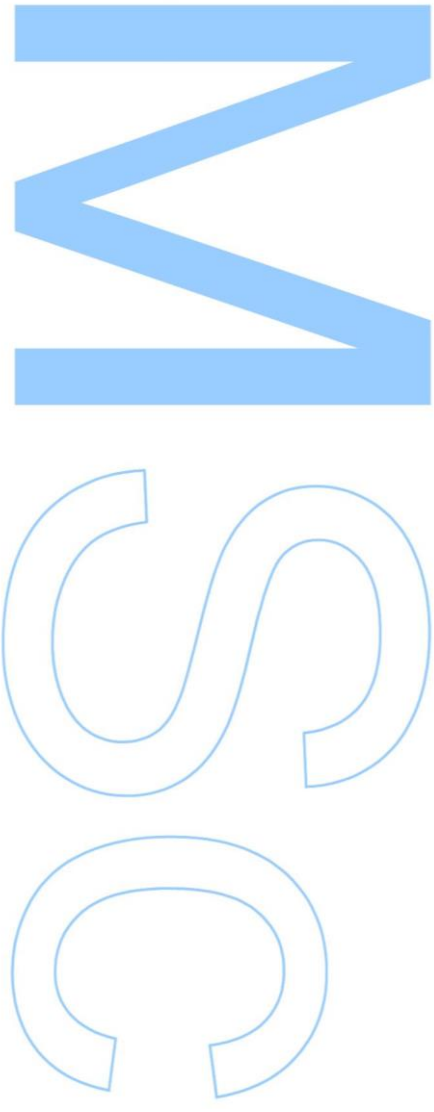
Identifying Topic Relevant Hashtags in Twitter Streams

Filipe Daniel Marques Figueiredo
Master's Degree in Computer Science
Computer Science Department
2018

Supervisor
Alípio Mário Guedes Jorge, Associate Professor, Faculty of Sciences,
University of Porto



U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____

Abstract

Hashtags have become a crucial social media tool. The categorization of posts in a simple and informal manner stimulates the dissemination of content through the web. At the same time, it enables users to find messages within a specific topic of their interest. However, the flexibility provided to the user to apply any hashtag carries some problems. Equivalent expressions, like synonyms, are handled like entirely different words, while the same hashtag may refer to distinct topics. Also, many hashtags are dynamic in the sense their meaning and connections with different subjects change through time and location. This factors may hinder content discovery, specially when discussing less popular subjects. One way to overcome this problem is to provide utilities to identify relevant hashtags. Some research in hashtag recommendation in Twitter has been conducted over recent years but with greater focus on proposing hashtags for new posts instead of for a topic in general. Additionally, most of the current approaches rely on databases which require time to be assembled and rigorous maintenance to keep updated.

The approach we propose for the identification of topic relevant hashtags is the development of a method to search Twitter, in real time, for hashtags relevant to a topic and represent them in a graph. For this task, we first retrieve tweets within some degrees of connection with the subject. Next, we employ Latent Dirichlet Allocation and Support Vector Machines, to classify tweets and collect their hashtags relevant to the subject. Finally, we use these hashtags to assemble a network of relations that can be used to deepen content retrieval on the original subject. This approach takes into account factors such as the popularity of the hashtags and current meaning.

Furthermore, we analyze the proposed algorithm, both qualitatively and quantitatively, and compare it with past approaches, in order to evaluate its performance. The outcomes of our method are usually equal or superior to the available alternatives, in relation to the number of returned hashtags, and current relevance to the topic. However, our process is by default significantly slower than the existing alternatives.

keywords: Text Mining, Topic Modeling, Latent Dirichlet Allocation, Support Vector Machines, Twitter, Hashtags

Resumo

As hashtags tornaram-se numa ferramenta crucial nas redes sociais. A categorização de posts de forma simples e informal estimula a disseminação de conteúdo através da internet. Ao mesmo tempo, permite que os utilizadores encontrem mensagens acerca de um tópico específico do seu interesse. No entanto, a flexibilidade oferecida ao utilizador para aplicar qualquer hashtag acarreta alguns problemas. Expressões equivalentes, como sinónimos, são tratadas como palavras totalmente diferentes, enquanto que a mesma hashtag pode ser associada a tópicos distintos. Além disso, muitas hashtags são dinâmicas, no sentido em que o seu significado e relações com diferentes assuntos mudam com o tempo e a localização. Esses fatores podem dificultar a descoberta de conteúdo, especialmente quando se discutem assuntos menos populares. Uma forma de ultrapassar este problema é a disponibilização de ferramentas para identificar hashtags relevantes a um tópico. Alguma pesquisa em recomendação de hashtag no Twitter foi levada a cabo nos últimos anos, mas com maior foco em propor hashtags para novos posts em vez de para um tópico em geral. Além disso, a maioria das abordagens atuais depende de bases de dados que necessitam de tempo para ser recolhidas e de manutenção rigorosa para que se mantenham atualizadas.

A abordagem que propomos para identificar hashtags relevantes para um tópico passa pelo desenvolvimento de um algoritmo para pesquisar no Twitter, em tempo real, por hashtags relevantes para um tópico e representá-las num grafo. Para essa tarefa, primeiro recolhemos tweets dentro de alguns graus de conexão com o assunto. Em seguida, aplicamos Alocação Latente de Dirichlet e Máquinas de Vetores de Suporte, para classificar tweets e recolher as hashtags relevantes para o tema. Finalmente, usamos essas hashtags para montar uma rede de relações que pode ser utilizada para aprofundar a recuperação de conteúdo sobre o tópico original. Essa abordagem tem em conta fatores como a popularidade das hashtags e o significado atual.

Além disso, analisamos o algoritmo proposto, tanto qualitativamente como quantitativamente, comparando-o com abordagens passadas, de forma a avaliar o seu desempenho. Os resultados do nosso método são geralmente iguais ou superiores às alternativas disponíveis, tanto em relação ao número de hashtags retornadas como na sua relevância atual para o tópico. No entanto, o nosso processo é, por defeito, significativamente mais lento do que as alternativas existentes.

palavras chave: Mineração de Texto, Modelos de Tópicos, Alocação Latente de Dirichlet, Máquinas de Vetores de Suporte, Twitter, Hashtags

Acknowledgments

I want to express my gratitude towards Prof. Alípio Jorge for accepting to be my mentor and thesis supervisor, for trusting in my capabilities to take part in his research and for all the guidance and support provided.

Furthermore, I am grateful for my family, who always supported me through my academic and personal life, specially for my parents Jorge and Anabela, my sister Filipa, my grandparents and for my girlfriend Cláudia for always standing by my side.

Finally, I want to acknowledge all my friends for the joyful moments shared over the years, in particular Pedro, Ricardo, Duarte, João and Gonçalo for their companionship and helpfulness through my university years.

I would never get this far without them.

The research leading to these results was conducted as part of the RECAP preterm Project which received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 733280.

Contents

List of Tables

List of Figures

List of Algorithms

Acronyms

CTM Correlated Topic Models

DF Data Frame

DTM Document Term Matrix

IDF Inverse Document Frequency

IR Information Retrieval

L2R Learning-to-Rank

LDA Latent Dirichlet Allocation

NLP Natural-Language Processing

pLSA Probabilistic Latent Semantic Analysis

POS Part-of-Speech-Tagging

SVM Support Vector Machine

TF-IDF Term Frequency - Inverse Document Frequency

TM Topic Model

UTF-8 8-bit Unicode Transformation Format

Chapter 1

Introduction

Twitter has become one of the most popular micro-blogging services in the world [24] with over 300 million active users every month [25]. The social network allows its users to communicate by posting short messages called "tweets", which consist in pieces of text up to 280 characters long (originally only 140 characters). Due to the massive amount of users, around 8,000 tweets are posted every second [74], which, despite the relatively small size of the messages, results in an astounding amount of information spread through the internet every day. However, accessing the relevant pieces of this information is not necessarily a straightforward task.

