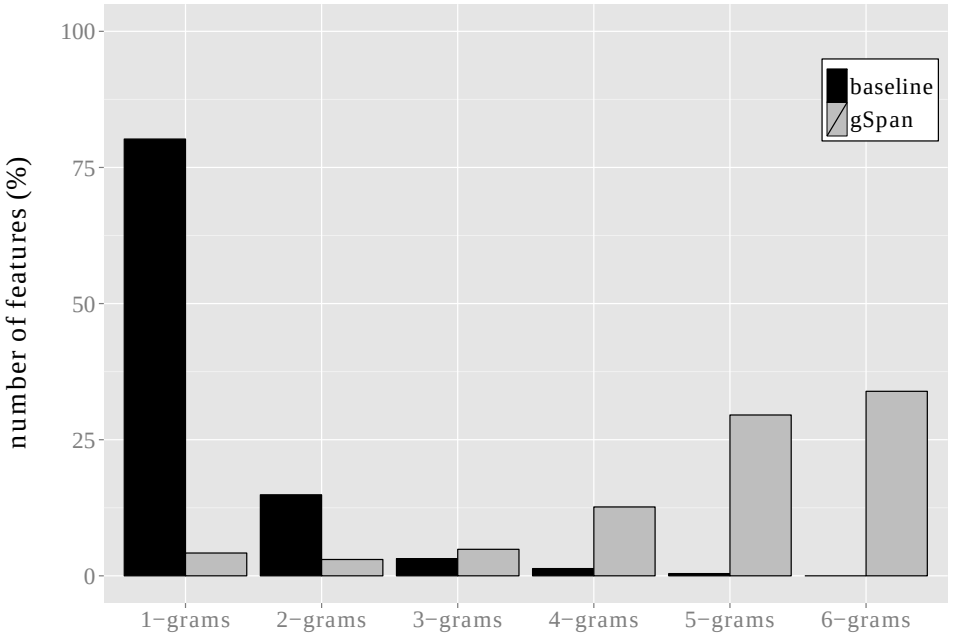
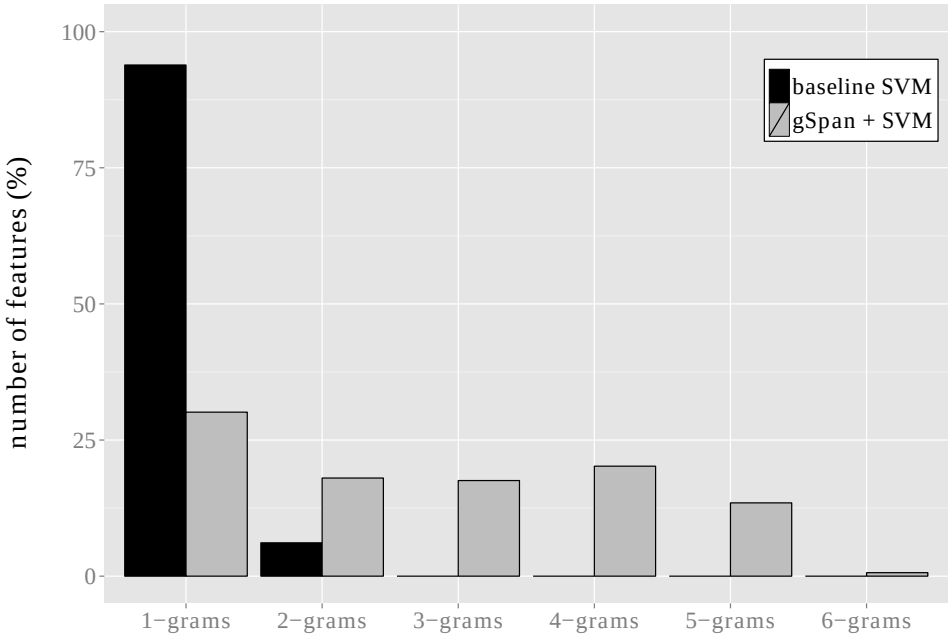


Figure 6.3 – Number of subgraph features/accuracy in test per support (%) on `WebKB` (left) and `R8` (right) datasets: black square correspond to the selected support value chosen via the elbow method and the dashed red line to the accuracy in test for the SVM baseline with n-grams.

instance. To obtain comparable results, we considered for the baseline n-grams with a minimum document frequency equal to the support. Otherwise, by definition, there are at least as many bigrams as there are unigrams and so forth.



(a) All features



(b) Top 5% most discriminative features

Figure 6.4 – Distribution of standard and long-distance n-grams among all features and among the top 5% most discriminative features for SVM on the `WebKB` dataset for various models.

Figure 6.4a shows that our approaches mine way more n-grams than unigrams compared to the baseline. This happens because with graph-of-words a subgraph of size n corresponds to a set of n terms while with bag-of-words a n-gram corresponds to a sequence of n terms. Figure 6.4b shows that the higher order n-grams still contribute indeed to the classification decision and in higher proportion than with the baseline. For instance, on the R8 dataset, {bank, base, rate} was a discriminative (top 5% SVM features) long-distance 3-gram for the category "interest" and occurred in documents in the form of "barclays bank cut its base lending rate", "midland bank matches its base rate" and "base rate of natwest bank dropped", pattern that would be hard to capture with traditional n-gram bag-of-words.

6.2.3 *Highlights, current limitations and future work*

We tackled the task of text categorization by representing documents as graph-of-words and then considering the problem as a graph classification one. We were able to extract more discriminative features that correspond to long-distance n-grams through frequent subgraph mining, controlling the support via an unsupervised method.

LIMITATIONS To the best of our knowledge, graph classification had never been tested at that scale – thousands of graphs and tens of thousands of unique node labels – and also in the multiclass scenario. For these reasons, we could not capitalize on all standard methods, in particular graph kernels. Moreover, mining all the original graphs for frequent subgraphs is an expensive process, which led us to the unsupervised feature selection technique presented in the next section.

FUTURE WORK We believe new kernels that support a very high number of unique node labels could yield even better performances. It is in this spirit that we recently started to explore with some colleagues shortest-path kernels between graph-of-words with promising results that led to a paper submission (Meladianos et al., 2015b).

6.3 COREWORD EXTRACTION FOR FEATURE SELECTION

Inspired by our coreword extraction technique previously introduced in Chapter 5, we thought of applying it on the graph-of-words representation to speed up the subgraph mining by only considering subgraphs appearing in the main cores. Even though we are dealing with unweighted graphs, the low precision in terms of keyword extraction reported in Section 5.2.3.2 was not a concern, especially with the high recall. Additionally, it turned out that this technique could also be used as an unsupervised feature selection step for standard n-gram features.

6.3.1 *Subgraphs of main-core-of-words as features*

SCALABILITY ISSUE Since the main drawback of mining frequent subgraphs for text categorization rather than chemical compound classification is the very high number of possible subgraphs because of the size of the graphs and the total number of graphs (more than 10x in both cases), we thought of ways to reduce the graphs' sizes while retaining as much classification information as possible.

MAIN CORE RETENTION The graph-of-words representation is designed to capture dependency between words, i.e. dependency between features in the context of ML but at the document-level. Initially, we wanted to capture recurring sets of words (i.e. take into account word inversion and subset matching) and not just sequences of words like with n-grams. In terms of subgraphs, this means words that co-occur with each other and form a dense subgraph as opposed to a path like for a n-gram. Therefore, when reducing the graphs, we need to keep their densest part(s) and that is why we considered extracting their main cores. Compared to other density-based algorithms, retaining the main core of a graph has the advantage of being linear in the number of edges, i.e. in the number of unique terms in a document in our case – the number of edges is at most the number of nodes times the fixed size of the sliding window, a small constant in practice (cf. Section 5.2.1).

DIMENSIONALITY REDUCTION In our second approach, denoted as "MC + gSpan + SVM", we repeat the same procedure as for "gSpan + SVM" from Section 6.2.2.2 except that we mine frequent subgraphs (gSpan) from the main core (MC) of each graph-of-words and then train a linear SVM on the resulting features. Main cores can vary from 1-core to 12-core depending on the graph structure, 5-core and 6-core being the most frequent (accounting for more than 60% of the main cores on all datasets). This yields results similar to the SVM baseline as shown on Table 6.3 for a faster mining and training compared to "gSpan + SVM" (6x faster on an Intel Core i5-3317U clocking at 2.6GHz and with 8GB of RAM). Table 6.4 shows the reduction in the dimension of the feature

Method \ Dataset	WebKB		R8	
	Accuracy	**F1-score**	**Accuracy**	**F1-score**
linear SVM	0.889	0.871	0.947	0.858
gSpan + SVM	**0.912***	**0.882**	**0.955***	**0.864**
MC + gSpan + SVM	0.901*	0.871	0.949*	0.858
MC + SVM	0.872	0.863	0.937	0.849

Method \ Dataset	LingSpam		Amazon	
	Accuracy	**F1-score**	**Accuracy**	**F1-score**
linear SVM	**0.991**	**0.973**	0.792	0.790
gSpan + SVM	**0.991**	0.972	**0.798***	**0.795**
MC + gSpan + SVM	0.990	**0.973**	**0.800***	**0.798**
MC + SVM	0.990	0.972	0.786	0.774

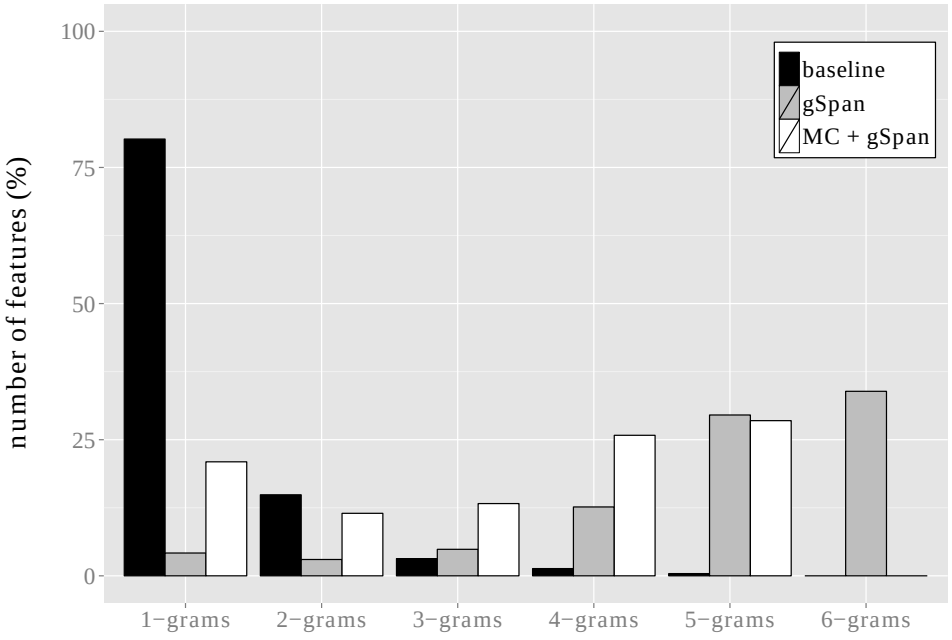
Table 6.3 – Effectiveness results (accuracy and macro-average F1-score in test) of long-distance n-grams over standard ones. Bold font marks the best performance in a column. * indicates statistical significance at $p < 0.05$ using the micro sign test against the SVM baseline of the same column.

Dataset	# subgraphs before	# subgraphs after	reduction
WebKB	30,868	10,113	67 %
R8	39,428	11,373	71 %
LingSpam	54,779	15,514	72 %
Amazon	16,415	8,745	47 %

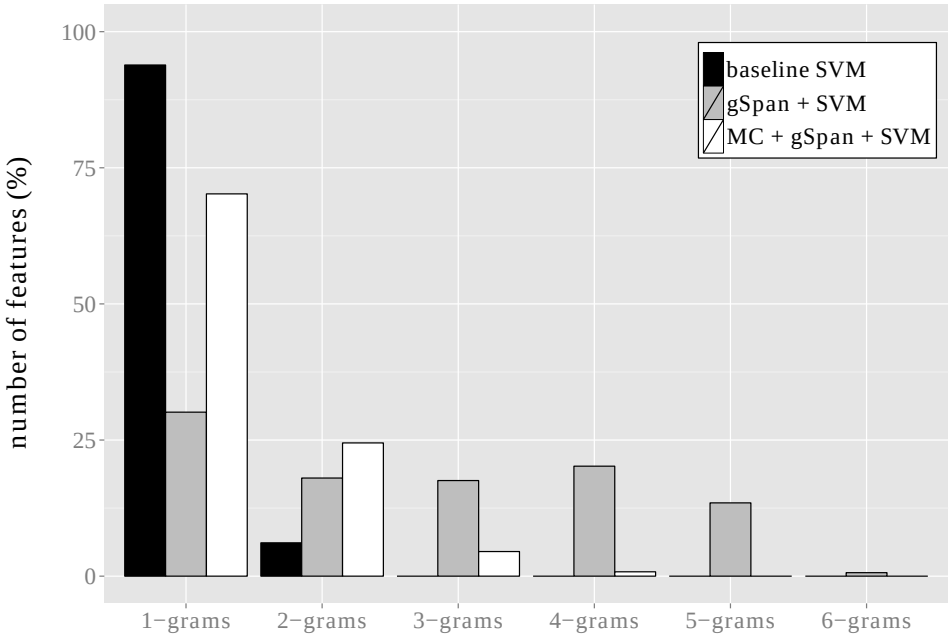
Table 6.4 – Total number of subgraph features vs. number of subgraph features present only in main cores along with the reduction of the dimension of the feature space.

space and we see that on average less than 60% of the subgraphs are kept for little to no cost in prediction effectiveness.

DISTRIBUTION OF MINED n-GRAMS Figure 6.5 illustrates the distribution of standard and long-distance n-grams among all features and among the top 5% most discriminative features for SVM on the `WebKB` dataset. We see that even when restricting to subgraph features from the main cores, there are still more n-grams of higher order with our proposed approach, which means the feature selection step retains the classification information.



(a) All features



(b) Top 5% most discriminative features

Figure 6.5 – Distribution of standard and long-distance n-grams among all features and among the top 5% most discriminative features for SVM on the `WebKB` dataset for various models.

Dataset	# n-grams before	# n-grams after	reduction
WebKB	1,849,848	735,447	60 %
R8	1,604,280	788,465	51 %
LingSpam	2,733,043	1,016,061	63 %
Amazon	583,457	376,664	35 %

Table 6.5 – Total number of n-gram features vs. number of n-gram features present only in main cores along with the reduction of the dimension of the feature space on all four datasets.

6.3.2 *Unsupervised n-gram feature selection*

Additionally, we can capitalize on the coreword extraction technique to still mine binary n-gram features for classification but considering only the terms belonging to the main core of each document. In that setting, the graph-of-words representation is only used as a pre-processing step to filter out unwanted terms.

DIMENSIONALITY REDUCTION Compared to most existing feature selection techniques in the field (Y. Yang and Pedersen, 1997), it is *unsupervised* and *corpus-independent* as it does not rely on any labeled data like Information Gain (IG), Mutual Information (MI) or χ^2 nor any collection-wide statistics like IDF, which can be of interest for large-scale TC in order to process documents in parallel, independently of each other. In some sense, it is similar to what Özgür et al. (2005) proposed with *corpus-based* and *class-based* keyword selection for TC except that we use here *document-based* keyword selection like Hulth and Megyesi (2006) with its supervised keyword extraction technique from Hulth (2003).