The original motivation for our work was to retrieve data from posts on social media in order to support a project (RECAP[1]) whose objective was to study physical conditions, on any stage of life, that could be related to a premature birth. Obstacles in determining the most advantageous manner of querying Twitter for collecting this information lead to the development of a tool for hashtag recommendation. This system was then improved and adapted to perform appropriately for virtually any other topic discussed in the platform.

Most of the functionality and tools of Twitter that make it so important in public communication today were user-led innovations, later integrated into the service itself. One of these distinctive components is the hashtag, mostly associated with Twitter but also adopted by other social networks and communication platforms. Because of Twitter's popularity, it became burdensome to search for relevant information about a specific topic on the service. This made it frustrating for users, specially ones with small follower bases, to openly engage in discussions in the platform. For that reason, users started applying dynamic, user-generated tagging signalized by a "#" to their tweets as a way to make them publicly available, categorized and, consequently, easier to encounter by other users interested in that same topic.

While the current hashtag implementation in Twitter brings many advantages for its users, because of its easiness of use, there are still some issues when searching for content of a specific topic. Since anyone can use any word or expression as a hashtag, limited only by the poster's perspective and imagination and by Twitter's imposed 280 characters cap, it may be difficult for

[1] https://recap-preterm.eu/

a user to find every tweet relevant to his search. Hashtags are often ambiguous and some of the concepts expressed may be more popularly represented by a synonym, an acronym or even an analogue expression, which are handled by Twitter like entirely different topics. Furthermore, an hashtag may simply express a sub-topic or a super-topic of a more relevant subject. On the other hand, hashtags are dynamic in a sense that their popularity and meaning are constantly fluctuating. Since only one hashtag can be searched at a time, it may not be immediately clear which one is the best to search for a certain topic, specially when dealing with less popular subjects.

This situations are embedded in the principles of the hashtag categorization mechanism and, as a consequence, are analogous in every other social network and services making use of it. Nonetheless, over recent years, a few works attempted to minimize these shortcomings in Twitter through techniques of hashtag recommendation [16][26][43][47][84][85].

However, the previous research focuses primarily in proposing hashtags for new posts being made, taking into consideration an entire tweet, while mostly neglecting hashtag recommendation towards a topic in general. Additionally, most of the current approaches are fully dependent on large-scale databases and, while this brings advantages in response time, on social media everything is rapidly changing, making these data sets require an high degree of maintenance in order to keep providing relevant results. In this manner, there is interest in studying hashtag recommendation taking into consideration the evolution of communication patterns, that is, on a streaming environment instead of a static database.

1.1 Goals and Methodology

The objective of this thesis is to develop a new methodology and a tool to efficiently identify and collect hashtags related to a topic in Twitter streams, along with their relations, and represent them in an understandable visual manner, assisting people to reach more information and communicate more effectively.

In this document, we present TORHID (Topic Relevant Hashtag Identification), a technique to identify and retrieve relevant hashtags for a topic in Twitter streams. Firstly, we describe an algorithm to collect tweets and retrieve hashtags within some degree of connection with the subject. The relations among the hashtags are also determined and depicted through a graph. We subsequently improve on this concept through the application of topic modeling, to automatically disambiguate tweets and their respective hashtags, filtering the non relevant ones, thereby pruning the network. More specifically, we employ Latent Dirichlet Allocation and support vector machines for the classification tasks. In this manner, we present a method that achieves an assortment of hashtags relevant to the original topic and the relationships among them in a network represented through a weighted, directed graph.

We proceed to implement our method in `R` and compare it with available alternatives, both qualitatively and quantitatively, in order to validate its precision and usefulness. We also present

a new evaluation metric when performing qualitative analysis that takes into consideration the temporal context of the results. Except for being slower in response time, our results are usually as good or better than previous techniques. We later improve the usability of our implementation, through the addition of a user interface, while attempting to minimize its run times, at a possible expense of accuracy. Finally, we demonstrate its versatility through a real world example.

1.2 Thesis outline

This chapter presented the motivation to this thesis, our objectives and contributions. The remainder of this document is structured as follows:

Chapter 2 - Basic Concepts presents the foundations of our work, defining the terminology and introducing the essential background concepts and theories applied through the remaining chapters;

Chapter 3 - Related Work discusses the past literature and contributions on the topic of this thesis, designating the respective limitations;

Chapter 4 - Design and Implementation describes the contributions of our work, explaining our theory and implementation;

Chapter 5 - Evaluation defines the experiments and metrics used to validate our method, in addition to a comparison with existing approaches;

Chapter 6 - Usability Improvement presents the development of a web application and some experimentation with our implementation;

Chapter 7 - Case Study demonstrates a possible application of our method with a real life example;

Chapter 8 - Conclusion summarizes the presented material and presents perspectives of future work;

Chapter 2

Background

The purpose of this chapter is to cover the theories and concepts required to fully understand the rest of this document. First we explain some of the terminology employed and then we cover the tools and algorithms used to develop our method.

2.1 Terminology

In this section, we introduce the reader to some of the important terminologies used through this thesis. To address the components and features of Twitter, the following terms will be frequently employed:

Twitter Status/Post/Tweet - a short message posted to Twitter, limited to a length of 280 characters. The content of tweets may include images, videos, URLs, emojies, mentions of other users, polls or simply plain text, while their meta data incorporates information like the ID, the location and the time of the post, the handle of the user who posted the status and the number of likes and retweets. Unless the account is set as private, all posts are public by default.

User - everyone with a Twitter account, that can post tweets and interact with other people. Every user is identified by a unique twitter handle which consists of the respective username preceded by an "*at*" sign (@), like in "*@user*".

Retweet - the repost of an already existing tweet from a different user, with the intention of sharing its content. In order to differentiate them and give credit to the primary posts, retweets are marked with a "RT" tag and a mention of the original poster as in "*RT @username*".

Mentions and Replies - posts that refer directly to another Twitter user and are intended to start or continue conversations. Both kinds of posts are public by default and marked with the handles of the recipients.

Follower - a user who subscribes to the feed of other users, thereby receiving updates on their future tweets. In this manner, the followers of an account represent its main audience. Unlike most social networks, following is not necessarily mutual on Twitter.

Feed and Timeline - a chronologically ordered list of tweets that is automatically updated with new content. Each account has a public timeline of tweets posted by itself and a private timeline with posts from users they follow.

Hashtag - a user-defined tag for categorizing messages in Twitter, signaled by a keyword preceded by an octothorpe (#) as in *#hashtag*. Tweets with the same hashtags can be listed together. Hashtags are used to group and spread conversations about a specific topic or event in a feed.

Trending Topics - one of the most popular hashtags at a given moment in Twitter. This hashtags are publicly listed to facilitate their discoverability and allow for the contributions of more users to the conversation. Trending topics highly variate through time and may be different according to the user' location.

In addition, the following nomenclatures were employed to simplify the explanation of some concepts and procedures:

Seed - a keyword that can accurately represent a given topic. In our work, seeds are hashtags characterizing the original topic for which we collect pertinent tweets and recommend related hashtags.

2.2 Theoretical Concepts

The purpose of this section is to cover the elemental theoretical concepts required to fully understand the methodology of our approach.

2.2.1 Support Vector Machines

In data science fields associated with machine learning, support vector machines are a set of methods for supervised learning, used for both classification and regression analysis. Being a supervised learning model, SVMs always require the training data to be labeled.

Provided with labeled training data, that is, a training set with every instance marked as related to one of two classes, a support vector machine builds a non-probabilistic binary classifier that assigns new cases to either one of those categories.

The support vector learning machine was developed by Vapnik et al. [78][66] to implement proposed principles from statistical learning [79], according to which *learning* is fundamentally

the same as estimating an accurate function from a set of examples (training set). In this manner, the task of a leaning machine is to determine the function from a given assortment of functions that minimizes the risk of being different from the actual unknown function.

In essence, SVMs operate through the representation of the instances as points in a n-dimensional space, attempting to maximize the margin between the closest points of both classes (support vectors), and the definition of an optimal separating hyper-plane between those points. For better understanding, these concepts are illustrated in Figure 2.1. In this manner, a margin can be defined as the minimal distance of an example to a decision surface. In cases where some instances are placed in the incorrect side of the discriminant margin, their weights are reduced in order to decrease their influence in the final function. The new cases are then mapped into the described space and, according to the side of the gap they get placed into, are classified as belonging to a category or the other. The fundamental principle of support vector learning is the proven insight that the risk of miscalculations is minimized when the discriminant margin is maximized [46].

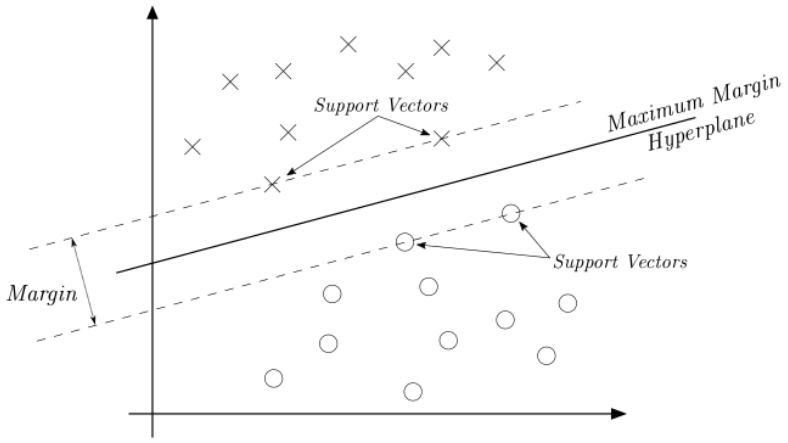


Figure 2.1: Hyperplane, Margins and Support Vectors of a SVM.

In addition to performing linear classification, in cases where it is impossible to determine a linear separator, support vector machines are able to perform non-linear classification through the application of kernel functions, where the data points are implicitly projected into higher-dimensional feature spaces with the purpose of making them linearly separable. Figure 2.2 illustrates an example of the use of kernels in support vector machines where a linear separation of the classes would usually be impossible.

2.2.1.1 Formalization of Hyper-plane Classifiers

The function of a classifier is to discover a rule that correctly assigns a new instance to one of several given categories, based on previous external knowledge.

In a simple case with l training examples $\{x_i, y_i\}$, $i = 1, \ldots, l$, each one with d inputs ($x_i \in \mathbb{R}^d$),

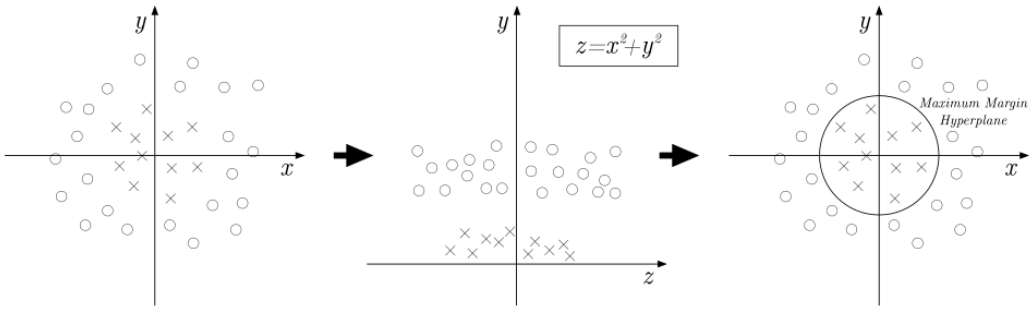


Figure 2.2: Example of SVM classification through the use of kernels.

and a class label with only two possible values ($y_i \in \{-1, +1\}$), one possible formalization of the problem is to estimate the function $f : \mathbb{R}^N \to \{-1, +1\}$. In this case, all hyper-planes in $\mathbb{R}^d$ are parameterized by a vector w and a constant b and can be expressed in the equation:

$$w \cdot x + b = 0 \tag{2.1}$$

. Given a hyper-plane (w, b) that properly divides the training data, we get the corresponding decision function:

$$f(x) = sign(w \cdot x + b) \tag{2.2}$$

.

Considering that every hyper-plane represented by (w, b) is equally expressed by all pairs $\{\lambda w, \lambda b\}$, for $\lambda \in \mathbb{R}^+$, we define the acceptable hyper-plane as the one which separates the data by at least 1. That is, we only consider those that satisfy the conditions:

$$x_i \cdot w + b \geq +1 \quad \text{when} \quad y_i = +1 \tag{2.3}$$

$$x_i \cdot w + b \leq +1 \quad \text{when} \quad y_i = -1 \tag{2.4}$$

or, more compactly:

$$y_i(x_i \cdot w + b \geq 1), \quad \forall i \tag{2.5}$$

. Which means that every hyper-plane has a functional distance ≥ 1. To calculate the euclidean distance (margin) from the hyper-plane to a data point, we normalize the function by the magnitude of w, that is, the distance is:

$$d((w, b), x_i) = \frac{y_i(x_i \cdot w + b)}{\|w\|} \geq \frac{1}{\|w\|} \tag{2.6}$$

.

Since we want the hyper-plane that maximizes the margin, that is, the distance to the closest data points, we need to minimize $\|w\|$ (Quadratic Programming Problem). This task is usually

achieved through the application of Lagrange multipliers [10], transforming the problem into:

$$\text{minimize:} \quad w(\alpha) = -\sum_{i=1}^{l} \alpha_1 + \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j (x_i \cdot x_j) \tag{2.7}$$

$$\text{subject to:} \quad \sum_{i=1}^{l} y_i \alpha_1 = 0 \tag{2.8}$$

$$0 \leq \alpha_i \leq C, \quad \forall i \tag{2.9}$$

where the vector of the l non-negative Lagrange multipliers to be determined is represented by α, and C is a constant. Essentially, if $C = \infty$, the optimal hyper-plane divides the classes completely [9], but when C is finite, the classifier allows for a soft-margin [12], that is, allows for a margin error. The matrix $(H)_{ij} = y_i y_j (x_i \cdot x_j)$ can be defined to introduce more compact notation:

$$\text{minimize:} \quad w(\alpha) = -\alpha^T 1 + \frac{1}{2}\alpha^T H \alpha \tag{2.10}$$

$$\text{subject to:} \quad \alpha^T y = 0 \tag{2.11}$$

$$0 \leq \alpha \leq C1 \tag{2.12}$$

. By deriving these equations, the optimal hyper-plane can be written as:

$$w = \sum_i \alpha_i y_i x_i \tag{2.13}$$

. Since only the support vectors are considered, that is, only the data points closest to the margin contribute to w, it can also be shown that:

$$\alpha_i (y_i (w \cdot x_i + b) - 1) = 0, \quad \forall i \tag{2.14}$$

. In order to specify the hyper-plane, after discovering the optimal α to construct w, it is still required to determine b. To perform this task, any positive and negative support vector is considered as x^+ and x^-:

$$(w \cdot x^+ + b) = +1 \tag{2.15}$$

$$(w \cdot x^- + b) = -1 \tag{2.16}$$

what is translated to:

$$b = -\frac{1}{2}(w \cdot x^+ + w \cdot x^-) \tag{2.17}$$

.