NUMBER OF NON-ZERO FEATURE VALUES Additionally, since a document is only represented by a subset of its original terms, the number of non-zero feature values per document also decreases, which matters for SVM, even for the linear kernel, when considering the dual formulation or in the primal with more recent optimization techniques (Joachims, 2006). Figure 6.6 shows the distribution of non-zero features before and after the feature selection on the R8 dataset. Similar changes in distribution can be observed on the other datasets, from a right-tail Gaussian to a power law distribution.

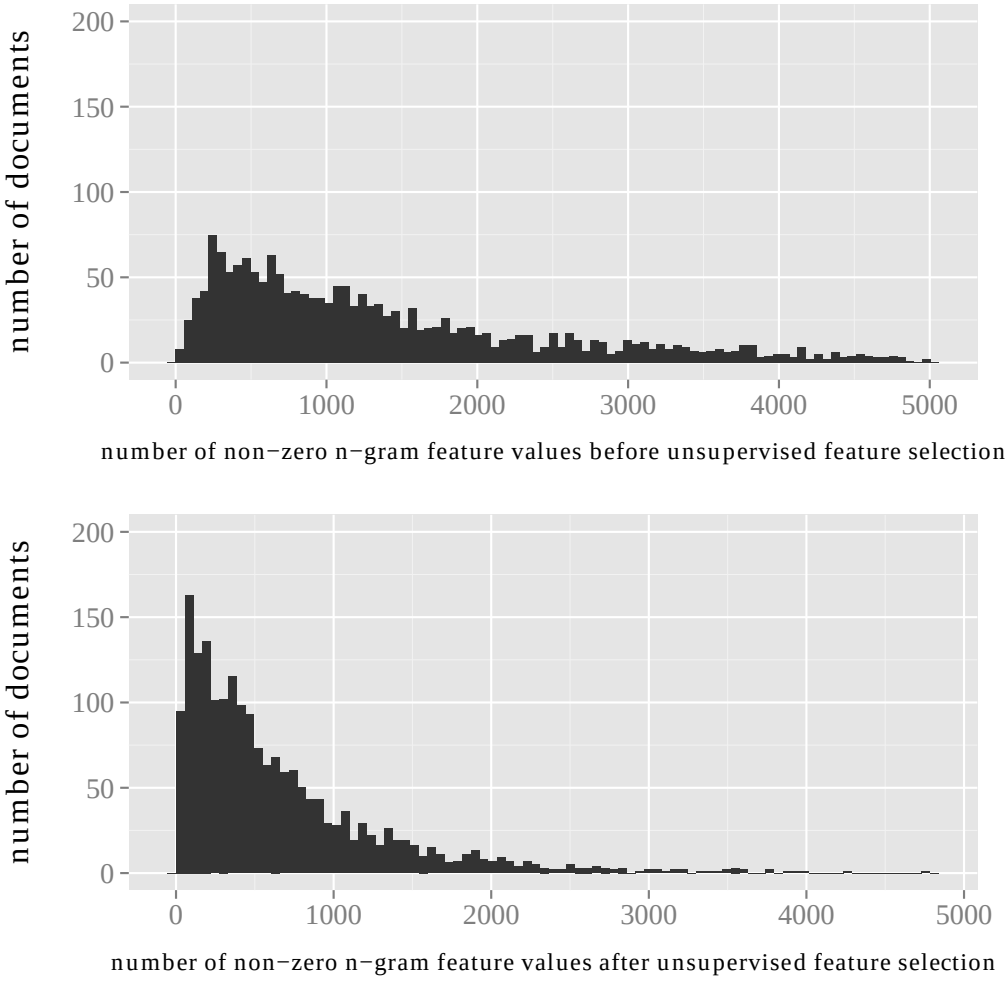


Figure 6.6 – Distribution of non-zero *n*-gram feature values before and after unsupervised feature selection (main core retention) on R8 dataset.

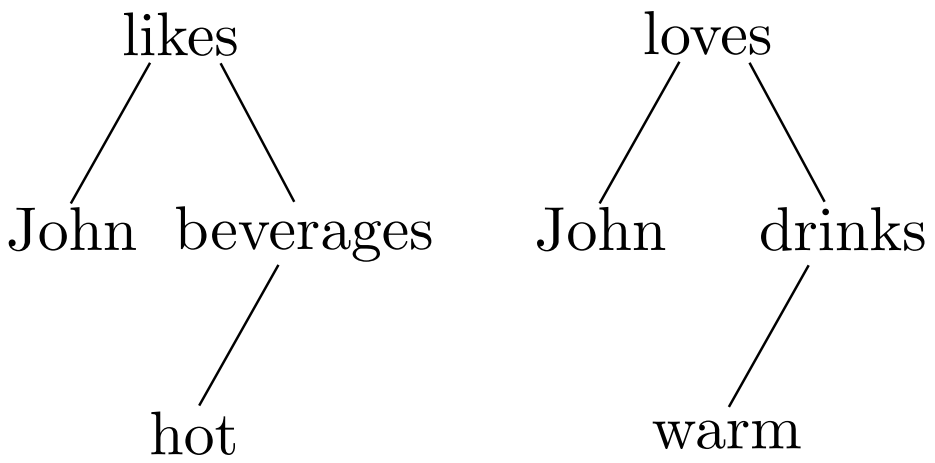
6.4 CONVOLUTIONAL SENTENCE KERNEL BASED ON DISTANCES IN WORD EMBEDDINGS FOR SHORT TEXT CATEGORIZATION

In this section, we tackle the **sparsity issue** in the n-gram feature space that arises when classifying short documents or when little training data is available by proposing a convolutional sentence kernel based on distances in word embeddings. Because of the absence of term repetition in sentences in general, our usual *statistical* graph-of-words representation provides little to no added value compared to the standard n-gram bag-of-words, even when considering subgraph features. Therefore, in this specific scenario, we rather experimented with *syntactic* graph-of-words and more specifically NLP dependency trees as reported in our work (J. Kim et al., 2015).

As we have previously seen, in Text Categorization (TC), words (unigrams) and phrases (n-grams) have been traditionally considered as document features and subsequently fed to a classifier such as an SVM. In the SVM dual formulation (cf. Section 6.1.2.5) that relies on kernels, i. e. similarity measures between documents, a linear kernel can be interpreted as the number of exact matching n-grams between two documents (since it is a dot product in the n-gram vector space). For example, consider the two following sentences: 'John likes Mary' and 'Mary likes John'. Then the kernel value between the two sentences is 3 for unigram features, same as the self-similarity (kernel value between the first sentence and itself), while for bigram features it goes up to 3+2 for the self-similarity, which is the main reason why we consider higher order n-grams in the first place: to take into account some local word dependency to counterbalance the term independence assumption behind the bag-of-words document representation.

To consider more syntactically meaningful phrases, the NLP community has also considered as n-grams downward paths in dependency trees rather than sequences in the original text. Figure 6.7a illustrates the dependency tree representation of the sentence 'John likes hot beverages'. Moreover, this representation also has the desirable effect of being more robust to the sparsity issue we are interested in. Indeed, word synonymy can counteract the potential of n-grams, e. g., considering 'John likes hot beverages' and 'John likes warm beverages', the kernel value between the two sentences is only one more with bigram features than with unigram features when using the original text because of the use of different, yet semantically close, adjectives. However, if we were to use the dependency tree representation, the kernel value would be 3+2 for bigrams. Nevertheless, it is still problematic when inner nodes are different as illustrated in Figure 6.7 with two sentences semantically close yet with low linear kernel value because of word synonymy, especially at the root level.

We thus propose to **relax the exact matching between words** by capitalizing on distances in a word embedding space. More specifically, we smooth the implicit delta word kernel behind the linear sentence kernel to capture the similarity between words that are different, yet semantically similar. Moreover, we also take advantage of the dependency tree structure to capture the syntactic



(a) 'John likes hot beverages' (b) 'John loves warm drinks'

Figure 6.7 – Illustration of two dependency tree representations of semantically close sentences yet with low linear kernel value because of word synonymy.

information hidden in the sentence and to consider more meaningful phrases than the ones extracted from the sequence of words (or set of words in a statistical graph-of-words). We then aggregate these word and phrase kernels into sentence and documents kernels through **convolution**. In the case of Figure 6.7, while the baseline approach would find a kernel value between the two sentences of 1 with bigram features, our method would find a value close to 4+3!

6.4.1 *Model definition*

We define here our proposed word, phrase, sentence and document kernels from distances in a word embedding space.

6.4.1.1 *Related work*

Siolas and d'Alché-Buc (2000) pioneered the idea of **semantic kernels** for text categorization, capitalizing on WordNet (Miller, 1995) to propose continuous word kernels based on the inverse of the path lengths in the tree rather than the common delta word kernel used so far, i. e. exact matching between unigrams. Bloehdorn, Basili, et al. (2006) extended it later to other tree-based similarity measures from WordNet while Mavroeidis et al. (2005) exploited its hierarchical structure to define a Generalized Vector Space Model kernel.

In parallel, Collins and Duffy (2001) developed the first *tree kernels* to compare trees based on their topology (e. g., shared subtrees) rather than the similarity between their nodes. Culotta and Sorensen (2004) used them as Dependency Tree Kernel (DTK) to capture *syntactic similarities* while Bloehdorn and Moschitti (2007) and Croce et al. (2011) used them on parse trees with respectively Semantic Syntactic Tree Kernel (SSTK) and Smoothing Partial Tree Kernel (SPTK), adding

node similarity based on WordNet to capture *semantic similarities* but limiting to comparisons between words of same POS tag.

Similarly, Gärtner et al. (2003b) developed *graph kernels* based on random walks and Srivastava et al. (2013) used them on dependency trees with Vector Tree Kernel (VTK), adding node similarity based on word embeddings from SENNA (Collobert et al., 2011) and reporting improvements over SSTK. The change from WordNet to SENNA was supported by the recent progress in word embeddings that are better suited for computing distances between words. Word embeddings have achieved significant success at representing words in a **low-dimension dense Euclidean space** emulating human semantic cognition as opposed to the naive sparse one-of approach (with a space of dimension the size of the vocabulary). They have proven to be practical for a wide variety of tasks such as POS-tagging and chunking (Collobert et al., 2011), paraphrase detection (Blacoe and Lapata, 2012) and sentiment analysis (Maas et al., 2011). Actually, in our experiments, **word2vec** by Mikolov, Chen, et al. (2013) led to better results than with SENNA for both VTK and our kernels. Moreover, it possesses an additional additive compositionality property obtained from the Skip-gram training setting (Mikolov, Sutskever, et al., 2013), e. g., the closest word to 'Germany' + 'capital' in the vector space is found to be 'Berlin'. Throughout the rest of this section, we assume word2vec as the word embedding – we initially started exploring with SENNA (Collobert et al., 2011) but with less success. We represent the embedding of a word w as $\mathbf{w}$, i. e. a 300-dimension dense vector.

More recently, for short text similarity, Song and Roth (2015) and Kenter and de Rijke (2015) proposed additional semantic meta-features based on word embeddings to enhance classification.

6.4.1.2 *Word Kernel (WK)*

We define a kernel between two words as a **polynomial kernel** over a cosine similarity in the word embedding space:

$$\mathrm{WK}(w_1, w_2) = \left[\frac{1}{2}\left(1 + \frac{\langle \mathbf{w}_1, \mathbf{w}_2 \rangle}{\|\mathbf{w}_1\| \|\mathbf{w}_2\|}\right)\right]^{\alpha} \tag{6.27}$$

where α is a scaling factor. We have also tried Gaussian, Laplacian and sigmoid kernels but they led to poorer results in our experiments. Note that a *delta* word kernel, i. e. the Dirac function $\mathbb{1}_{w_1=w_2}$, leads to a document kernel corresponding to the standard linear kernel over n-grams.

6.4.1.3 *Phrase Kernel (PhK)*

Next, we define a kernel between phrases consisting of several words. We considered two types of phrases: (1) **statistical phrases** defined as contiguous sequences of words in the original sentence; and (2) **syntactic phrases** defined as downward paths in the dependency tree representation. With this dependency tree involved, we expect to have phrases that are syntactically more meaningful.

Note that the Vector Tree Kernel (VTK) from Srivastava et al. (2013) considers *random walks* in dependency trees instead of downward paths, i. e. potentially taking into account same nodes multiple times for phrase length greater than two, phenomenon known as *tottering*.

Once we have phrases to compare, we may construct a kernel between them as the product of word kernels if they are of the same length l. That is, we define the Product Kernel (PK) as:

$$\mathrm{PK}(p_1, p_2) = \prod_{i=1}^{l} \mathrm{WK}(w_i^1, w_i^2) \tag{6.28}$$

where w_i^j is the i^{th} word in phrase p_j of length l.

Alternatively, in particular for phrases of different lengths, we may embed phrases into the embedding space by taking a composition operation on the constituent word embeddings. We considered two common forms of composition (Blacoe and Lapata, 2012): vector addition ($+$) and element-wise multiplication ($\odot$). Then we define the Composition Kernel (CK) between phrases as:

$$\mathrm{CK}(p_1, p_2) = \mathrm{WK}(\mathbf{p}_1, \mathbf{p}_2) \tag{6.29}$$

where $\mathbf{p}_j$, the embedding of the phrase p_j, can be obtained either by addition ($\mathbf{p}_j = \sum_{i=1}^{l} \mathbf{w}_i^j$) or by element-wise multiplication ($\mathbf{p}_j = \bigodot_{i=1}^{l} \mathbf{w}_i^j$) of its word embeddings. For CK, we do not require the two phrases to be of the same length so the kernel has a desirable property of being able to compare 'Berlin' with 'capital of Germany' for instance.

6.4.1.4 *Sentence Kernel (SK)*

We can then formulate a sentence kernel in a similar way to Zelenko et al. (2003). It is defined through convolution as the sum of all local phrasal similarities, i. e. kernel values between phrases contained in the sentences:

$$\mathrm{SK}(s_1, s_2) = \sum_{\substack{p_1 \in \phi(s_1) \\ p_2 \in \phi(s_2)}} {\lambda_1}^{\epsilon} \, {\lambda_2}^{\eta} \, \mathrm{PhK}(p_1, p_2) \tag{6.30}$$

where $\phi(s_k)$ is the set of either statistical or syntactic phrases (or set of random walks for VTK) in sentence s_k, λ_1 is a decaying factor penalizing longer phrases, $\epsilon = \max\{|p_1|, |p_2|\}$ is the maximum length of the two phrases, λ_2 is a *distortion* parameter controlling the length difference η between the two phrases ($\eta = ||p_1| - |p_2||$) inspired by the IBM model from statistical machine translation (Brown, Della Pietra, et al., 1993) and PhK is a phrase kernel, either PK, CK^{+} or $\mathrm{CK}^{\odot}$. Since the composition methods we consider are associative, we employ the dynamic programming approach from Zelenko et al. (2003) to avoid duplicate computations. If we limit the maximum length of phrases to L, the computation of one kernel value between sentences of n words can be performed in $\mathcal{O}(n^2 L^2 d)$ time.