2.2.1.2 Kernels

The use of linear functions for the classifier gives the impression of being a limiting factor to support vector machines since many data sets cannot be linearly separable. However, it is possible to have linear models with a set of nonlinear decision functions through the application of the

kernel trick [12], that essentially consists in performing a type of *preprocessing* to the training data, transforming the problem into finding a normal hyper-plane.

Employing the kernel trick for support vector machines works through the creation of a non-linear mapping $\phi : \mathbb{R}^N \to F$, which fits the original input space into a new feature space F, usually of higher dimensionality. The maximum margin hyper-plane is now just required to separate this transformed data $(\phi(x_1), y_1)$. Usually, if $\phi(x)$ casts the input vector into a high enough space, the training data becomes separable [9].

Given a mapping $z = \phi(x)$ and using it to replace all occurrences of x with $\phi(x)$, the quadratic problem is still transformed into:

$$\text{minimize:} \quad w(\alpha) = -\alpha^T 1 + \frac{1}{2}\alpha^T H \alpha \tag{2.18}$$

in resemblance with (2.10). However, the matrix $(H)_{ij} = y_i y_j (x_i \cdot x_j)$ is converted to $(H)_{ij} = y_i y_j (\phi(x_i) \cdot \phi(x_j))$. The equation (2.13) becomes:

$$w = \sum_i \alpha_i y_i \phi(x_i) \tag{2.19}$$

. And equation (2.2) also becomes:

$$f(x)) = sign(w \cdot \phi(x) + b) \tag{2.20}$$

$$= sign\left(\left[\sum_i \alpha_i y_i \phi(x_i)\right] \cdot \phi(x) + b\right) \tag{2.21}$$

$$= sign\left(\sum_i \alpha_i y_i (\phi(x_i) \cdot \phi(x)) + b\right) \tag{2.22}$$

.

One important aspect of these transformations is that wherever there is a $\phi(x_i)$ in a equation, it is always in a dot product with another $\phi(x_j)$. As a result, if there exists a kernel function for the dot product in the higher dimensional feature space,

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j) \tag{2.23}$$

the use of that kernel would only be required in the training algorithm and it would never be necessary to even explicitly know the mapping function $z = \phi(x)$. This means that, in reality, the optimization matrix would simply be represented as $(H)_{ij} = y_i y_j (K(x_i, x_j))$ and the classifier as $f(x) = sign\left(\sum_i \alpha_i y_i (K(x_i, x)) + b\right)$. Once the kernel is applied, the process to find the optimal hyper-plane proceeds conventionally. Only in the original feature space, the delineation of the data will be curved and possibly non-continuous instead of linear.

The Mercer's condition provides the mathematical properties to verify if a prospective kernel K is a dot product in some feature space [78], but not to construct the mapping function itself. Finding the best kernel function for a data set has been subject of research [64] [75] and it was found that different kernels may give similar classification accuracy and use similar support vectors [65], suggesting that each problem may be characterized by a fixed support data.

2.2.2 Latent Dirichlet Allocation

In data science fields associated with natural-language processing and text-mining, Latent Dirichlet Allocation is a generative probabilistic model used to group unclassified sets of discrete data into a number of previously defined categories, most often used to analyze the topics in text corpora and organize the documents into the respective classes. When applied to a corpus, Latent Dirichlet Allocation considers each document as a combination of various topics and attributes to them the probability of the presence of each word. It is similar to probabilistic latent semantic analysis, except that Latent Dirichlet Allocation assumes that each document does not incorporate the entire set of topics and each topic does only employ a small set of words [23].

Latent Dirichlet allocation is a three-level hierarchical Bayesian model first introduced by Blei et al. [8] to analyze text corpora and became one of the most popular techniques in text-mining [81]. Some extensions to the method were later developed like dynamic topic models [7], hierarchical Dirichlet processes [76] and correlated topic models [6]. Other researchers have also successfully adapted Latent Dirichlet Allocation to images [33] and video analysis [80].

Essentially, Latent Dirichlet Allocation assumes that when each document from a corpus is being composed, the author firstly decides the total number of words and the topic composition, according to a Dirichlet distribution, and then populates the document with words from those topics. The model attempts to backtrack from the final set of documents to find the set of topics that are likely to have been used to generate the collection.

Given a corpus to find the topic representation of each document and a given number of topics to determine (either an informed estimate or trial and error), the algorithm begins to semi-randomly assign every word of every document to a temporary topic, according to a Dirichlet distribution (if a word appears twice, different topics may be assigned to each occurrence). Stopwords are removed and not assigned to any of the topics. This initial step provides a randomized topic representation of every document and an inaccurate word distribution for every topic. Next, the algorithm checks every word and updates the topic assignment based on the proportion of words in that document that are currently assigned to each topic (how prevalent is each topic in the document) and the proportion of assignments to each topic over all documents that come from the word (how frequent is the word across topics). In this step, it is assumed that all topic assignments except for the current word are correct and its assignment is updated using the proposed model of how documents are generated. Through the repetition of the last procedure, the topic assignment gets improved until becoming acceptable and can be used to estimate the topic composition of the documents and the word composition of the topics.

Latent Dirichlet allocation does not consider that the topics are defined either semantically or epistemologically, but simply characterized by the calculated probability of co-occurrence of each set of words. However, a lexical term may be included in more than one topic, with different probabilities. It is an unsupervised machine learning technique since the analysis is performed only according to word frequencies and, as a consequence, the data doesn't need to be labeled.

Being a generative model, avoids making any initial assumptions about how the text relates to the categories.

2.2.2.1 Formalization of the Generative Process

To formalize Latent Dirichlet Allocation, we start by defining the generative process with more detail as follows:

1. For each document d in a corpus D:
   (a) Select a total number of words $N \sim Poison(\xi)$
   (b) Select a topic distribution, $\theta_d \sim Dir(\alpha)$, where $Dir(\cdot)$ is drawn from a uniform Dirichlet distribution with a scaling parameter α
   (c) For each of the N words w_n:
      i. Select a specific topic $z_n \sim Multinomial(\theta)$
      ii. Select a word w_n from $p(w_n|z_n, \beta)$, a multinomial probability conditioned on the topic z_n

A selection of a k-dimensional Dirichlet distribution returns a random variable θ which can take values in the $(k-1)$-simplex and has the following probability density:

$$p(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\prod_{i=1}^{k} \Gamma\alpha_i} \theta_1^{\alpha_1 - 1} \cdots \theta_k^{\alpha_k - 1} \tag{2.24}$$

where α is the parameter of the Dirichlet distribution representing a k-vector with components $a_i > 0$ and $\Gamma(x)$ is the gamma function.

Given the parameter β, a $k \times V$ word probability matrix for each topic k and each term V, where $\beta_{ij} = p(w^j = 1|p(z^i = 1$, the joint distribution of a topic mixture θ, a set of N topics z and a set of N words w is given by:

$$p(\theta, z, w|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta) p(w_n|z_n, \beta) \tag{2.25}$$

where $p(z_n|\theta)$ is θ_i for the unique i such that $z_n^i = 1$.

The marginal distribution of a document is obtained by integrating the Equation (2.25) over θ and summing over z:

$$p(w|\alpha, \beta) = \int p(\theta|\alpha) \left(\prod_{n=1}^{N} \sum_{z_n} p(z_n|\theta) p(w_n|z_n, \beta) \right) d\theta \tag{2.26}$$

.