6.4.1.5 *Document kernel*

Finally, we sum sentence kernel values for all pairs of sentences between two documents to get the document kernel. Once we have obtained all document kernel values K_{ij} between documents i and j, we may normalize them by $\sqrt{K_{ii}K_{jj}}$ as the length of input documents might not be uniform.

6.4.2 *Experiments*

In this section, we present the experiments we carried out to validate our proposed sentence kernel. We first describe the datasets, the experimental settings, the evaluation metrics and the considered models. We then report the results we obtained and discuss their interpretations.

6.4.2.1 *Datasets, experimental settings, evaluation metrics and models*

DATASETS We considered four tasks: (1) binary **sentiment analysis** with a movie review dataset of 10,662 sentences (`PL05`) (Pang and Lilian Lee, 2005) and a product review dataset (`Amazon`) of 2,000 multi-line documents for 4 different product groups (Blitzer et al., 2007); (2) ternary **sentiment analysis** with the SemEval 2013 Task B dataset (`Twitter`) containing 12,348 tweets classified as positive, neutral, or negative (Nakov et al., 2013); (3) binary **subjectivity detection** with a dataset of 10,000 sentences (`PL04`) (Pang and Lilian Lee, 2004) and another of 11,640 sentences (`MPQA`) (Wiebe et al., 2005); and (4) seven-class **topic spotting** with a news dataset (`News`) with 32,602 one-line news summaries (Vitale et al., 2012).

EXPERIMENTAL SETTINGS We used the FANSE parser (Tratz and Hovy, 2011) to generate dependency trees and the pre-trained version of word2vec [1], a 300 dimensional representation of 3 million English words trained over a Google News dataset of 100 billion words using the Skip-gram model and a context size of 5. While fine-tuning the embeddings to a specific task or on a given dataset may improve the result for that particular task or dataset (Levy, Goldberg, and Dagan, 2015), it makes the expected results less generalizable and the method harder to use as an off-the-shelf solution – re-training the neural network to obtain task-specific embeddings requires a certain amount of training data, admittedly unlabeled, but still not optimal under our scenario with short documents and little task-specific training data available. Moreover, tuning the hyperparameters to maximize the classification accuracy needs to be carried out on a validation set and therefore requires additional labeled data. Here, we are more interested in showing that distances in a given word vector space can enhance classification in general. As for the dependency-based word embeddings proposed by Levy and Goldberg (2014), we do not think they are better suited for the problem we are tackling. As we will see in the results, we

1. https://code.google.com/p/word2vec#Pre-trained_word_and_phrase_vectors

do benefit from the dependency tree structure in the phrase kernel but we still want the word kernel to be based on topical similarity rather than functional similarity.

To train and test the SVM classifier, we used the LibSVM library (Chang and Lin, 2011) and employed the one-vs-one strategy for multi-class tasks. To prevent overfitting, we tuned the parameters using cross-validation on 80% of PL05 dataset ($\alpha = 5$, $\lambda_1 = 1$ for PK since there is no need for distortion as the phrases are of same length by definition, and $\lambda_1 = \lambda_2 = 0.5$ for CK) and used the same set of parameters on the remaining datasets. We performed normalization for our kernel and baselines only when it led to performance improvements on the training set (PL05, News, PL04 and MPQA).

EVALUATION METRICS To evaluate the performance of our proposed approaches over standard baselines, we computed on the test set both the micro- and macro-average F1-score (cf. Section 2.3.3.2). Because we are dealing with single-label classification, the micro-average F1-score corresponds to the accuracy and is a measure of the overall prediction effectiveness (cf. Section 2.3.3.2). Conversely, the macro-average F1-score takes into account the skewed class label distributions by weighting each class uniformly. In the next subsection, we will report accuracy on the remaining 20% for PL05, on the standard test split for Twitter (25%) and News (50%) and from 5-fold cross-validation for the other datasets (Amazon, PL04 and MPQA). We only report accuracy for space constraints as the macro-average F1-scores led to similar conclusions (and except for Twitter and News, the class label distributions are balanced). Results for phrase lengths longer than 2 were omitted since they were marginally different at best. Statistical significance of improvement over the bigram baseline with the same phrase definition was assessed using the micro sign test considering one-sided p-values less than 0.01 to reject the null hypothesis (cf. Section 2.3.4). For the Amazon dataset, we report the average of each metric over the four sub-collections.

MODELS We considered (1) 3 phrase definition: statistical, syntactic and random walk; (2) 3 phrase kernels: PK, CK^{+} or $CK^{\odot}$; (3) 2 word kernels: delta and polynomial; as well as (4) 2 phrase lengths: unigram and bigram. The typical unigram and bigram baseline approaches correspond to the models with a delta word kernel, i.e. exact matching between all the words of a phrase. The VTK baseline capitalizes on a random walk phrase definition, i.e. a different function $\phi(\cdot)$ that enumerates all random walks in the dependency tree representation following Gärtner et al. (2003a) whereas we only consider the downward paths. To ensure a fair comparison, we used a polynomial word kernel and word2vec for VTK as they led to better results than the originally proposed sigmoid kernel and SENNA.

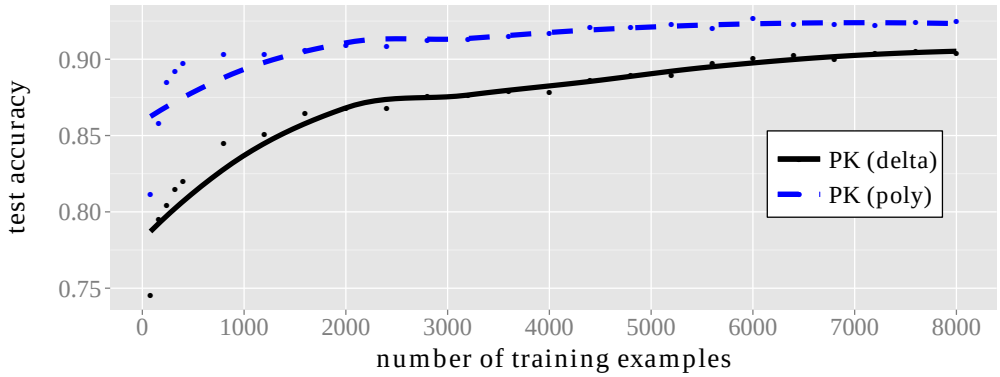


Figure 6.8 – Test accuracy vs. number of training examples for our polynomial word kernel vs. a delta word kernel (baseline) on the PL04 dataset, both with bigram syntactic phrases.

6.4.2.2 *Classification results*

Table 6.6 presents the results from our convolutional sentence kernel and the baseline approaches. The 3rd row corresponds to DTK (Culotta and Sorensen, 2004) and the 4th one to VTK (Srivastava et al., 2013), differing with our model on the 9th row only by the phrase definition. It shows good performance across all datasets but its computation was more than 700% slower than with our kernel.

Overall, we obtained better results than the n-gram baselines, DTK and VTK, especially with syntactic phrases. With regard to phrase kernels, PK generally produced better results than CK, implying that the semantic linearity and ontological relation encoded in the embedding is not sufficient enough and treating them separately is more beneficial. However, we believe CK has more room for improvement with the use of more accurate phrase embeddings.

There was little contribution to the accuracy from non-unigram features, indicating that large part of the performance improvement is credited to the word embedding resolving the sparsity issue. This can be well observed with the following experiment on the number of training examples. Figure 6.8 shows the accuracy on the same test set (20% of the dataset) when the learning was done on 1% to 100% of the training set (80% of the dataset) for our polynomial word kernel (dashed curve) vs. a delta word kernel (plain curve) on the PL04 dataset, both with bigram syntactic phrases. We see that our kernel starts to plateau earlier in the learning curve than the baseline and also reaches the maximum baseline accuracy with only about 1,500 training examples.

6.4.2.3 *Computational complexity*

Solving the SVM in the primal for the baselines requires $\mathcal{O}(NnL)$ time where N is the number of training documents, n is the number of words in the doc-

phrase definition	phrase kernel	phrase length	word kernel	PL05	Amazon	Twitter
statistical	PK	1	delta	0.742	0.768	0.623
statistical	PK	2	delta	0.739	0.765	0.611
syntactic	PK	2	delta	0.748	0.791	0.646
random walk	PK	2	poly	0.799	0.810	0.698
statistical	PK	1	poly	0.789*	0.797	0.776*
statistical	PK	2	poly	0.784*	0.798	0.762*
statistical	CK$^+$	2	poly	0.796*	0.778	0.613
statistical	CK$^\odot$	2	poly	**0.801***	0.783	0.757*
syntactic	PK	2	poly	0.796*	**0.813***	**0.808***
syntactic	CK$^+$	2	poly	0.794*	0.780	0.741*
syntactic	CK$^\odot$	2	poly	0.797*	0.774	0.744*

phrase definition	phrase kernel	phrase length	word kernel	News	PL04	MPQA
statistical	PK	1	delta	0.769	0.904	0.754
statistical	PK	2	delta	0.766	0.907	0.754
syntactic	PK	2	delta	0.767	0.910	0.757
random walk	PK	2	poly	0.802	**0.927**	**0.797**
statistical	PK	1	poly	**0.806***	0.923*	0.793*
statistical	PK	2	poly	0.801*	0.926*	0.794*
statistical	CK$^+$	2	poly	0.792*	0.917*	0.796*
statistical	CK$^\odot$	2	poly	0.793*	0.918*	0.794*
syntactic	PK	2	poly	0.805*	**0.927***	0.796*
syntactic	CK$^+$	2	poly	0.788*	0.918*	0.794*
syntactic	CK$^\odot$	2	poly	0.792*	0.918*	0.794*

Table 6.6 – Accuracy results on the test set for PL05 (20%), standard test split for Twitter (25%) and News (50%) and from 5-fold CV for the other datasets (Amazon, PL04 and MPQA). Bold font marks the best performance in a column. * indicates statistical significance at $p < 0.01$ using the micro sign test against the bigram baseline (delta word kernel) with the same phrase definition of the same column.

ument and L is the maximum phrase length considered. The computation of VTK reduces down to power series computation of the adjacency matrix of the product graph, and since we require kernel values between all documents, it requires $\mathcal{O}(N^2(n^2d + n^4L))$ time where d is the dimension of the word embedding space (300 for the word2vec version we used).

Our kernel is the sum of phrase kernels starting from every pair of nodes between two sentences, for all phrasal lengths and distortions under the consideration. By storing intermediate values of composite vectors, phrase kernel can be computed in $\mathcal{O}(d)$ time regardless of the phrase length, therefore the whole computation process has $\mathcal{O}(N^2n^2L^2d)$ complexity. Although our kernel has the squared terms of the baseline's complexity, we are tackling the sparsity issue that arise with short text (small n) or when little training data is available (small N). Moreover, we were able to get better results with only bigrams (small L). Hence, the loss in efficiency is acceptable considering significant gains in effectiveness.

6.4.3 *Highlights, current limitations and future work*

We proposed a novel convolutional sentence kernel based on word embeddings that overcomes the sparsity issue, which arises when classifying short documents or when little training data is available. We described a general framework that can encompass the standard n-gram baseline approach as well as more relaxed versions with smoother word and phrase kernels. Because we targeted short documents, we could not capitalize on the statistical graph-of-words we had been using so far but we switched to syntactic graph-of-words to still capture some additional and meaningful word dependency.

LIMITATIONS Our main regret in this work was that CK did not perform better than PK, especially when increasing the maximum phrase length. Having a phrase kernel than can compare phrases of different length, with an appropriate distortion parameter, seems intuitively well-suited for overcoming the sparsity issue. We think better phrase embeddings such as the one recently proposed by Le and Mikolov (2014), Yin and Schütze (2014), and Yu and Dredze (2015) would greatly help.

FUTURE WORK A direct follow-up might involve designing a new kernel that puts more emphasis on the syntactic structure, e. g., having word kernels taking into account POS tags, or devising a new convolution kernel with word embeddings on a syntactic parse tree with appropriate similarity measures between non-terminal nodes.

7 CONCLUSIONS

In this chapter, we conclude the dissertation by first highlighting the main findings uncovered during our research and reported in great details in the previous chapters centered around our alternative document representation, namely graph-of-words. We then mention the potential follow-ups of our work as well as research directions we deem promising and to the best of our knowledge uncharted at the time of the writing.

7.1 SUMMARY

The Ph.D. was the occasion to get accustomed to a certain level to the fields of IR, ML, NLP and more generally TM. And while acquiring a better understanding of the state-of-the-art methods and baseline approaches, we were able to either propose a unifying mathematical framework encompassing both the TF-IDF and BM25 scoring functions or explore novel applications of the graph-of-words representation in terms of term weighting normalizations, community structure or more relaxed classification features.

COMPOSITION OF TF NORMALIZATIONS It was already known that the seminal TF-IDF and BM25 scoring functions satisfy the same set of heuristic retrieval constraints and we went further by explaining their sole difference in terms of concave transformation and the order of composition between all the term frequency normalizations. More generally, by explicitly proposing the function composition mathematical operator to combine the various transformations to apply, we think it makes it easier for researchers to specify which normalizations and in which order they used them when reporting results as the TF$\times$IDF denomination is way too vague, encompassing models ranging from raw frequencies to fully normalized ones.

TW-IDF When assessing a query term's weight in a document, rather than considering the overall term frequency of a word and then applying a concave transformation to ensure a decreasing marginal gain in relevance, one should instead consider for each word the number of distinct contexts of co-occurrence with other words so as to favor terms that appear with a lot of other terms, i. e. consider the node degree in the corresponding unweighted graph-of-words. In particular, for ad hoc IR, we challenged TF-IDF and BM25 with a novel retrieval model called TW-IDF where we replaced the normalized term frequency with a graph-based term weight. The proposed scoring function appeared to require (1) no concave transformation because an additional edge is added to the graph-

of-words only if the context of occurrence for the term is new, which holds the same amount of information whatever the term weight value already is; (2) no tuning for the pivoted document length penalization; as well as (3) no lower-bounding regularization.

COREWORDS When assigning keywords to full papers or even abstracts, researchers tend to select as keywords not only central but also densely connected nodes in the corresponding weighted graph-of-words, property captured in our experiments by reducing each graph to its main core using the concept of graph degeneracy. Rather than relying on more complex centrality measures, we found out that it was more effective to capitalize on the community structure of the network to find cohesive sets of nodes – interpreted as corewords of the document – and more efficient to utilize the concept of k-core as opposed to more complex community detection techniques.

LONG-DISTANCE n-GRAMS From our experiments in text categorization, subgraphs of unweighted graph-of-words – interpreted as long-distance n-grams – appear to be more discriminative features than standard n-grams, partly because they can capture more variants of the same set of terms compared to fixed sequences of terms and therefore appear in more documents. Not only for the same support value, the proportion of higher order n-grams was superior when considering graph-of-words over the standard bag-of-words but that remained true in the top 5% most discriminative features for the SVM with subgraphs of size 4 and 5, even when reducing beforehand the networks to their main cores as an unsupervised feature selection step.

CONVOLUTIONAL SENTENCE KERNEL When classifying short documents or when little training data is available, sparsity issues arise because of word synonymy for instance. Therefore, we proposed to relax the exact matching between words by capitalizing on distances in word embeddings. More specifically, we smooth the implicit delta word kernel behind the traditional n-gram linear kernel to capture the similarity between words that are different, yet semantically similar. Moreover, we also took advantage of syntactic graph-of-words to capture the dependency structure hidden in the sentence and to consider more meaningful phrases than the ones extracted from the sequences of words in bag-of-words or sets of words in statistical graph-of-words. We then aggregated these word and phrase kernels into sentence and documents kernels through convolution.

7.2 FUTURE WORK

In this section, we briefly present the additional research ideas that came to mind while conducting the research presented in this dissertation. Some of them are direct follow-ups of our published works while others are more general and might be worth another Ph.D.! They were left unexplored mostly because of time but also sometimes because of the lack of actual applications to evaluate on.

TW-IDW As discussed in Section 4.2.3.2, we could challenge the document independence assumption usually made in TM and consider a collection of documents as a graph-of-documents instead of a bag-of-documents in order for instance to compute an alternative to IDF. Following one of our previous works (Rousseau and Vazirgiannis, 2013b), we could then envisage exploring term weights based on TW×IDW rather than TF×IDF. This assumes in particular that we have a graph-of-documents, even for Web-scale datasets, which is a research issue in itself, especially if the document similarity capitalizes on the graph-of-words representation, e. g., using the graph kernel we proposed in Meladianos et al. (2015b).

CGOW FROM HOP-1 NEIGHBORHOOD As briefly mentioned in Section 3.1.1 and Section 6.4.1.1, word2vec is a recent word embedding space that has been proposed by Mikolov, Chen, et al. (2013) and that is learnt using the Continuous Bag-Of-Words (CBOW) model over phrases. We think learning word embeddings over a set of hop-1 neighborhoods using the corresponding Continuous Graph-Of-Words (CGOW) model would yield a model that can better take into account word inversion and subset matching for instance, similarly to what we have observed in Chapter 6 for text categorization.

GRAPH-OF-WORDS EMBEDDINGS From word embeddings, researchers have been deriving document embeddings, e. g., as a dimensionality reduction step before classification, by considering the document as the centroid of its words following the bag-of-words representation. We think it would be interesting to derive a document embedding from its graph-of-words representation, e. g., as a weighted centroid with weights based on the core number of each word.

BETTER IDF FROM WORD EMBEDDINGS As described in Section 6.4, we tackled the sparsity issue in short document categorization by capitalizing on words embeddings to define a polynomial word kernel and basically take into account word synonymy in the term frequency weights (J. Kim et al., 2015). We think IDF could also benefit from word similarity since its estimation proves difficult when processing short documents or when little training data is available because it is harder to distinguish between a true rare word and a rare word in the small collection with a lot of synonymous.

GRAPH KERNELS WITH CONTINUOUS NODE SIMILARITY As introduced in Section 6.2.1.1, most of the existing graph kernels developed so far put the emphasis on the topology of the network and considered delta kernels for node similarity, including our proposed shortest-path graph-of-words kernel (Meladianos et al., 2015b). The main reason is probably because continuous node similarities made less sense in the case of chemical compounds, the data of interest in most existing works. In the case of text, vertices correspond to words and we could capitalize on word embeddings like we did with dependency trees (J. Kim et al., 2015) to take into account node similarity on top of edge similarity, the main challenge being in terms of efficiency as we would potentially need to compare every path of one graph to every path of the other one.

COMMUNITY-BASED GRAPH KERNEL As reviewed in Section 6.2.1.1, proposed graph kernels are based on random walks, shortest paths, cycles or subtrees. We believed it would be interesting to explore kernels based on graph properties at a higher level and in particular based on communities. For text, our initial experiments from coreword extraction showed single-component main cores and not with multiple components that would correspond to the various topics mentioned in the document as expected. Therefore, we lack a collection of graphs where a similarity measure based on the community structure would be relevant.

FUNCTIONAL ML: LEARNING TO COMPOSE Our earliest published work revolved around the functional composition of normalizations and its optimal order to get the best scoring function for ad hoc IR (Rousseau and Vazirgiannis, 2013a). In practice, researchers have only proposed three normalizations so far to satisfy a set of several heuristic retrieval constraints and therefore all the combinations can be explored in a brute force manner. As briefly mentioned in Section 4.1.4, with more transformations, we believe the machine could be learning to compose in a functional ML scenario. Actually, at another level, we could think of learning how to best compose several dimensionality reduction steps for instance – in some sense, what happens in a multi-layer neural network where each layer corresponds to a transformation applied on the input data.

7.3 EPILOGUE

As mentioned in introduction, we firmly believe in the potential of machines, the limits of humans and thus the need for intelligent machines, especially when it comes to understand text, our preferred mean of information storage and knowledge transfer. Throughout this dissertation, we have presented our understanding of the research field associated to that quest and hopefully reported findings that will help its comprehension, if not progress. A lot has been accomplished by the community already but we are not done yet...

BIBLIOGRAPHY

[1] W. D. Abilhoa and L. N. de Castro. A keyword extraction method from twitter messages represented as graphs. *Applied mathematics and computation*, 240:308–325, 2014 (cited on pages 31, 34, 65).

[2] C. C. Aggarwal and C. Zhai. A survey of text classification algorithms. *Mining text data*, pages 163–222. Springer New York, 2012 (cited on page 12).

[3] A. Ahmed, V. Batagelj, X. Fu, S.-H. Hong, D. Merrick, and A. Mrvar. Visualisation and analysis of the internet movie database. In *Proceedings of the 6th international asia-pacific symposium on visualization*. In APVIS '07. IEEE Computer Society, 2007, pages 17–24 (cited on page 68).

[4] M. A. Aizerman, E. A. Braverman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and remote control*, 25:821–837, 1964 (cited on page 96).

[5] J. I. Alvarez-Hamelin, L. Dall'Asta, A. Barrat, and A. Vespignani. Large scale networks fingerprinting and visualization using the k-core decomposition. In *Advances in neural information processing systems 18*. In NIPS '05. The MIT Press, 2005, pages 41–50 (cited on page 68).

[6] D. R. Amancio, R. Fabbri, O. N. Oliveira Jr, M. d. G. V. Nunes, and L. d. F. Costa. Opinion discrimination using complex network features. *Complex networks*, number 116 in Communications in Computer and Information Science, pages 154–162. Springer-Verlag Berlin, 2011 (cited on page 98).

[7] G. Amati and C. J. van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM transactions on information systems*, 20(4):357–389, Oct. 2002 (cited on pages 11, 44).

[8] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, G. Paliouras, and C. D. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. In *Proceedings of the workshop on machine learning in the new information age, 11th european conference on machine learning*, 2000, pages 9–17 (cited on page 102).

[9] V. N. Anh, O. de Kretser, and A. Moffat. Vector-space ranking with effective early termination. In *Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval*. In SIGIR '01. ACM, 2001, pages 35–42 (cited on page 48).

[10] L. Antiqueira, M. d. G. V. Nunes, O. N. Oliveira Jr, and L. d. F. Costa. Strong correlations between text quality and complex networks features. *Physica a: statistical mechanics and its applications*, 373:811–820, 2007 (cited on page 31).

[11] S. Arora, E. Mayfield, C. Penstein-Rosé, and E. Nyberg. Sentiment classification using automatically extracted subgraph features. In *Proceedings of the NAACL HLT 2010 workshop on computational approaches to analysis and generation of emotion in text*. In CAAGET '10. ACL, 2010, pages 131–139 (cited on pages 99, 100).

[12] S. Arora and E. Nyberg. Interactive annotation learning with indirect feature voting. In *Proceedings of human language technologies: the 2009 annual conference of the north american chapter of the association for computational linguistics, companion volume: student research workshop and doctoral consortium*. In NAACL '09. ACL, 2009, pages 55–60 (cited on page 30).

[13] R. Assam, M. Hassani, M. Brysch, and T. Seidl. (k, d)-core anonymity: structural anonymization of massive networks. In *Proceedings of the 26th international conference on scientific and statistical database management*. In SSDBM '14. ACM, 2014, 17:1–17:12 (cited on page 68).

[14] H. Balinsky, A. Balinsky, and S. J. Simske. Automatic text summarization and small-world networks. In *Proceedings of the 11th ACM symposium on document engineering*. In DocEng '11. ACM, 2011, pages 175–184 (cited on pages 29, 30).

[15] K. Barker and N. Cornacchia. Using noun phrase heads to extract document keyphrases. In *Proceedings of the 13th biennial conference of the canadian society on computational studies of intelligence: advances in artificial intelligence*. In AI '00. Springer-Verlag London, 2000, pages 40–52 (cited on page 13).

[16] R. Barzilay and L. Lee. Learning to paraphrase: an unsupervised approach using multiple-sequence alignment. In *Proceedings of the 2003 conference of the north american chapter of the association for computational linguistics on human language technology*. Volume 1. In HLT-NAACL '03. ACL, 2003, pages 16–23 (cited on page 30).

[17] N. Bassiou and C. Kotropoulos. Word clustering using PLSA enhanced with long distance bigrams. In *Proceedings of the 20th international conference on pattern recognition*. In ICPR '10. Springer-Verlag Berlin, 2010, pages 4226–4229 (cited on pages 28, 35, 74).

[18] V. Batagelj and M. Zaveršnik. Generalized cores. *The computing research repository (CoRR)*, cs.DS/0202039, 2002 (cited on pages 67, 82).

[19] V. Batagelj and M. Zaveršnik. An o(m) algorithm for cores decomposition of networks. *The computing research repository (CoRR)*, cs.DS/0310049, 2003 (cited on page 67).

[20] M. Baur, M. Gaertler, R. Görke, M. Krug, and D. Wagner. Generating graphs with predefined k-core structure. In *Proceedings of the 4th european conference on complex systems*. In ECCS '07, 2007 (cited on page 68).

[21] A. Bavelas. Communication patterns in task-oriented groups. *Journal of the acoustical society of america*, 22:725–730, 1950 (cited on page 61).

[22] M. A. Beauchamp. An improved index of centrality. *Behavioral science*, 10(2):161–163, 1965 (cited on page 61).

[23] S. Beliga and S. Martinčić-Ipšić. Node selectivity as a measure for graph-based keyword extraction in croatian news. In *Proceedings of the 6th international conference on information technologies and information society*. In ITIS '14, 2014, pages 8–17 (cited on page 31).

[24] N. J. Belkin and W. B. Croft. Information filtering and information retrieval: two sides of the same coin? *Communications of the ACM*, 35(12):29–38, Dec. 1992 (cited on page 10).

[25] W. Blacoe and M. Lapata. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*. In EMNLP-CoNLL '12. ACL, 2012, pages 546–556 (cited on pages 115, 116).

[26] R. Blanco and C. Lioma. Random walk term weighting for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval*. In SIGIR '07. ACM, 2007, pages 829–830 (cited on pages 30, 34, 46–48, 55).

[27] R. Blanco and C. Lioma. Graph-based term weighting for information retrieval. *Information retrieval*, 15(1):54–92, Feb. 2012 (cited on pages 29, 46, 48, 55, 77).

[28] I. Blank, L. Rokach, and G. Shani. Leveraging the citation graph to recommend keywords. In *Proceedings of the 7th ACM conference on recommender systems*. In RecSys '13. ACM, 2013, pages 359–362 (cited on page 14).

[29] J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boomboxes and blenders: domain adaptation for sentiment classification. In *Proceedings of the 45th annual meeting of the association of computational linguistics*. In ACL '07. ACL, 2007, pages 440–447 (cited on pages 102, 117).

[30] S. Bloehdorn, R. Basili, M. Cammisa, and A. Moschitti. Semantic kernels for text classification based on topological measures of feature similarity. In *Proceedings of the 6th IEEE international conference on data mining*. In ICDM '06. IEEE Computer Society, 2006, pages 808–812 (cited on page 114).

[31] S. Bloehdorn and A. Moschitti. Structure and semantics for expressive text kernels. In *Proceedings of the 16th ACM international conference on information and knowledge management*. In CIKM '07. ACM, 2007, pages 861–864 (cited on page 114).

[32] V. D. Blondel and P. Senellart. Automatic extraction of synonyms in a dictionary. In *Proceedings of the SIAM text mining workshop*, 2002 (cited on page 30).

[33] T. Bohne, S. Rönnau, and U. M. Borghoff. Efficient keyword extraction for meaningful document perception. In *Proceedings of the 11th ACM symposium on document engineering*. In DocEng '11. ACM, 2011, pages 185–194 (cited on page 14).

[34] B. Bollobás. *Extremal graph theory*. Academic Press, London, 1978 (cited on page 65).

[35] P. Bonacich. Power and centrality: a family of measures. *American journal of sociology*, 92(5):1170–82, 1987 (cited on page 62).

[36] F. Bonchi, F. Gullo, A. Kaltenbrunner, and Y. Volkovich. Core decomposition of uncertain graphs. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining*. In KDD '14. ACM, 2014, pages 1316–1325 (cited on page 68).

[37] A. Bookstein. The anomalous behaviour of precision in the swets model, and its resolution. *Journal of documentation*, 30(4):374–380, 1974 (cited on page 17).

[38] S. Bordag, G. Heyer, and U. Quasthoff. Small worlds of concepts and other principles of semantic search. *Innovative internet community systems*, in Lecture Notes in Computer Science, pages 10–19. Springer-Verlag Berlin, 2003 (cited on page 30).

[39] K. M. Borgwardt and H.-P. Kriegel. Shortest-path kernels on graphs. In *Proceedings of the 5th IEEE international conference on data mining*. In ICDM '05. IEEE Computer Society, 2005, pages 74–81 (cited on page 99).

[40] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th annual workshop on computational learning theory*. In COLT '92. ACM, 1992, pages 144–152 (cited on page 96).

[41] F. Boudin. A comparison of centrality measures for graph-based keyphrase extraction. In *Proceedings of the 6th international joint conference on natural language processing*. In IJCNLP '13. ACL, 2013, pages 834–838 (cited on pages 31, 48, 65, 75).

[42] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, New York, NY, USA, 2004 (cited on page 95).

[43] U. Brandes. A faster algorithm for betweenness centrality. *Journal of mathematical sociology*, 25(2):163–177, 2001 (cited on page 64).

[44] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30:107–117, 1998 (cited on page 63).

[45] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. *Computer networks and ISDN systems*, 29(8):1157–1166, 1997 (cited on page 9).

[46] C. Bron and J. Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577, Sept. 1973 (cited on page 68).

[47] P. F. Brown, V. J. Della Pietra, S. A. Della Pietra, and R. L. Mercer. The mathematics of statistical machine translation: parameter estimation. *Computational linguistics*, 19(2):263–311, June 1993 (cited on page 116).

[48] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. Della Pietra, and J. C. Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, Dec. 1992 (cited on page 28).

[49] J. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on research and development in information retrieval*. In SIGIR '98. ACM, 1998, pages 335–336 (cited on page 10).

[50] A. Cardoso-Cachopo. Improving methods for single-label text categorization. Ph.D. thesis. Lisbon, Portugal: Instituto Superior Técnico, Universidade de Lisboa, Oct. 2007. 167 pages (cited on page 102).

[51] C. Carpineto and G. Romano. Semantic search log k-anonymization with generalized k-cores of query concept graph. *Advances in information retrieval*, pages 110–121. Springer, 2013 (cited on page 68).