Finally, we obtain the probability of a corpus by calculating the product of the marginal probabilities of every single document:

$$p(D|\alpha,\beta) = \prod_{d=1}^{M} \int p(\theta_d|\alpha) \left(\prod_{n=1}^{N_d} \sum_{z_{dn}} p(z_{dn}|\theta_d) p(w_{dn}|z_{dn},\beta) \right) d\theta_d \tag{2.27}$$

Where the parameters α and β are only sampled once when the corpus is being generated, the variables θ_d are sampled once per document and the variables w_{dn} and z_{dn} are sampled once for every word.

2.2.2.2 Inference

The central inferential problem that needs to be solved in order to use Latent Dirichlet Allocation is computing the posterior distribution of the latent variables given a document:

$$p(\theta, z|w,\alpha,\beta) = \frac{p(\theta, z, w|\alpha,\beta)}{p(w|\alpha,\beta)} \tag{2.28}$$

Since this distribution is intractable to compute, in order to normalize the distribution, the latent variables are marginalized and Equation (2.26) is rewritten in terms of the model parameters:

$$p(w|\alpha,\beta) = \frac{\Gamma(\sum_i \alpha_i)}{\prod_i \Gamma(\alpha_i)} \int \left(\prod_{i=1}^{k} \theta_i^{\alpha_i - 1} \right) \left(\prod_{n=1}^{N} \sum_{i=1}^{k} \prod_{j=1}^{V} (\theta_i \beta_{ij})^{w_n^j} \right) d\theta \tag{2.29}$$

which is also intractable due to the coupling between θ and β [15]. However, while the posterior distribution is not tractable for the exact inference, numerous approximate inference techniques can be applied adjusted for Latent Dirichlet Allocation, like the Laplace approximation [8], variational approximation [8] and Markov chain Monte Carlo [34].

2.3 Packages and APIs

Each one of the algorithms and experiments described through the course of this thesis will be implemented in the *R* programming language. In this section we cover the most relevant packages we will be using and their respective applications.

2.3.1 TwitteR and the Twitter Search API

Part of the novelty in our methodology depends on our approach of data collection. For the task, we use the package *twitteR* [22], which provides a straightforward interface in `R` to the Twitter Search API [31].

The Twitter Search API is an interface provided by Twitter to facilitate searches in their platform. Using this interface, developers have official support for querying Twitter for a word,

expression, hashtag, or a username and receive a set of tweets containing that specified query. In addition, friends and favourites of a particular user can be searched as well. This API also supports the restriction of the results interval of time, a geographical area or a language.

However, despite its obvious advantages, the Twitter Search service and, by extension, the Search API are not originally meant to be an exhaustive source of Tweets. For this reason, various restrictions were imposed on the service: the API can only retrieve tweets posted in the interval between the moment of the query and seven days earlier. Twitter also limits the search rate to 180 calls of the `Get Function` for each 15 minute window intervals [32], meaning that, after fetching a few thousand tweets, the API is blocked for the next 15 minutes.

The package *twitteR* attempts to mitigate some of these limitations with the addition of some tools, particularly the built in options for pausing the search when the limit of tweets is reached, automatically resuming after the 15 minutes interval.

2.3.2 tm

The package *tm* [18] is a popular package for text-mining functionality. It offers options for facilitating the management of text, incorporating straightforward functions for document manipulation. An integrated database back-end support is also included to minimize memory demands.

We will use the package capabilities predominantly to preprocess data sets, through actions such as converting characters to different standards, removing white spaces, word stemming and extraction of stopwords. Furthermore, we will also employ the corresponding functions to create and manipulate corpus and document-term matrices.

2.3.3 topicmodels

The R package `topicmodels` [27] provides basic infrastructure for fitting topic models and correlated topic models. It is based on data structures from the text mining package `tm`, and complements its functionality. Two algorithms are provided for fitting topic models: the variational expectation-maximization algorithm and Gibbs sampling.

In our implementation, this package will be used to build topic models with Latent Dirichlet Allocation, as well as for calculating the posterior probabilities of tweets.

2.3.4 e1071

The R package `e1071` [49] was originally developed as a mixture of useful functions of the Probability Theory Group, formerly E1071, from the Department of Statistics of TU Wien. It is mostly known from its functions for training classifiers based on support vector machines

(SVM), despite also providing other methods like shortest path computation, short time Fourier transform, naive Bayes classifier and bagged clustering.

2.3.5 igraph

The R package `igraph` [13] provides an infrastructure in `R` for igraph, a library collection of open source network analysis tools for creating and manipulating graphs. Functions like graph generation, computation of network properties based on path length and calculation of clusters in graphs are included.

We will use the tools available in this package to represent the relations among the collected hashtags in a visual manner.

2.3.6 shiny

The R package `shiny` [62] was developed by RStudio [63] for facilitating the creation of interactive web applications directly from `R` code. Shiny applications can be further extended with `CSS`, `html`, and `JavaScript`. RStudio also provides the option for hosting Shiny apps locally or on their own hosting servers.

After describing and evaluating our method, we will use `Shiny` to build a user-friendly application through the addition of an interactive user interface to our implementation.

Chapter 3

Related Work

The purpose of this chapter is to introduce some of the existing literature and research that contributed as a basis to the development of this thesis.

The popularization and growth of social networks in recent years attracted a lot of attention from the research community in different areas. Twitter, being the most popular micro-blogging service in the world, has become appealing due to the public availability of millions of short texts shared every day by a multitude of people, covering a wide variety of subjects.

3.1 Short Text Classification

One of the first works in improving search and message filtering for small texts is presented by Esparza et al. [21] as an initial study in category recommendation applied to a micro-blogging environment. Their approach relies in a previous manual categorization of each tweet into five predetermined categories (books, movies, games, music and applications) to train a classifier to predict classes of new tweets, validating the possibility of indexing Twitter data for category recommendation and information retrieval.

Many pieces of literature have applied Latent Dirichlet allocation (LDA) to news articles, books, academic papers and other large text corpora. However, Ramage et al. [61] are the first to experiment with the utilization of topic modeling to perform classification tasks in short documents like tweets. The authors employ Labled-LDA [60], a partially supervised learning model based on Latent Dirichlet allocation that incorporates supervision in the form of tweet-level labels, to classify the contents of a Twitter feed into four previously determined dimensions: *substance topics*, about events and ideas; *social topics*, comprised of language used toward communication; *status topics*, describing personal updates; and *style topics*, that embody broader trends in language usage. This classifications are then used with the purpose of profiling Twitter users and their habits for better follower recommendations.

Considering that data sparseness is the main cause why many classification tasks are

unsuccessful in achieving a decent accuracy rate when applied to short segments of text like tweets, Phan et al. [57] propose a method to expand the coverage of classifiers through the addition of external knowledge. The authors introduce a framework whose underlying idea is that, for each new classification task, a *universal data set*, consisting of large-scale data, is collected from an external source like Wikipedia, and used to build a classifier on both the small set of labeled training data and the rich set of hidden topics derived from that external data collection. The proposed framework was considered to be general enough to be applied to a wide variety of data domains like Twitter feeds.

In order to facilitate the retrieval of information from Twitter, Antenucci et al. [3] proposes a method to learn the relationships between the content of a tweet and the hashtags that could accurately describe it. In their approach, each tweet is represented as a frequency list of its comprising words, which are neither stopwords or hashtags, and the corresponding categories are the hashtags employed in the tweet. They introduce a technique to cluster hashtags in meaningful topic groups using a combination of co-occurrence frequency [58], graph clustering and textual similarity. Then, a method is described to classify a tweet, based on the content of the topic groups previously defined, through the employment of a combination of principal component analysis (PCA), dimensionality reduction and a variety of multi-class categorization algorithms. In this manner, the authors were capable of performing supervised classification tasks on otherwise intractable problems.

To achieve the task of empirically compare the content of Twitter with a traditional news medium using unsupervised topic modeling, Zhao et al. [87] propose the unsupervised topic model Twitter-LDA, a variation of Latent Dirichlet Allocation, developed specifically for taking into consideration the size limitations of tweets. Assuming that each tweet has only one topic, the Twitter-LDA classifier is employed to automatically collect posts classified as discussion topics. The topics in the New York Times are then used to semi-automatically group the relevant topics of the collected tweets into distinct topic categories. Finally, all topics are manually classified. Further experimentation reveals that Twitter-LDA may outperform regular LDA when coping with the short text characteristics of tweets.

Not completely satisfied with some shortcomings of Twitter-LDA, Mehrotra et al. [48] investigate techniques for tweet aggregation, with the goal of improving the accuracy of topic modeling in micro-blogs without modifying the fundamental concepts of latent Dirichlet Allocation. The authors experiment with five different schemes to aggregate tweets in a preprocessing step for training the topic model: the *Basic scheme*, the baseline where each tweet is considered as a single document and the model is trained on all tweets; the *Author-wise Pooling*, a previously suggested process [82][29], where for each author in the data set, the corresponding tweets are combined in the construction of a single document; the *Burst-score-wise scheme*, which first discovers trending topics [52], through the detection of excessive frequency of a term into the data, and proceeds to aggregate the tweets into a single document for each burst-term; the *Temporal scheme*, which attempts to capture the temporal consistency of the posts, through the chronological aggregation of tweets into intervals of one hour; and the *Hashtag-based scheme*,

which considers hashtags as indications to the context of the posts, through the combination of tweets sharing the same hashtag into a single document for every hashtag. The trained topic models were later compared in three different data sets. The proposed hashtag based pooling scheme was found to significantly outperform all the other alternative methods, revealing that hashtags may be valuable indicators of the substances of a tweet.

3.2 Hashtag Recommendation

The majority of previous research in hashtag recommendation applied to Twitter focused in suggesting a set of hashtags to complement a tweet in the moment it is being posted. For this reason, most of the methods described in this section can be used to recommend hashtags for an entire tweet but not for a single word.

In contrast to the earliest techniques of keyword extraction [84][44], Mazzia and Juett [47] introduce the use of probability distributions applied to hashtag suggestion for micro-blogging websites. In particular, the proposed method considers every hashtag as a category and, correspondingly, the tagged tweets as labeled data, thus allowing the application of a naive Bayes model to calculate the maximum *a posteriori* probability of each hashtag class, given the words of the tweet. The authors also propose some preprocessing procedures to adapt Twitter data to common machine learning techniques. However, one of the major drawbacks of their approach is the assumption of the mutual independence of the words inside tweets.

Other method that leverages on the similarity between tweets collected in a data set is proposed by Zangerle, et al. [85] whose goal is to discourage the usage of synonymous hashtags through the recommendation of homogeneous hashtags to a tweet. Their approach calculates the resemblance of tweets and attributes them a score based on a Term Frequency-Inverse Document Frequency (TF-IDF) scheme. Then, the hashtags are extracted and restricted to a final set of those with a score above a threshold. They experiment ranking this set based on three different metrics: the *OverallPopularityRank score*, based on the overall popularity of the tweets; the *RecommendationPopularityRank score*, based on the popularity within the most similar tweets; and the *SimilarityRank score*, based on the most similar tweets. The latest metric is reported to provide the best results, particularly when recommending five hashtags.

Taking into account that Twitter users adopt distinct writing styles when posting messages in the platform, an attempt to improve on the method proposed above was later suggested by Kywe, et al. [37], who develop a personalized hashtag recommendation system considering not only the tweet content, but also the user preferences. Their approach applies an explicit user profile based collaborative filtering method, which represents each user as a vector weighted by a Term Frequency-Inverse Document Frequency (TF-IDF) scheme, analyzing the frequency of some key-words in user profiles. After calculating the weights of every user, they proceed to extract the hashtags from their most similar tweets, which are subsequently ranked and recommended to the user. This method recommends hashtags collected from one month of posts and was found

to outperform the previous approach by up to 20 percent. The authors also discovered that the inclusion of other hashtags and tweets previously posted by the user improved performance slightly.

A different method based on language modelling is proposed by Efron [17] which attempts to improve Twitter search through the retrieval of hashtags related to a keyword query, from a previously collected data set of tweets. He estimates the probability of every word in the data set to co-occur with every hashtag in a tweet and smooths the models through Baysian updating with Dirichlet priors. The amount of information contained in every hashtag is later assessed through the employment of an Inverse Document Frequency (IDF) scheme. Finally, KL-divergence [36] is used to compare the models of every hashtag with the one from the original query, measuring their respective relevance, according to which the hashtags are ranked. In this manner, hashtags are predicted through query expansion techniques.

Due to the volatility in the appropriate number of clusters to be found, Li and Wu [43] consider that the common procedures for classification of regular text are inadequate for hashtag prediction and employ network relatedness methods to recommend a hashtag for a new tweet. In their work, they argue that if the correlation between tweets can be mathematically measured, then it can also be represented as a network of points in a high dimensional space through a latent space model. The similarity of the tweets is then calculated by the Euclidean distance between those points, considering the words in a dictionary as the orthonormal bases. The semantic correlation among words is determined through the construction of a matrix based on WordNet similarity [51] and its subsequently weighting through their co-occurrence in tweets. Finally, the tweets that are closest to the original are collected and new ones are added to the set until a certain hashtag becomes dominant, that is, appears in the majority of the tweets. One of the advantages of this method is the easiness of updating the dictionary, since the addition of a new word only requires the computation of an extra column in the matrix.

Topic modeling is first employed for hashtag recommendation in Twitter by Godin et al. [26], attempting to facilitate indexing and search of new tweets. In their work, they begin by introducing a binary language classifier for tweets based on the Naive Bays method and Expectation-Maximization to filter out posts written in languages other than English from their database. A Latent Dirichlet allocation model is then trained to cluster the remaining tweets in a set of general topics. Given a new tweet, its underlying topic distribution is also generated and top keywords from the dominant subjects are recommended as hashtags. The authors argue that the suggestion of general words as hashtags instead of already existing ones is advantageous when compared to previous work, since it enables categorization and search of tweets. A qualitative evaluation of this approach concluded that in 80% of the cases, a suitable hashtag is recommended from a selection of five possibilities.

She and Chen [69] develop a supervised topic model-based solution for hahstag recommendation on Twitter, abbreviated TOMOHA. They employ an adaptation of Twitter-LDA [87] which considers hashtags as the labels of the local topics to generate a model capable of analyzing relationships among words, hashtags and topics of different posts. Their work follows some

assumptions like that each tweet is solely about one of the topics of the corpus, and that every word is either a local topic word or a background word, common in most tweets. Considering that users can be influenced by their network of contacts, the study also takes into account user following relationships into the model. Finally, the probability of every hashtag to be contained in a new tweet is calculated in order to recommend the most probable ones, using asymmetric Dirichlet distribution. The authors also propose the use of parallel computing to accelerate the model training process, through the random assignment of tweets to different processors.