[52] J. Casas-Roma and F. Rousseau. Community-preserving generalization of social networks. In *Proceedings of the social media and risk ASONAM 2015 workshop*. In SoMeRis '15. IEEE Computer Society, 2015 (cited on pages x, 68).

[53] W. B. Cavnar and J. M. Trenkle. N-gram-based text categorization. In *Proceedings of the 3rd annual symposium on document analysis and information retrieval*. In SDAIR '94, 1994, pages 161–175 (cited on page 28).

[54] P. Chandar and B. A. Carterette. Diversification of search results using webgraphs. In *Proceedings of the 33rd annual international ACM SIGIR conference on research and development in information retrieval*. In SIGIR '10. ACM, 2010, pages 869–870 (cited on page 11).

[55] C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines. *ACM transactions on intelligent systems and technology*, 2(3):27:1–27:27, 2011 (cited on page 118).

[56] M. Choudhury, M. Thomas, A. Mukherjee, A. Basu, and N. Ganguly. How difficult is it to develop a perfect spell-checker? a cross-linguistic analysis through complex network approach. In *Proceedings of the 2nd workshop on TextGraphs: graph-based algorithms for natural language processing*. In TextGraphs-2. ACL, 2007, pages 81–88 (cited on page 31).

[57] K. W. Church and W. A. Gale. Inverse document frequency (IDF): a measure of deviation from poisson. In *Proceedings of the 3rd workshop on very large corpora*, 1995, pages 121–130 (cited on pages 42, 44).

[58] K. W. Church and P. Hanks. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29, Mar. 1990 (cited on page 29).

[59] C. L. A. Clarke, G. V. Cormack, and F. J. Burkowski. Shortest substring ranking (MultiText experiments for TREC-4). In *Proceedings of the 4th text REtrieval conference*. In TREC-4, 1995, pages 295–304 (cited on page 46).

[60] C. L. A. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. In *Proceedings of the 31st annual international ACM SIGIR conference on research and development in information retrieval*. In SIGIR '08. ACM, 2008, pages 659–666 (cited on page 11).

[61] W. S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the american statistical association*, 74(368):829–836, 1979 (cited on page 54).

[62] C. W. Cleverdon. On the inverse relationship of recall and precision. *Journal of documentation*, 28(3):195–201, 1972 (cited on page 17).

[63] C. W. Cleverdon and J. Mills. The testing of index language devices. In *ASLIB proceedings*. Volume 15, 1963, pages 106–130 (cited on page 17).

[64] S. Clinchant and E. Gaussier. Information-based models for ad hoc IR. In *Proceedings of the 33rd annual international ACM SIGIR conference on research and development in information retrieval*. In SIGIR '10. ACM, 2010, pages 234–241 (cited on pages 11, 39).

[65] M. Collins and N. Duffy. Convolution kernels for natural language. In *Advances in neural information processing systems 14*. In NIPS '01. The MIT Press, 2001, pages 625–632 (cited on page 114).

[66] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12:2493–2537, Nov. 2011 (cited on page 115).

[67] W. J. Conover. *Practical nonparametric statistics*. John Wiley and Sons, New York, NY, USA, 2nd edition, 1980 (cited on pages 22, 26).

[68] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. An improved algorithm for matching large graphs. In *Proceedings of the 3rd IAPR-TC15 workshop on graph-based representations in pattern recognition*, 2001, pages 149–159 (cited on page 100).

[69] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. The MIT Press, 3rd edition, 2009 (cited on page 67).

[70] C. Cortes and V. N. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995 (cited on page 97).

[71] R. H. Creecy, B. M. Masand, S. J. Smith, and D. L. Waltz. Trading MIPS and memory for knowledge engineering. *Communications of the ACM*, 35(8):48–64, Aug. 1992 (cited on page 13).

[72] D. Croce, A. Moschitti, and R. Basili. Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of the 2011 conference on empirical methods in natural language processing*. In EMNLP '11. ACL, 2011, pages 1034–1046 (cited on page 114).

[73] M. Crochemore and R. Vérin. Direct construction of compact directed acyclic word graphs. *Combinatorial pattern matching*, in Lecture Notes in Computer Science, pages 116–129. Springer-Verlag Berlin, 1997 (cited on page 30).

[74] W. B. Croft and D. J. Harper. Using probabilistic models of document retrieval without relevance information. *Journal of documentation*, 35(4):285–295, 1979 (cited on page 44).

[75] G. Csardi and T. Nepusz. The igraph software package for complex network research. *InterJournal, complex systems*, 1695(5):1–9, 2006 (cited on page 5).

[76] A. Culotta and J. Sorensen. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd annual meeting of the association for computational linguistics*. In ACL '04. ACL, 2004, pages 423–429 (cited on pages 114, 119).

[77] P. Cunningham. Dimension reduction. (UCD-CSI-2007-7). University College Dublin, 2007 (cited on page 8).

[78] D. Cutting and J. O. Pedersen. Optimization for dynamic inverted index maintenance. In *Proceedings of the 13th annual international ACM SIGIR conference on research and development in information retrieval*. In SIGIR '90. ACM, 1990, pages 405–411 (cited on page 12).

[79] J. Davis and M. Goadrich. The relationship between precision-recall and ROC curves. In *Proceedings of the 23rd international conference on machine learning*. In ICML '06. ACM, 2006, pages 233–240 (cited on pages 19, 78).

[80] F. Debole and F. Sebastiani. An analysis of the relative hardness of reuters-21578 subsets: research articles. *Journal of the american society for information science and technology*. JASIST '05 56(6):584–596, Apr. 2005 (cited on page 102).

[81] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the american society for information science*. JASIS '90 41(6):391–407, 1990 (cited on page 27).

[82] M. Deshpande, M. Kuramochi, N. Wale, and G. Karypis. Frequent substructure-based approaches for classifying chemical compounds. *IEEE transactions on knowledge and data engineering*, 17(8):1036–1050, Aug. 2005 (cited on page 98).

[83] P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine learning*, 29(2):103–130, 1997 (cited on page 85).

[84] S. N. Dorogovtsev and J. F. F. Mendes. Language as an evolving word web. *Proceedings of the royal society of london. series b: biological sciences*, 268(1485):2603–2606, 2001 (cited on page 30).

[85] M. Eidsaa and E. Almaas. S-core network decomposition: a generalization of k-core analysis to weighted networks. *Physical review e*, 88(6):062819, 2013 (cited on page 68).

[86] G. Erkan and D. R. Radev. LexRank: graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22(1):457–479, July 2004 (cited on pages 29, 30).

[87] S. Eyheramendy, D. D. Lewis, and D. Madigan. On the naive bayes model for text categorization. In *Proceedings of the 9th international workshop on artificial intelligence and statistics*. In AISTATS '03, 2003, pages 332–339 (cited on page 87).

[88] J. L. Fagan. Automatic phrase indexing for document retrieval. In *Proceedings of the 10th annual international ACM SIGIR conference on research and development in information retrieval*. In SIGIR '87. ACM, 1987, pages 91–101 (cited on page 28).

[89] H. Fang, T. Tao, and C. Zhai. A formal study of information retrieval heuristics. In *Proceedings of the 27th annual international ACM SIGIR conference on research and development in information retrieval*. In SIGIR '04. ACM, 2004, pages 49–56 (cited on pages 11, 38, 39, 41, 42).

[90] H. Fang, T. Tao, and C. Zhai. Diagnostic evaluation of information retrieval models. *ACM transactions on information systems*, 29(2):7:1–7:42, Apr. 2011 (cited on pages 11, 38).

[91] F. Faul, E. Erdfelder, A.-G. Lang, and A. Buchner. G*power 3: a flexible statistical power analysis program for the social, behavioral, and biomedical sciences. *Behavior research methods*, 39(2):175–191, 2007 (cited on page 26).

[92] R. Ferrer i Cancho and R. V. Solé. The small world of human language. *Proceedings of the royal society of london. series b, biological sciences*, 268:2261–2266, 2001 (cited on pages 30–32, 35).

[93] R. Ferrer i Cancho, R. V. Solé, and R. Köhler. Patterns in syntactic dependency networks. *Physical review e*, 69(5):051915, 2004 (cited on pages 30, 31).

[94] O. Ferret. Using collocations for topic segmentation and link detection. In *Proceedings of the 19th international conference on computational linguistics*. Volume 1. In COLING '02. ACL, 2002, pages 1–7 (cited on page 30).

[95] F. Fidler, C. Geoff, B. Mark, and T. Neil. Statistical reform in medicine, psychology and ecology. *The journal of socio-economics*, 33(5):615–630, 2004 (cited on page 26).

[96] K. Filippova. Multi-sentence compression: finding shortest paths in word graphs. In *Proceedings of the 23rd international conference on computational linguistics*. In COLING '10. ACL, 2010, pages 322–330 (cited on pages 30, 32, 77).

[97] J. R. Firth. A synopsis of linguistic theory, 1930-55. *Studies in linguistic analysis*:1–32, 1957 (cited on pages 27, 29).

[98] N. Ford. Information retrieval and creativity: towards support for the original thinker. *Journal of documentation*, 55(5):528–542, 1999 (cited on page 10).

[99] G. Forman. Tackling concept drift by temporal inductive transfer. In *Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval*. In SIGIR '06. ACM, 2006, pages 252–259 (cited on page 89).

[100] D. P. Foster and E. I. George. The risk inflation criterion for multiple regression. *The annals of statistics*:1947–1975, 1994 (cited on page 93).

[101] E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin, and C. G. Nevill-Manning. Domain-specific keyphrase extraction. In *Proceedings of the 16th international joint conference on artificial intelligence*. Volume 2. In IJCAI '99. Morgan Kaufmann Publishers Inc., 1999, pages 668–673 (cited on page 13).

[102] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615, 1987 (cited on page 67).

[103] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, (40):35–41, 1977 (cited on page 61).

[104] L. C. Freeman. Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239, 1979 (cited on page 47).

[105] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007 (cited on page 68).

[106] J. H. Friedman. Another approach to polychotomous classification. Department of Statistics, Stanford University, 1996 (cited on page 97).

[107] J. H. Friedman. On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data mining and knowledge discovery*, 1(1):55–77, Jan. 1997 (cited on page 85).

[108] J. Fürnkranz. A study using n-gram features for text categorization. (OEFAI-TR-98-30). Austrian Research Institute for Artificial Intelligence, 1998 (cited on pages 28, 101).

[109] M. Gamon. Graph-based text representation for novelty detection. In *Proceedings of the 1st workshop on graph based methods for natural language processing*. In TextGraphs-1. ACL, 2006, pages 17–24 (cited on pages 30, 32).

[110] K. Ganesan, C. Zhai, and J. Han. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd international conference on computational linguistics*. In COLING '10. ACL, 2010, pages 340–348 (cited on page 30).

[111] A. Garas, F. Schweitzer, and S. Havlin. A k-shell decomposition method for weighted networks. *New journal of physics*, 14(8):083030, 2012 (cited on page 68).

[112] M. R. Garey and D. S. Johnson. *Computers and intractability; a guide to the theory of NP-completeness*. W. H. Freeman & Co., 1990 (cited on page 100).

[113] T. Gärtner, P. Flach, and S. Wrobel. On graph kernels: hardness results and efficient alternatives. *Learning theory and kernel machines*, pages 129–143. Springer, Jan. 2003 (cited on pages 99, 118).

[114] T. Gärtner, P. Flach, and S. Wrobel. On graph kernels: hardness results and efficient alternatives. In *Proceedings of the annual conference on computational learning theory*. In COLT '03, 2003, pages 129–143 (cited on page 115).

[115] A. Genkin, D. D. Lewis, and D. Madigan. Large-scale bayesian logistic regression for text categorization. *Technometrics*, 49(3):291–304, 2007 (cited on page 13).

[116] G. Giannakopoulos, V. Karkaletsis, G. Vouros, and P. Stamatopoulos. Summarization system evaluation revisited: n-gram graphs. *ACM transactions on speech and language processing*, 5(3):1–39, Oct. 2008 (cited on page 30).

[117] C. Giatsidis, D. M. Thilikos, and M. Vazirgiannis. D-cores: measuring collaboration of directed graphs based on degeneracy. In *Proceedings of the 11th IEEE international conference on data mining*. In ICDM '11. IEEE Computer Society, 2011, pages 201–210 (cited on pages 60, 68).

[118] C. Giatsidis, D. M. Thilikos, and M. Vazirgiannis. Evaluating cooperation in communities with the k-core structure. In *Proceedings of the IEEE international conference on advances in social networks analysis and mining*. In ASONAM '11. ACM, 2011, pages 87–93 (cited on page 68).

[119] C. Giatsidis, M. Vazirgiannis, D. M. Thilikos, and B. Cautis. Quantifying trust dynamics in signed graphs, the s-cores approach. In *Proceedings of the 13th SIAM international conference on data mining*. In SDM '14. SIAM, 2014, pages 668–676 (cited on page 68).

[120] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002 (cited on page 68).

[121] J. T. Goodman. A bit of progress in language modeling: extended version. (MSR-TR-2001-72). Microsoft Research, 2001 (cited on page 28).

[122] D. M. Green and J. A. Swets. *Signal detection theory and psychophysics*. John Wiley and Sons, New York, NY, USA, 1966 (cited on page 19).

[123] M. Grineva, M. Grinev, and D. Lizorkin. Extracting key terms from noisy and multitheme documents. In *Proceedings of the 18th international conference on world wide web*. In WWW '09. ACM, 2009, pages 661–670 (cited on pages 65, 74).

[124] D. Guthrie, B. Allison, W. Liu, L. Guthrie, and Y. Wilks. A closer look at skip-gram modelling. In *Proceedings of the 5th international conference on language resources and evaluation*. In LREC '06, 2006, pages 1–4 (cited on page 28).

[125] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th python in science conference*. In SciPy '08, 2008, pages 11–15 (cited on page 5).

[126] Z. Harris. Distributional structure. *Word*, 10(23):146–162, 1954 (cited on pages 8, 27).

[127] K. S. Hasan and V. Ng. Conundrums in unsupervised keyphrase extraction: making sense of the state-of-the-art. In *Proceedings of the 23rd international conference on computational linguistics: posters*. In COLING '10. ACL, 2010, pages 365–373 (cited on page 76).

[128] S. Hassan, R. Mihalcea, and C. Banea. Random-walk term weighting for improved text classification. In *Proceedings of the international conference on semantic computing*. In ICSC '07, 2007, pages 242–249 (cited on pages 30, 98).

[129] T. Hastie, R. Tibshirani, and J. H. Friedman. *The elements of statistical learning*. Volume 2(1). Springer, 2009 (cited on page 91).

[130] M. H. Heine. The inverse relationship of precision and recall in terms of the swets model. *Journal of documentation*, 29:81–84, 1973 (cited on page 17).

[131] C. Helma, R. D. King, S. Kramer, and A. Srinivasan. The predictive toxicology challenge 2000–2001. *Bioinformatics*, 17(1):107–108, 2001 (cited on pages 100, 101).

[132] A. E. Hoerl and R. W. Kennard. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970 (cited on page 91).

[133] T. Horváth, T. Gärtner, and S. Wrobel. Cyclic pattern kernels for predictive graph mining. In *Proceedings of the 10th ACM SIGKDD international conference on knowledge discovery and data mining*. In KDD '04. ACM, 2004, pages 158–167 (cited on page 99).

[134] J. Huan, W. Wang, and J. Prins. Efficient mining of frequent subgraphs in the presence of isomorphism. In *Proceedings of the 3rd IEEE international conference on data mining*. In ICDM '03. IEEE Computer Society, 2003, pages 549–552 (cited on page 98).

[135] D. A. Hull. Using statistical testing in the evaluation of retrieval experiments. In *Proceedings of the 16th annual international ACM SIGIR conference on research and development in information retrieval*. In SIGIR '93. ACM, 1993, pages 329–338 (cited on pages 19, 22, 23).

[136] D. A. Hull, G. Grefenstette, B. M. Schulze, É. Gaussier, H. Schütze, and J. O. Pedersen. Xerox TREC-5 site report: routing, filtering, NLP, and spanish tracks. In *Proceedings of the 5th text REtrieval conference*. In TREC-5, 1996 (cited on page 10).

[137] A. Hulth. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 8th conference on empirical methods in natural language processing*. In EMNLP '03. ACL, 2003, pages 216–223 (cited on pages 13–15, 75, 76, 111).

[138] A. Hulth and B. B. Megyesi. A study on automatically extracted keywords in text categorization. In *Proceedings of the 21st international conference on computational linguistics and the 44th annual meeting of the association for computational linguistics*. In ACL '06. ACL, 2006, pages 537–544 (cited on page 111).

[139] H. Jeffreys. *The theory of probability*. Oxford University Press, 2nd edition, 1948 (cited on page 43).

[140] C. Jiang, F. Coenen, R. Sanderson, and M. Zito. Text classification using graph mining-based feature extraction. *Knowledge-based systems*, 23(4):302–308, 2010 (cited on pages 30, 99, 100).

[141] N. Jin, C. Young, and W. Wang. GAIA: graph classification using evolutionary computation. In *Proceedings of the 2010 ACM SIGMOD international conference on management of data*. In SIGMOD '10. ACM, 2010, pages 879–890 (cited on page 99).

[142] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of the 10th european conference on machine learning*. In ECML '98, 1998, pages 137–142 (cited on pages 9, 12, 101, 103).

[143] T. Joachims. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining*. In KDD '06. ACM, 2006, pages 217–226 (cited on page 111).

[144] W. E. Johnson. Probability: the deductive and inductive problems. *Mind*, 41(164):409–423, 1932 (cited on page 43).

[145] H. Joho and M. Sanderson. Document frequency and term specificity. In *Proceedings of the 8th international conference on computer-assisted information retrieval*. In RIAO '07, 2007, pages 350–359 (cited on page 44).

[146] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information processing letters*, 31(1):7–15, Apr. 1989 (cited on page 32).

[147] V. Kapustin and A. Jamsen. Vertex degree distribution for the graph of word co-occurrences in russian. In *Proceedings of the 2nd workshop on TextGraphs: graph-based algorithms for natural language processing*. In TextGraphs-2. ACL, 2007, pages 89–92 (cited on page 31).

[148] M. Karkali, V. Plachouras, C. Stefanatos, and M. Vazirgiannis. Keeping keywords fresh: a BM25 variation for personalized keyword extraction. In *Proceedings of the 2nd temporal web analytics workshop*. In TempWeb '12, 2012, pages 17–24 (cited on page 14).

[149] M. Karkali, F. Rousseau, A. Ntoulas, and M. Vazirgiannis. Efficient online novelty detection in news streams. In *Proceedings of the 14th international conference on web information systems engineering*. In WISE '13. Springer-Verlag Berlin, 2013, pages 57–71 (cited on pages ix, 11, 43, 44).

[150] H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In *Proceedings of the 20th international conference on machine learning*. Volume 3. In ICML '03. AAAI Press, 2003, pages 321–328 (cited on page 99).

[151] A. Kent, M. M. Berry, F. U. Luehrs, and J. W. Perry. Machine literature searching VIII. operational criteria for designing information retrieval systems. *American documentation*, 6(2):93–101, 1955 (cited on page 17).

[152] T. Kenter and M. de Rijke. Short text similarity with word embeddings. In *Proceedings of the 24th ACM international conference on information and knowledge management*. In CIKM '15. ACM, 2015 (cited on page 115).

[153] J. Kim, F. Rousseau, and M. Vazirgiannis. Convolutional sentence kernel from word embeddings for short text categorization. In *Proceedings of the 2015 conference on empirical methods in natural language processing*. In EMNLP '15. ACL, 2015 (cited on pages ix, 28, 113, 125, 126).

[154] S.-B. Kim, H.-C. Seo, and H.-C. Rim. Poisson naive bayes for text classification with feature weighting. In *Proceedings of the 6th international workshop on information retrieval with asian languages*. Volume 11. In AsianIR '03, 2003, pages 33–40 (cited on page 87).

[155] M. Kitsak, L. K. Gallos, S. Havlin, F. Liljeros, L. Muchnik, H. E. Stanley, and H. A. Makse. Identification of influential spreaders in complex networks. *Nature physics*, 6(11):888–893, 2010 (cited on page 68).

[156] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, Sept. 1999 (cited on page 63).

[157] P. Koehn, F. J. Och, and D. Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 conference of the north american chapter of the association for computational linguistics on human language technology*. Volume 1. In NAACL '03. ACL, 2003, pages 48–54 (cited on page 28).

[158] Z. Kozareva, E. Riloff, and E. H. Hovy. Semantic class learning from the web with hyponym pattern linkage graphs. In *Proceedings of the 46th annual meeting of the association for computational linguistics: human language technologies*. Volume 8. In ACL '08. ACL, 2008, pages 1048–1056 (cited on page 30).

[159] M. Krapivin, A. Autaeu, and M. Marchese. Large dataset for keyphrases extraction. (DISI-09-055). University of Trento, May 2009 (cited on page 75).

[160] T. Kudo, E. Maeda, and Y. Matsumoto. An application of boosting to graph classification. In *Advances in neural information processing systems 17*. In NIPS '04. The MIT Press, 2004, pages 729–736 (cited on page 98).

[161] T. Kudo and Y. Matsumoto. A boosting algorithm for classification of semi-structured text. In *Proceedings of the 9th conference on empirical methods in natural language processing*. Volume 4. In EMNLP '04. ACL, 2004, pages 301–308 (cited on page 99).

[162] S. Lahiri. Complexity of word collocation networks: a preliminary structural analysis. In *Proceedings of the student research workshop at the 14th conference of the european chapter of the association for computational linguistics*. In EACL '14. ACL, 2014, pages 96–105 (cited on page 30).

[163] S. Lahiri, S. R. Choudhury, and C. Caragea. Keyword and keyphrase extraction using centrality measures on collocation networks. *The computing research repository (CoRR)*, abs/1401.6571, 2014 (cited on pages 30, 65, 75).

[164] S. Lahiri and R. Mihalcea. Using n-gram and word network features for native language identification. In *Proceedings of the 8th workshop on innovative use of NLP for building educational applications*. In NAACL-HLT '13. ACL, 2013, pages 251–259 (cited on page 30).

[165] F. W. Lancaster. *Information retrieval systems: characteristics, testing, and evaluation*. Of *Information Sciences Series*. John Wiley and Sons, New York, NY, USA, 1968 (cited on page 10).

[166] P.-S. Laplace. *Essai philosophique sur les probabilités*. Mme. Ve Courcier, Paris, France, 1814 (cited on page 43).

[167] L. S. Larkey and W. B. Croft. Combining classifiers in text categorization. In *Proceedings of the 19th annual international ACM SIGIR conference on research and development in information retrieval*. In SIGIR '96. ACM, 1996, pages 289–297 (cited on pages 13, 103).

[168] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st international conference on machine learning*. Volume 32. In ICML '14. JMLR Workshop and Conference Proceedings, 2014, pages 1188–1196 (cited on page 121).

[169] L. Lee. IDF revisited: a simple new derivation within the robertson-spärck jones probabilistic model. In *Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval*. In SIGIR '07. ACM, 2007, pages 751–752 (cited on page 44).

[170] J. T. Leek and R. D. Peng. Statistics: p values are just the tip of the iceberg. *Nature*, 520(7549):612–612, 2015 (cited on page 26).

[171] J. Leskovec, M. Grobelnik, and N. Milic-Frayling. Learning semantic graph mapping for document summarization. In *Proceedings of the ECML/PKDD workshop on knowledge discovery and ontologies*. In KDO '04, 2004 (cited on page 30).

[172] O. Levy and Y. Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd annual meeting of the association for computational linguistics*. Volume 2. In ACL '14. ACL, 2014, pages 302–308 (cited on page 117).

[173] O. Levy, Y. Goldberg, and I. Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the association for computational linguistics*, 3:211–225, 2015 (cited on page 117).

[174] F. Li and Y. Yang. A loss function analysis for classification methods in text categorization. In *Proceedings of the 20th international conference on machine learning*. In ICML '03. AAAI Press, 2003, pages 472–479 (cited on page 90).

[175] R.-H. Li, J. X. Yu, and R. Mao. Efficient core maintenance in large dynamic graphs. *IEEE transactions on knowledge and data engineering*, 26(10):2453–2465, Oct. 2014 (cited on page 68).

[176] W. Liang, Y. Shi, C. K. Tse, J. Liu, Y. Wang, and X. Cui. Comparison of co-occurrence networks of the chinese and english languages. *Physica a: statistical mechanics and its applications*, 388(23):4901–4909, 2009 (cited on page 31).

[177] W. Liang, C.-N. Huang, M. Li, and B.-L. Lu. Extracting keyphrases from chinese news articles using TextRank and query log knowledge. In *Proceedings of the 23rd pacific asia conference on language, information and computation*. In PACLIC '09, 2009, pages 733–740 (cited on page 64).

[178] M. Litvak and M. Last. Graph-based keyword extraction for single-document summarization. In *Proceedings of the workshop on multi-source multilingual information extraction and summarization*. In MMIES '08, 2008, pages 17–24 (cited on pages 13, 14, 30, 32, 65, 77).

[179] Z. Liu, P. Li, Y. Zheng, and M. Sun. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of the 2009 conference on empirical methods in natural language processing*. In EMNLP '09. ACL, 2009, pages 257–266 (cited on pages 75, 76, 82).

[180] D. E. Losada, L. Azzopardi, and M. Baillie. Revisiting the relationship between document length and relevance. In *Proceedings of the 17th ACM international conference on information and knowledge management*. In CIKM '08. ACM, 2008, pages 419–428 (cited on page 40).

[181] H. P. Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM journal of research and development*, 1(4):309–317, Oct. 1957 (cited on pages 9, 38).

[182] H. P. Luhn. The automatic creation of literature abstracts. *IBM journal of research and development*, 2(2):159–165, Apr. 1958 (cited on page 14).

[183] Y. Lv and C. Zhai. Adaptive term frequency normalization for BM25. In *Proceedings of the 20th ACM international conference on information and knowledge management*. In CIKM '11. ACM, 2011, pages 1985–1988 (cited on pages 39, 44).

[184] Y. Lv and C. Zhai. Lower-bounding term frequency normalization. In *Proceedings of the 20th ACM international conference on information and knowledge management*. In CIKM '11. ACM, 2011, pages 7–16 (cited on pages 11, 41, 42, 45, 49, 50, 54, 55).

[185] Y. Lv and C. Zhai. When documents are very long, BM25 fails! In *Proceedings of the 34th annual international ACM SIGIR conference on research and development in information retrieval*. In SIGIR '11. ACM, 2011, pages 1103–1104 (cited on pages 42, 44, 45).

[186] Y. Lv and C. Zhai. A log-logistic model-based interpretation of TF normalization of BM25. In *Proceedings of the 34th european conference on information retrieval*. In ECIR '12. Springer-Verlag, 2012, pages 244–255 (cited on page 44).

[187] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*. Volume 1. In HLT '11. ACL, 2011, pages 142–150 (cited on page 115).

[188] P. Mahé, N. Ueda, T. Akutsu, J.-L. Perret, and J.-P. Vert. Extensions of marginalized graph kernels. In *Proceedings of the 21st international conference on machine learning*. In ICML '04. ACM, 2004, pages 70–78 (cited on page 99).

[189] F. D. Malliaros and K. Skianis. Graph-based term weighting for text categorization. In *Proceedings of the social media and risk ASONAM 2015 workshop*. In SoMeRis '15. IEEE Computer Society, 2015 (cited on page 98).

[190] F. D. Malliaros and M. Vazirgiannis. To stay or not to stay: modeling engagement dynamics in social graphs. In *Proceedings of the 22nd ACM international conference on information and knowledge management*. In CIKM '13. ACM, 2013, pages 469–478 (cited on page 68).

[191] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge University Press, New York, NY, USA, 2008 (cited on pages 2, 8, 10, 21, 39, 42, 43, 89).

[192] C. D. Manning and H. Schütze. *Foundations of statistical natural language processing*. MIT Press, 1999 (cited on page 29).

[193] A. Markov, M. Last, and A. Kandel. Fast categorization of web documents represented by graphs. *Advances in web mining and web usage analysis*, number 4811 in Lecture Notes in Artificial Intelligence, pages 56–71. Springer, 2007 (cited on pages 30, 99).

[194] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki. The DET curve in assessment of detection task performance. In *Proceedings of the 5th european conference on speech communication and technology*. In Eurospeech '97, 1997, pages 1899–1903 (cited on page 19).

[195] A. P. Masucci and G. J. Rodgers. Network properties of written human language. *Physical review e*, 74(2):026102, 2006 (cited on page 30).

[196] S. Matsumoto, H. Takamura, and M. Okumura. Sentiment classification using word sub-sequences and dependency sub-trees. In *Proceedings of the 9th pacific-asia conference on advances in knowledge discovery and data mining*. In PAKDD '05. Springer-Verlag, 2005, pages 301–311 (cited on page 99).

[197] Y. Matsuo, Y. Ohsawa, and M. Ishizuka. A document as a small world. *New frontiers in artificial intelligence*, pages 444–448. Springer, 2001 (cited on page 35).

[198] Y. Matsuo, Y. Ohsawa, and M. Ishizuka. KeyWorld: extracting keywords from a document as a small world. In *In proceedings the 4th international conference on discovery science*. In DS '01, 2001 (cited on pages 13, 30, 62, 64, 68).

[199] D. Mavroeidis, G. Tsatsaronis, M. Vazirgiannis, M. Theobald, and G. Weikum. Word sense disambiguation for exploiting hierarchical thesauri in text classification. In *Proceedings of the 9th european conference on principles and practice of knowledge discovery in databases*. In ECML PKDD '05. Springer-Verlag Berlin, 2005, pages 181–192 (cited on page 114).

[200] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *Proceedings of the AAAI workshop on learning for text categorization*. In AAAI '98. AAAI Press, 1998, pages 41–48 (cited on pages 13, 86, 103).

[201] K. R. McKeown, R. J. Passonneau, D. K. Elson, A. Nenkova, and J. Hirschberg. Do summaries help? In *Proceedings of the 28th annual international ACM SIGIR conference on research and development in information retrieval*. In SIGIR '05. ACM, 2005, pages 210–217 (cited on page 14).