Pursuing to encourage more widespread adoption and usage of hashtags in Twitter, Dovgopol et al. [16] experiment with different methods of hashtag recommendation for new posts. The authors argue that hashtag recommendation carries two major challenges that set it apart from traditional document tag recommendations: the fact that data consists of huge volumes of small and noisy content, for which they propose some preprocessing methods; and the size limitations of tweets that discourage the repetition of words [77], making traditional classification techniques like TF-IDF inefficient, for which they present an original alternative. They start by using Inverse Document Frequency (IDF) to discover the three most important words in the target tweet and filter out all tweets not containing those words, with the purpose of improving the speed of the system. Next, they present an hybrid classifier that employs the Bayes' Theorem, used to rank hashtags by their probability of co-occurring with each term in a tweet, and k-Nearest Neighbor, used to calculate the number of hashtags that occurred in similar tweets, and use it to classify and recommend hashtags for the given tweet.

Attempting to improve user interactions with reports, articles and publications, Xiao et al. [83] propose a method to recommend Twitter hashtags related to a keyword that represents a news related topic. They begin by collecting news articles from major agencies and employ an original Probabilistic Inside-Outside Log (P-IOLog) method to cluster them into vectors representing different topics, according to the presence of topic-specific informative words that co-occur with the given target word in the editorials. Tweets related to news which were published concurrently to the articles are also collected from a set of manually selected accounts and concatenated according to their hashtags, with the purpose of building a similar hashtag vector. The similarity between each news-topic vector and each hashtag vector is subsequently calculated, and the hashtags with the highest similarity scores are recommended for the news topic. Experiments performed by the authors revealed that the proposed process outperforms more common methods like Term Frequency-Inverse Document Frequency (TF-IDF), recommending more relevant hashtags to the selected news topic. However, they acknowledge that their approach is specifically tailored to news related keywords, being incapable of recommending hashtags to topics not reported by news agencies.

Shi et al. [70] also propose a news related hashtag recommendation solution denominated *Hashtagger*, with the purpose of finding relevant hashtags for stimulating the dissemination of news articles in Twitter. Opposing to the most modern alternatives, instead of focusing on topic models, they suggest a learning-to-rank (L2R) approach for modelling hashtag relevance, in order to avoid the need of continuously retrain the model to keep it updated. They adapt standard

Information Retrieval (IR) methods to accept a news article as the initial query and a set of tweets representing their embraced hashtags as the documents. For each new article, a ranking is performed based on automatic query formulation, to retrieve candidate hashtags, and a previously trained pointwise learning-to-rank (L2R) method is subsequently used to score the relevance of each hashtag for each article. In the process of generating keyword queries from the news articles, the proposed method employs part-of-speech-tagging (POS) to select pairs of nouns/phrases and calculates a score based on Term Frequency-Inverse Document Frequency (TF-IDF) to select the top 5 pairs. The authors argue that the features that make a hashtag relevant to an article remain constant over time, allowing the L2R model to be trained on manually labeled article-hashtag pairs only once and used several times. However, the procedure requires to wait until enough tweets are collected for being able to provide recommendations, resulting in up to 12 hours run-times. For this reason, the same authors later propose *Hashtagger+* [71], an improved version of *Hashtagger* that employs cold-start algorithms to accelerate the recommendation process. This particular method was found to perform exceptionally well with less popular articles.

Focusing in the volatility of hashtag popularity and data sparsity, Otsuka et al. [54] suggest a method employing an adapted ranking system to recommend more current trending hashtags to new Twitter posts. Inspired by classic information retrieval methods, their process requires an earlier compilation of millions of tweets into two nested maps data structures: a Term to Hashtag Frequency Map (THFM), that maps each word with the frequency they co-occur with each hashtag; and a Hashtag Frequency Map (HFM), an analogous data structure that maps each hashtag with the respective term frequencies. Next, these mappings are manipulated by a Hashtag Frequency-Inverse Hashtag Ubiquity scheme (HF-IHU), a variation of the popular Term Frequency-Inverse Document Frequency (TF-IDF) weighting scheme adapted to score hashtag relevancy while taking into account the data sparseness of data sets collected from Twitter, to rank the hashtags in order of relevance to the tweet. The algorithm also makes use of the Hadoop distributed computing platform [55] with Map-Reduce [14] to accelerate the mapping process. The authors conducted experiments on a large Twitter data set and concluded that their proposed method successfully yielded relevant hashtags, outperforming other popular schemes like k-nearest neighbors, k-popularity and Naive Bayes.

One of the very few approaches that take into consideration the chronological evolution of hashtags in Twitter is proposed by Harvey and Crestani [28], who investigate the exploitation of temporal patterns for hashtag recommendation for new posts, attempting to tackle the heavily fluctuation of hashtag popularity over time [40][45]. The authors consider that hashtags can be splitted in two different types: *organizational hashtags*, which remain popular over long periods of time; and *conversational hashtags*, which refer to particular events and are only relevant during a short interval of time. The proposed approach first applies a collaborative filtering personalization technique, extending a popular method based on Term Frequency-Inverse Document Frequency (TF-IDF) [85], to determine a set of candidate hashtags from the users' previous posts. Subsequently, the tweets containing those hashtags are reviewed, taking into account the time when they were posted, in order to re-weight the scores of possible hashtags by their temporal relevance. Experimental results were positive, showing that even when the

temporal weighting is not completely successful, the damages imposed to the original ranking are usually insignificant.

A specially distinct approach is taken by Hamidreza Alvari [2] who proposes a simple method to recommend relevant hashtags for a new tweet, based solely on the hashtag usage history of the users. The author argues that the content of the tweets is not necessarily required to perform good hashtag recommendation and that most alternatives in the literature carry an excessively computational weight due to the lack of strong natural language processing techniques to analyze the contents of tweets. The proposed process considers hashtag recommendation as a collaborative filtering based problem, formulates it into an optimization problem and proceeds to solve it through the employment of low-rank matrix factorization to characterize users and hashtags by inferring vectors of latent factors from the user's hashtag usage history. According to the author, when users compose a new post, they are very likely to select hashtags similar to the ones they have already adopted in previous tweets. Empirical experiments demonstrate that the proposed method is indeed capable of proper hashtag recommendation.

In contrast with the majority of the previously described methods, Li et al. [42] propose a method to suggest relevant Twitter hashtags for a given concept represented by a keyword, instead of for an entire tweet. The authors were the first to exploit distributed language representations of words in hashtag recommendation for a subject, through the construction of word-embeddings [50] from a database containing billions of words extracted from tweets, thus considering both the semantic and syntactic of the terms. When querying for a keyword, every hashtag extracted from the tweets is ranked according to the cosine similarity score between their corresponding word-embedding vector and the embedding vector of the keyword itself. The highest ranked hashtags are then recommended. The authors concluded that their approach outperformed the conventional term co-occurrence based methods and, despite being tested with health-related concepts, the procedure could be applied in any content domain.

3.2.1 Proprietary Methods

In recent years, due to the improvement and popularization of social media platforms, as well as the reduced broadcasting cost, marketers and communicators have increased their focus on online advertisement and content propagation on social networks. Being the worlds' most popular micro-blogging service, Twitter became an obvious target of product research, attempting to direct its multiple conveniences and instruments towards marketing and content propagation to the masses.

For companies and influencers, the employment of the appropriate hashtags is capable of significantly improving brand awareness and connection with their target audience, through the encouragement of conversations related to their products or ideas. Hashtags are also an important tool for community building and management of user-generate content, considering that posts containing hashtags receive twice the engagement of those without them [67].

The need to better measure the success of online campaigns and collect more knowledge about the target audiences to optimize the reach of social media posts drove the establishment of a few companies and websites whose services focus on hashtag management, analytics and recommendation.