[202] Q. McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947 (cited on page 26).

[203] P. Meladianos, G. Nikolentzos, F. Rousseau, Y. Stavrakas, and M. Vazirgiannis. Degeneracy-based real-time sub-event detection in twitter stream. In *Proceedings of the 9th AAAI international conference on web and social media*. In ICWSM '15. AAAI Press, 2015, pages 248–257 (cited on pages ix, 14, 29, 30).

[204] P. Meladianos, G. Nikolentzos, F. Rousseau, Y. Stavrakas, and M. Vazirgiannis. Shortest-path graph kernels for document similarity. In *Proceedings of the 15th IEEE international conference on data mining*. In ICDM '15. IEEE Computer Society, 2015 (cited on pages x, 99, 107, 125, 126).

[205] J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical transactions of the royal society of london. series a, containing papers of a mathematical or physical character*, 209(456):415–446, 1909 (cited on page 96).

[206] V. Metsis, I. Androutsopoulos, and G. Paliouras. Spam filtering with naive bayes-which naive bayes? In *Proceedings of the 3rd conference on email and anti-spam*. In CEAS '06, 2006, pages 27–28 (cited on pages 13, 86).

[207] D. Metzler. Generalized inverse document frequency. In *Proceedings of the 17th ACM international conference on information and knowledge management*. In CIKM '08. ACM, 2008, pages 399–408 (cited on page 44).

[208] R. Mihalcea and P. Tarau. TextRank: bringing order into texts. In *Proceedings of the 9th conference on empirical methods in natural language processing*. In EMNLP '04. ACL, 2004, pages 404–411 (cited on pages 13, 14, 29–32, 64, 74–77).

[209] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *Proceedings of workshop at international conference on learning representations*. In ICLR '13, 2013 (cited on pages 28, 115, 125).

[210] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems 26*. In NIPS '13. Neural Information Processing Systems, 2013, pages 3111–3119 (cited on page 115).

[211] S. Milgram. The small world problem. *Psychology today*, 2(1):60–67, 1967 (cited on page 35).

[212] G. A. Miller. WordNet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, Nov. 1995 (cited on page 114).

[213] D. Miorandi and F. De Pellegrini. K-shell decomposition for dynamic complex networks. In *Proceedings of the 8th international symposium on modeling and optimization in mobile, ad hoc and wireless networks (WiOpt)*, 2010, pages 488–496 (cited on page 68).

[214] M. Mitra, C. Buckley, A. Singhal, and C. Cardie. An analysis of statistical and syntactic phrases. In *Proceedings of the 5th international conference on computer-assisted information retrieval*. Volume 97. In RIAO '97, 1997, pages 200–214 (cited on page 28).

[215] C. N. Mooers. Zatocoding applied to mechanical organization of knowledge. *American documentation*, 2(1):20–32, 1951 (cited on page 10).

[216] L. Moreira-Matias, J. Mendes-Moreira, J. Gama, and P. Brazdil. Text categorization using an ensemble classifier based on a mean co-association matrix. In *Proceedings of the 8th international conference on machine learning and data mining in pattern recognition*. In MLDM '12, 2012, pages 525–539 (cited on page 21).

[217] S.-H. Na. Two-stage document length normalization for information retrieval. *ACM transactions on information systems*, 33(2):8:1–8:40, Feb. 2015 (cited on page 40).

[218] P. Nakov, Z. Kozareva, A. Ritter, S. Rosenthal, V. Stoyanov, and T. Wilson. Semeval-2013 task 2: sentiment analysis in twitter. In *Proceedings of the 7th international workshop on semantic evaluation*. In SemEval-2013, 2013 (cited on page 117).

[219] B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24(2):227–234, Apr. 1995 (cited on page 93).

[220] A. Nenkova and K. R. McKeown. Automatic summarization. *Foundations and trends in information retrieval*, 5(2):103–233, 2011 (cited on page 14).

[221] M. E. J. Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003 (cited on page 63).

[222] A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: a comparison of logistic regression and naive bayes. In *Advances in neural information processing systems 15*. In NIPS '02. The MIT Press, 2002, pages 841–848 (cited on page 94).

[223] S. Nijssen and J. N. Kok. A quickstart in frequent structure mining can make a difference. In *Proceedings of the 10th ACM SIGKDD international conference on knowledge discovery and data mining*. In KDD '04. ACM, 2004, pages 647–652 (cited on page 98).

[224] Y. Ohsawa, N. E. Benson, and M. Yachida. KeyGraph: automatic indexing by co-occurrence graph based on building construction metaphor. In *Proceedings of the advances in digital libraries conference*. In ADL '98. IEEE Computer Society, 1998, pages 12–18 (cited on pages 13, 29–32, 64).

[225] J. O'Madadhain, D. Fisher, S. White, P. Smyth, and Y. Boey. Analysis and visualization of network data using JUNG. *Journal of statistical software*, 10(2):1–35, 2005 (cited on page 5).

[226] I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, and C. Lioma. Terrier: a high performance and scalable information retrieval platform. In *Proceedings of ACM SIGIR '06 workshop on open source information retrieval*. In OSIR '06. ACM, 2006 (cited on page 5).

[227] A. Özgür, L. Özgür, and T. Güngör. Text categorization with class-based and corpus-based keyword selection. In *Proceedings of the 20th international conference on computer and information sciences*. In ISCIS '05, 2005, pages 606–615 (cited on page 111).

[228] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: bringing order to the web. (1999-0120). Stanford University, 1999 (cited on page 63).

[229] G. K. Palshikar. Keyword extraction from a single document using centrality measures. In *Proceedings of the 2nd international conference on pattern recognition and machine intelligence*. In PReMI '07. Springer-Verlag, 2007, pages 503–510 (cited on pages 30, 65).

[230] B. Pang and L. Lee. A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on association for computational linguistics*. In ACL '04. ACL, 2004, pages 271–278 (cited on page 117).

[231] B. Pang and L. Lee. Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*. In ACL '05. ACL, 2005, pages 115–124 (cited on page 117).

[232] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the 7th conference on empirical methods in natural language processing*. Volume 10. In EMNLP '02. ACL, 2002, pages 79–86 (cited on page 37).

[233] K. Papineni. Why inverse document frequency? In *Proceedings of the 2nd meeting of the north american chapter of the association for computational linguistics on language technologies*. In NAACL '01. ACL, 2001, pages 1–8 (cited on page 44).

[234] K. Pearson. The problem of the random walk. *Nature*, 72(1865):294, 1905 (cited on page 59).

[235] K. Pearson. *Tables of incomplete beta functions*. Cambridge University Press, Cambridge, England, 2nd edition, 1968 (cited on page 25).

[236] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: machine learning in python. *Journal of machine learning research*, 12:2825–2830, 2011 (cited on page 5).

[237] J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999 (cited on page 97).

[238] R Core Team. R: a language and environment for statistical computing. URL: http://www.R-project.org/. R Foundation for Statistical Computing, Vienna, Austria, 2012 (cited on page 5).

[239] J. Ramon and T. Gärtner. Expressivity versus efficiency of graph kernels. In *Proceedings of the first international workshop on mining graphs, trees and sequences*, 2003, pages 65–74 (cited on page 99).

[240] S. E. Robertson. Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of documentation*, 60(5):503–520, 2004 (cited on page 44).

[241] S. E. Robertson and K. Spärck Jones. Relevance weighting of search terms. *Journal of the american society for information science*. JASIS '76 27(3):129–146, 1976 (cited on pages 42–44).

[242] S. E. Robertson and K. Spärck Jones. Simple, proven approaches to text retrieval. (UCAM-CL-TR-356). Computer Laboratory, University of Cambridge, Dec. 1994 (cited on pages 40, 42).

[243] S. E. Robertson and S. Walker. On relevance weights with little relevance information. *ACM SIGIR forum*, 31:16–24, SI, July 1997 (cited on page 43).

[244] S. E. Robertson, S. Walker, K. Spärck Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Proceedings of the 3rd text REtrieval conference*. In TREC-3, 1994, pages 109–126 (cited on pages 11, 38, 41, 44).

[245] F. Rousseau, J. Casas-Roma, and M. Vazirgiannis. Community-preserving anonymization of social networks. *ACM transactions on knowledge discovery from data*, 2015 (cited on pages ix, 68).

[246] F. Rousseau, E. Kiagias, and M. Vazirgiannis. Text categorization as a graph classification problem. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing*. Volume 1. In ACL-IJCNLP '15. ACL, 2015, pages 1702–1712 (cited on pages ix, 97).

[247] F. Rousseau and M. Vazirgiannis. Composition of TF normalizations: new insights on scoring functions for ad hoc IR. In *Proceedings of the 36th annual international ACM SIGIR conference on research and development in information retrieval*. In SIGIR '13. ACM, 2013, pages 917–920 (cited on pages ix, 11, 38, 45, 126).

[248] F. Rousseau and M. Vazirgiannis. Graph-of-word and TW-IDF: new approach to ad hoc IR. In *Proceedings of the 22nd ACM international conference on information and knowledge management*. In CIKM '13. ACM, 2013, pages 59–68 (cited on pages ix, 31, 46, 77, 125).

[249] F. Rousseau and M. Vazirgiannis. Main core retention on graph-of-words for single-document keyword extraction. In *Proceedings of the 37th european conference on information retrieval*. In ECIR '15. Springer-Verlag, 2015, pages 382–393 (cited on pages ix, 35, 68).

[250] H. Rubenstein and J. B. Goodenough. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633, Oct. 1965 (cited on page 27).

[251] H. Saigo, S. Nowozin, T. Kadowaki, T. Kudo, and K. Tsuda. gBoost: a mathematical programming approach to graph classification and regression. *Machine learning*, 75(1):69–89, 2009 (cited on pages 99, 100, 103).

[252] T. Sakai. Statistical reform in information retrieval? *ACM SIGIR forum*, 48(1):3–12, June 2014 (cited on page 26).

[253] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information processing and management*, 24(5):513–523, 1988 (cited on pages 40–42).

[254] G. Salton, A. Wong, and C.-S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975 (cited on page 27).

[255] G. Salton, C.-S. Yang, and C. T. Yu. A theory of term importance in automatic text analysis. *Journal of the american society for information science*. JASIS '75 26(1):33–44, 1975 (cited on page 28).

[256] M. Sanderson and J. Zobel. Information retrieval system evaluation: effort, sensitivity, and reliability. In *Proceedings of the 28th annual international ACM SIGIR conference on research and development in information retrieval*. In SIGIR '05. ACM, 2005, pages 162–169 (cited on page 26).

[257] T. Sandler, J. Blitzer, P. P. Talukdar, and L. H. Ungar. Regularized learning with networks of features. In *Advances in neural information processing systems 22*. In NIPS '09. Neural Information Processing Systems, 2009, pages 1401–1408 (cited on page 30).

[258] B. Santorini. Part of speech tagging guidelines for the penn treebank project. (MS-CIS-90-47). Department of Computer and Information Science, University of Pennsylvania, 1990 (cited on page 34).

[259] R. Sarikaya, G. E. Hinton, and B. Ramabhadran. Deep belief nets for natural language call-routing. In *Proceedings of the 2011 IEEE international conference on acoustics, speech and signal processing*. In ICASSP '11, 2011, pages 5680–5683 (cited on page 13).

[260] J. Savoy. Statistical inference in retrieval effectiveness evaluation. *Information processing and management*, 33(4):495–512, July 1997 (cited on pages 22, 23).

[261] A. Schenker. Graph-theoretic techniques for web content mining. Ph.D. thesis. Tampa, FL, USA: University of South Florida, June 2003. 146 pages (cited on pages 31, 32, 99).

[262] A. Schenker, H. Bunke, M. Last, and A. Kandel. *Graph-theoretic techniques for web content mining*. Volume 62 of *Machine Perception and Artificial Intelligence*. World Scientific, 2005 (cited on page 30).

[263] N. Schluter. Centrality measures for non-contextual graph-based unsupervised single document keyword extraction. In *Actes de la 21ème conférence sur le traitement automatique des langues naturelles*. Volume 2. In TALN '14, 2014, pages 455–460 (cited on page 65).

[264] K.-M. Schneider. A comparison of event models for naive bayes anti-spam e-mail filtering. In *Proceedings of the 10th conference of the european chapter of the association for computational linguistics*. Volume 1. In EACL '03. ACL, 2003, pages 307–314 (cited on page 86).

[265] O. Schwarzkopf. The extensible drawing editor ipe. In *Proceedings of the 11th annual symposium on computational geometry*. In SCG '95, 1995, pages 410–411 (cited on page 5).

[266] F. Sebastiani. Machine learning in automated text categorization. *ACM computing surveys*, 34(1):1–47, Mar. 2002 (cited on page 12).

[267] S. B. Seidman. Network structure and minimum degree. *Social networks*, 5:269–287, 1983 (cited on page 65).

[268] P. Senellart. Extraction of information in large graphs; automatic search for synonyms. M.Sc. thesis. Louvain-la-Neuve, Belgium: Université catholique de Louvain, Aug. 2001. 17 pages (cited on page 35).

[269] N. Shervashidze. Graph kernels. 2009. URL: http://www.di.ens.fr/~shervashidze/code.html (visited on 06/12/2015) (cited on page 100).

[270] J. Sinclair. *Corpus, concordance, collocation*. Volume 1. Oxford University Press, 1991 (cited on page 29).

[271] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In *Proceedings of the 19th annual international ACM SIGIR conference on research and development in information retrieval*. In SIGIR '96. ACM, 1996, pages 21–29 (cited on pages 40, 54).

[272] A. Singhal, J. Choi, D. Hindle, D. D. Lewis, and F. Pereira. AT&t at TREC-7. In *Proceedings of the 7th text REtrieval conference*. In TREC-7, 1999, pages 239–252 (cited on pages 11, 37, 42, 44).

[273] A. Singhal, G. Salton, and C. Buckley. Length normalization in degraded text collections. Ithaca, NY, USA: Cornell University, 1995 (cited on page 42).

[274] G. Siolas and F. d'Alché-Buc. Support vector machines based on a semantic kernel for text categorization. In *Proceedings of the IEEE-INNS-ENNS international joint conference on neural networks*. Volume 5. In IJCNN '00. IEEE Computer Society, 2000, pages 205–209 (cited on page 114).

[275] F. A. Smadja. From n-grams to collocations: an evaluation of xtract. In *Proceedings of the 29th annual meeting of the association of computational linguistics*. In ACL '91. ACL, 1991, pages 279–284 (cited on page 29).

[276] F. A. Smadja. Retrieving collocational knowledge from textual corpora. an application: language generation. Ph.D. thesis. New York, NY, USA: Computer Science Department, Columbia University, Apr. 1991 (cited on page 29).

[277] F. A. Smadja. Retrieving collocations from text: xtract. *Computational linguistics*, 19(1):143–177, Mar. 1993 (cited on page 29).

[278] F. A. Smadja and K. R. McKeown. Automatically extracting and representing collocations for language generation. In *Proceedings of the 28th annual meeting on association for computational linguistics*. In ACL '90. ACL, 1990, pages 252–259 (cited on page 29).

[279] M. D. Smucker, J. Allan, and B. A. Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the 16th ACM international conference on information and knowledge management*. In CIKM '07. ACM, 2007, pages 623–632 (cited on page 26).

[280] M. M. Soares, G. Corso, and L. S. Lucena. The network of syllables in portuguese. *Physica a: statistical mechanics and its applications*, 355(2):678–684, 2005 (cited on page 30).

[281] S. S. Sonawane and P. A. Kulkarni. Graph based representation and analysis of text document: a survey of techniques. *International journal of computer applications*, 96(19):1–8, June 2014 (cited on page 29).

[282] Y. Song and D. Roth. Unsupervised sparse vector densification for short text similarity. In *Proceedings of the 2015 conference of the north american chapter of the association for computational linguistics: human language technologies*. In NAACL-Short '15. ACL, 2015, pages 1275–1280 (cited on page 115).

[283] K. Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972 (cited on pages 9, 42).

[284] K. Spärck Jones and R. G. Bates. Research on automatic indexing 1974-1976. (5464). Computer Laboratory, University of Cambridge, British Library Research and Development, 1977 (cited on pages 21, 22).

[285] S. Srivastava, D. Hovy, and E. H. Hovy. A walk-based semantically enriched tree kernel over distributed word representations. In *Proceedings of the 2013 conference on empirical methods in natural language processing*. In EMNLP '13. ACL, 2013, pages 1411–1416 (cited on pages 115, 116, 119).

[286] E. Svenonius. An experiment in index term frequency. *Journal of the american society for information science*. JASIS '72 23(2):109–121, 1972 (cited on page 42).

[287] J. A. Swets. Information retrieval systems. *Science*, 141(3577):245–250, 1963 (cited on pages 15, 17).

[288] T. Tao and C. Zhai. An exploration of proximity measures in information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval*. In SIGIR '07. ACM, 2007, pages 295–302 (cited on page 28).

[289] M. Thoma, H. Cheng, A. Gretton, J. Han, H.-P. Kriegel, A. J. Smola, L. Song, P. S. Yu, X. Yan, and K. M. Borgwardt. Near-optimal supervised feature selection among frequent subgraphs. In *Proceedings of the SIAM international conference on data mining*. In SDM '09. SIAM, 2009, pages 1076–1087 (cited on page 99).

[290] R. Thorndike. Who belongs in the family? *Psychometrika*, 18(4):267–276, Dec. 1953 (cited on page 74).

[291] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the royal statistical society. series b (methodological)*:267–288, 1996 (cited on page 91).

[292] A. N. Tikhonov and V. I. Arsenin. *Solutions of ill-posed problems*. Winston, 1977 (cited on page 91).

[293] S. Tratz and E. H. Hovy. A fast, accurate, non-projective, semantically-enriched parser. In *Proceedings of the 2011 conference on empirical methods in natural language processing*. In EMNLP '11. ACL, 2011, pages 1257–1268 (cited on pages 30, 117).

[294] G. Tsatsaronis, I. Varlamis, and K. Nørvåg. SemanticRank: ranking keywords and sentences using semantic graphs. In *Proceedings of the 23rd international conference on computational linguistics*. In COLING '10. ACL, 2010, pages 1074–1082 (cited on pages 65, 75).

[295] P. D. Turney. Learning to extract keyphrases from text. (ERB-1057). National Research Council of Canada, Institute for Information Technology, 1999 (cited on pages 13–15, 28, 74, 76).

[296] A. Turpin, Y. Tsegay, D. Hawking, and H. E. Williams. Fast generation of result snippets in web search. In *Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval*. In SIGIR '07. ACM, 2007, pages 127–134 (cited on page 14).

[297] J. Urbano, M. Marrero, and D. Martín. A comparison of the optimality of statistical significance tests for information retrieval evaluation. In *Proceedings of the 36th annual international ACM SIGIR conference on research and development in information retrieval*. In SIGIR '13. ACM, 2013, pages 925–928 (cited on page 26).

[298] K. Valle and P. Öztürk. Graph-based representations for text classification. In *Proceedings of the india-norway workshop on web concepts and technologies*. In INWWCT '11, 2011 (cited on pages 30, 48, 98).

[299] C. J. van Rijsbergen. A theoretical basis for the use of co-occurrence data in information retrieval. *Journal of documentation*, 33(2):106–119, 1977 (cited on page 28).

[300] C. J. van Rijsbergen. *Information retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition, 1979 (cited on page 17).

[301] V. N. Vapnik. *Estimation of dependences based on empirical data*. Volume 41. Springer-Verlag New York, 1982 (cited on page 94).

[302] V. N. Vapnik. Principles of risk minimization for learning theory. In *Advances in neural information processing systems 4*. In NIPS '91. Morgan Kaufmann Publishers Inc., 1991, pages 831–838 (cited on pages 90, 91).

[303] V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, 1995 (cited on page 97).

[304] V. N. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. *Automation and remote control*, 24:774–780, 1963 (cited on page 94).

[305] L. Velikovich, S. Blair-Goldensohn, K. Hannan, and R. McDonald. The viability of web-derived polarity lexicons. In *Human language technologies: the 2010 annual conference of the north american chapter of the association for computational linguistics*. In HLT '10. ACL, 2010, pages 777–785 (cited on page 30).

[306] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt. Graph kernels. *Journal of machine learning research*, 11:1201–1242, 2010 (cited on page 99).

[307] D. Vitale, P. Ferragina, and U. Scaiella. Classification of short texts by deploying topical annotations. In *Proceedings of the 34th european conference on information retrieval*. In ECIR'12. Springer-Verlag, 2012, pages 376–387 (cited on page 117).

[308] X. Wan and J. Xiao. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the 23rd national conference on artificial intelligence*. Volume 2. In AAAI '08. AAAI Press, 2008, pages 855–860 (cited on page 64).

[309] X. Wan and J. Xiao. Exploiting neighborhood knowledge for single document summarization and keyphrase extraction. *ACM transactions on information systems*, 28(2):8:1–8:34, June 2010 (cited on page 64).

[310] X. Wan, J. Yang, and J. Xiao. Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction. In *Proceedings of the 45th annual meeting on association for computational linguistics*. In ACL '07. ACL, 2007, pages 552–559 (cited on page 30).

[311] F. Wang, Z. Wang, Z. Li, and J.-R. Wen. Concept-based short text classification and ranking. In *Proceedings of the 23rd ACM international conference on information and knowledge management*. In CIKM '14. ACM, 2014, pages 1069–1078 (cited on page 28).

[312] R. Wang, W. Liu, and C. McDonald. Corpus-independent generic keyphrase extraction using word embedding vectors. In *Proceedings of the workshop on deep learning for web search and data mining held in conjunction with the 8th ACM international conference on web search and data mining*. In DL-WSDM '15, 2015, page 39 (cited on page 64).

[313] T.-Y. Wang and H.-M. Chiang. Fuzzy support vector machine for multiclass text categorization. *Information processing and management*, 43(4):914–929, July 2007 (cited on page 21).

[314] W. Wang, D. B. Do, and X. Lin. Term graph model for text classification. In *Proceedings of the 1st international conference on advanced data mining and applications*. In ADMA '05. Springer-Verlag, 2005, pages 19–30 (cited on page 30).

[315] S. Wasserman and K. Faust. *Social network analysis: methods and applications*. Volume 8. Cambridge University Press, New York, NY, USA, 1994 (cited on pages 61, 67).

[316] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998 (cited on pages 35, 61).

[317] H. Wickham. *Ggplot2: elegant graphics for data analysis*. Springer New York, 2009 (cited on page 5).

[318] D. Widdows and B. Dorow. A graph model for unsupervised lexical acquisition. In *Proceedings of the 19th international conference on computational linguistics*. Volume 1. In COLING '02. ACL, 2002, pages 1–7 (cited on page 30).

[319] J. M. Wiebe, T. Wilson, and C. Cardie. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2):165–210, 2005 (cited on page 117).

[320] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning. KEA: practical automatic keyphrase extraction. In *Proceedings of the 4th ACM conference on digital libraries*. In DL '99, 1999, pages 254–255 (cited on pages 13–15, 28, 74, 76).

[321] Z. Xie. Centrality measures in text mining: prediction of noun phrases that appear in abstracts. In *Proceedings of the ACL student research workshop*. In ACLstudent '05. ACL, 2005, pages 103–108 (cited on page 29).

[322] X. Yan and J. Han. Gspan: graph-based substructure pattern mining. In *Proceedings of the 2nd IEEE international conference on data mining*. In ICDM '02. IEEE Computer Society, 2002, pages 721–724 (cited on pages 98, 100).

[323] Y. Yang. Expert network: effective and efficient learning from human decisions in text categorization and retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on research and development in information retrieval*. In SIGIR '94. ACM, 1994, pages 13–22 (cited on page 13).

[324] Y. Yang and C. G. Chute. A linear least squares fit mapping method for information retrieval from natural language texts. In *Proceedings of the 14th international conference on computational linguistics*. Volume 2. In COLING '92. ACL, 1992, pages 447–453 (cited on page 13).

[325] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on research and development in information retrieval*. In SIGIR '99. ACM, 1999, pages 42–49 (cited on pages 22, 23).

[326] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the 14th international conference on machine learning*. In ICML '97. Morgan Kaufmann Publishers Inc., 1997, pages 412–420 (cited on page 111).

[327] W. Yin and H. Schütze. An exploration of embeddings for generalized phrases. In *Proceedings of the ACL student research workshop*. In ACLstudent '14. ACL, 2014, pages 41–47 (cited on page 121).

[328] M. Yu and M. Dredze. Learning composition models for phrase embeddings. *Transactions of the association for computational linguistics*, 3:227–242, 2015 (cited on page 121).

[329] D. Zelenko, C. Aone, and A. Richardella. Kernel methods for relation extraction. *Journal of machine learning research*, 3:1083–1106, 2003 (cited on page 116).

[330] H. Zha. Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering. In *Proceedings of the 25th annual international ACM SIGIR conference on research and development in information retrieval*. In SIGIR '02. ACM, 2002, pages 113–120 (cited on page 30).

[331] C. Zhai and J. D. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval*. In SIGIR '01. ACM, 2001, pages 334–342 (cited on page 11).

[332] T. Zhang and F. J. Oles. Text categorization based on regularized linear classification methods. *Information retrieval*, 4(1):5–31, 2001 (cited on page 13).

[333] M. Zhu, Z. Cai, and Q. Cai. Automatic keywords extraction of chinese document using small world structure. In *Proceedings of the 2003 international conference on natural language processing and knowledge engineering*. In ICNLPKE '03. IEEE Computer Society, 2003, pages 438–443 (cited on pages 31, 64).

[334] J. Zobel and A. Moffat. Exploring the similarity space. *ACM SIGIR forum*, 32(1):18–34, Apr. 1998 (cited on page 45).

[335] J. Zobel and A. Moffat. Inverted files for text search engines. *ACM computing surveys*, 38(2), July 2006 (cited on page 12).

NOTATION

LINEAR ALGEBRA

SYMBOL	MEANING
$\|\cdot\|_0$	L^0-norm, i. e. number of non-zero values of a vector
$\|\cdot\|_1$, $\lvert\cdot\rvert$	L^1-norm, i. e. length of a vector
$\|\cdot\|_2$, $\|\cdot\|$	L^2-norm, i. e. Euclidean length of a vector
$\|\cdot\|_\infty$	L^∞-norm, i. e. maximum value of a vector

MACHINE LEARNING

SYMBOL	MEANING
c	class label, a. k. a. category
$\boldsymbol{\mathcal{C}}$	set of all class labels
y	binary class label $\in \{-1, +1\}$
$\boldsymbol{\mathcal{Y}}$	set of all binary class labels $\in \{-1, +1\}$
p	number of features
$\boldsymbol{x}$	example, i. e. a p-dimensional feature vector
$\boldsymbol{\mathcal{X}}$	set of all examples
N	total number of training examples
$\boldsymbol{\theta}$	feature weights
b	intercept
$[\cdot]_+$	$\max(0, \cdot)$

TEXT MINING

SYMBOL	MEANING
$\boldsymbol{d}$	document
$\boldsymbol{\mathcal{D}}$	collection: set of all documents
N	collection size: number of documents in $\boldsymbol{\mathcal{D}}$
$\lvert\boldsymbol{d}\rvert$	document length: number of terms in a document
L	collection length: sum of all document lengths of $\boldsymbol{\mathcal{D}}$
w	word
t	term: a processed word
$\boldsymbol{\mathcal{V}}$	vocabulary (a. k. a. lexicon or dictionary): a set of terms
n	number of unique terms in a document
$tf(t, \boldsymbol{d})$	term frequency: number of occurrences of t in $\boldsymbol{d}$
$cf(t)$	collection frequency: number of occurrences of t in $\boldsymbol{\mathcal{D}}$
$df(t)$	document frequency: number of documents with t
$\boldsymbol{q}$	query
$\hat{r}(\boldsymbol{d}, \boldsymbol{q})$	scoring function: estimate of relevance r of $\boldsymbol{d}$ w. r. t. $\boldsymbol{q}$

ACRONYMS

Acc	Accuracy ... 16
AP	Average Precision ... 18
AUC	Area Under the Curve ... 18
BOW	Bag-Of-Words ... 8
CDF	Cumulative Distribution Function ... 23
CK	Composition Kernel ... 116
CS	Computer Science ... 8
DCV	Document Cut-off Value ... 19
DET	Detection Error Trade-off ... 19
DFS	Depth First Search ... 101
DTK	Dependency Tree Kernel ... 114
F_β	F-measure ... 17
F_1	F1-score ... 18
FN	False Negative ... 15
FP	False Positive ... 15
IDF	Inverse Document Frequency ... 9
i.i.d.	independent and identically distributed ... 93
IG	Information Gain ... 111
IR	Information Retrieval ... 3
KKT	Karush-Kuhn-Tucker ... 96
kNN	k-Nearest Neighbors ... 13
KwE	Keyword Extraction ... 3
LLSF	Linear Least Square Fit ... 13
LOWESS	Locally Weighted Scatterplot Smoothing ... 54
LR	Logistic Regression ... 13
LSI	Latent Semantic Indexing ... 27
MAP	Mean Average Precision ... 20
MI	Mutual Information ... 111
ML	Machine Learning ... 3
MLE	Maximum Likelihood Estimate ... 9
NB	Naive Bayes ... 9
NLP	Natural Language Processing ... 3
P	Precision ... 17
PK	Product Kernel ... 116
P/R	Precision/Recall ... 18
P@10	Precision at 10 ... 19

POS	Part-Of-Speech	15
QF	Query Frequency	38
R	Recall	17
ROC	Receiver Operating Characteristic	19
SERP	Search Engine Results Page	10
SSTK	Semantic Syntactic Tree Kernel	114
SPTK	Smoothing Partial Tree Kernel	114
SVD	Singular Value Decomposition	27
SVM	Support Vector Machine	13
TC	Text Categorization	3
TF	Term Frequency	9
TM	Text Mining	3
TN	True Negative	15
TP	True Positive	15
TREC	Text REtrieval Conference	19
VSM	Vector Space Model	27
VTK	Vector Tree Kernel	115

COLOPHON

This document was typeset in LaTeX using the typographical look-and-feel `classicthesis`. Most of the graphics in this dissertation are generated using the `Ipe` extensible drawing editor and the R `ggplot2` library. The bibliography is typeset using `biblatex`.