Companies like *Sprout Social* [1] have specialized in hashtag analytics directed to enhance social media strategies of a specific brand, building reports about which hashtags are commonly employed when referring to a particular company in Twitter. Other websites like *Hashtagify.me* [2] attempt to amplify their clients' reach, through hashtag marketing, as well as allowing them to track their competitors, discovering common variations of hashtags frequently used alongside a queried keyword, as well as its major influencers. Ritekit develops services like *Ritetag*[3], a website that focuses in hashtag suggestions for pictures and Tweets and not only recommends relevant hashtags, along with some extra information, but also discourages the use of some other hashtags.

Despite their usefulness for professionals, companies for hashtag analystics and recommendation like the ones described above carry two significant drawbacks. Due to their commercial nature, these services require a costly premium subscription to unlock most of their useful features and tweaks, or even to be used at all. Furthermore, it is of the best interest of the companies maintaining these services to keep their technology secret, therefore there is little to no information on their methodology of work, diminishing their contribution from a scientific perspective.

3.2.2 Shortcomings of Current Hashtag Recommendation

While advancements in research related to hashtag recommendation in Twitter has reached some significantly breakthroughs, current approaches, still carry some shortcomings.

First of all, the majority of the represented literature limits their focus on hashtag recommendation to the improvement of new posts, thereby converging their attention in suggesting hashtags to short texts and not to a topic or a keyword. In this manner, while these methods are advantageous for increasing the reach and consistency of new content, they are less convenient when searching for new information and connections.

Furthermore, the described approaches are universally dependant on the previous extraction of extensive databases populated with millions of random tweets for the process of training their classifiers and recommender systems. Due to the static character of these data sets, very few studies take into consideration the evolution of the suggested hashtags, whose connotation and popularity frequently change over time. For example, in 2010 the hashtag *#Johannesburg* would presumably be considered relevant to the topic *#WorldCup*, considering that the FIFA World Cup was hosted in South Africa in that same year. However, that would probably not be

[1] *https://sproutsocial.com/about*
[2] *http://hashtagify.me/explorer/about*
[3] *https://ritekit.com/about/*

perpetuated until 2014, when the same event was held in Brazil. Surprisingly, even professional hashtag recommendation services make this kind of mistakes, specially when searching for less popular topics, revealing that their databases may not be renovated frequently enough.

Therefore, in this thesis, we propose and develop techniques to properly identify, collect and recommend hashtags currently relevant to a given topic. Our approach employs methods for querying Twitter to assemble smaller but more consistent data sets, which are easier to maintain updated, to train a classifier. In addition, our metrics for relevance are less influenced by co-occurrence than most of the alternatives.

Chapter 4

Design and Implementation

As stated in the previous chapter, few works in the literature focus on hashtag recommendation for a word or a concept. While their theories are usually sufficient when posting new tweets, they are inadequate for content discovery. In addition, scarcely any of the current approaches take into consideration the evolution of the context of the hashtags through time. As far as we know, we propose the first method that tackles both issues simultaneously.

In this chapter, we will focus on describing our own implementation of a recommendation system that is capable of identifying and collecting hashtags currently relevant for a given topic in Twitter, assuming that topic to be denoted itself by a hashtag. In the interest of keeping our explanation consistent and straightforward, through the course of the present and posterior chapters, we assume that every topic of interest can be accurately expressed through a hashtag that will be designated as *seed*.

We based our approach on two main ideas that differentiate it from current approaches: first, instead of employing a random sample of tweets, we will be using the Twitter Search API to collect posts containing a query, thus avoiding the acquisition of unnecessary data; in addition, instead of simply considering the tweets gathered from searching the *seed*, we will be extending our search with the hashtags found to be relevant to the topic. In this manner, we expect to also retrieve hashtags used alternately with the *seed* but rarely together in the same tweet, like variations of event names according to the current year.

We will be presenting TORHID in the following order: we start by introducing a new approach for the process of data set collection, which attempts to mitigate some prevailing issues in current methods. Afterwards, we describe the process of training a classifier to label new tweets as *relevant* or *irrelevant*, with the purpose of discarding the irrelevant ones. We finish by merging those processes together in order to construct a system to properly recommend hashtags to a given topic.

4.1 Data set Collection and Preprossessing

As customary for text mining processes, in order to implement any algorithm of hashtag recommendation, a primary mandatory step is the collection of training data. Current works in the literature are almost universally dependent of extensive data sets, containing millions of arbitrary tweets, to train their classifiers. This data is usually acquired with resort to the Twitter Streaming API, which allows the collection of a random sample from the live streaming of messages posted to Twitter in a given interval of time.

Despite representing the common practice, we consider that this method entails considerable inconveniences due to its random character. As a matter of fact, some recent research works recognize that the great majority of tweets don't contain hashtags at all [30][37][35]. For this reason, without employing appropriate methods of keyword extraction for hashtags, a practice increasingly depreciated in recent works, a considerable percentage of the collected tweets becomes ineffective for hashtag recommendation. This situation is aggravated when dealing with less popular topics, for which there is an increased probability of not being discussed enough while the data collection process is taking place. Therefore, most approaches choose to extract millions of tweets in the sampling process, thus increasing memory and processing needs. This disadvantages are exacerbated in cases where maintenance is required to keep the databases temporally relevant.

The task of collecting appropriate and comprehensive data for accurately training the classifier is certainly more challenging than it seems at first sight. Therefore we devoted some time to develop a more efficient method of gathering data and decided to focus this approach in the given topic from the start, generating smaller individual data sets for each case, instead of a massive multi-topic database. For this task, we employ the Twitter Search API, which allows the retrieval of messages in Twitter containing a specified query, to collect tweets that are already somewhat relevant to the desired topic. A straightforward idea would be using this process to collect thousands of tweets containing the *seed* and inspect the corpus to discover new hashtags. However, the temporal limitations of the Twitter Searching API make this method impractical if not unusable.

We rather collect the data in the following manner: we start by searching for the *seed* and obtain a combination of the most recent and the most popular tweets containing that hashtag. This selection of tweets is stored and inspected, looking for new hashtags. We perform a basic preprocessing step by only considering the hashtags which appear a number of times above a threshold to be added to a queue. This queue of hashtags is expanded through the repetition of this process, searching for the hashtags in the queue, one at a time, instead of the *seed*, until we empty the queue or reach a limit on the number of searches (in order to prevent loops).

The Twitter Search API also supports the restriction of the results within an interval of time, a geographical area or a language, reducing the need of performing some of the preprocessing steps usually required. In this manner, we obtain a crude data set containing tweets with hashtags related among themselves and to the *seed*, facilitating the process of training the classifier.

The pseudo-code representation of the method described above to collect the data set with tweets is shown in Algorithm 1. We implemented the proposed data collection process in the `R` programming language, employing the package `twitteR` [22]. This package provides a straightforward interface in `R` to the Twitter Search API, including tools to mitigate some of its limitations. Table 4.1 discloses some examples of tweets collected with this method for the seed *#preterm*.

Input: *seed*, *max.searches*, *threshold*
$HashtagQueue \leftarrow$ initiate queue with *seed*
$AllTweets \leftarrow \emptyset$
while *HashtagQueue not empty* ***and*** $max.searches > 0$ **do**
  $CurrentHT \leftarrow$ dequeue $HashtagQueue$
  $NewTweets \leftarrow$ set of retrieved tweets containing $CurrentHT$
  $AllTweets \leftarrow AllTweets \cup NewTweets$
  $NewHashtags \leftarrow$ hashtags appearing in $NewTweets$ more than *threshold* times
  $HashtagQueue \leftarrow HashtagQueue \cup NewHashtags$
  $max.searches \leftarrow max.searches - 1$
end
output $AllTweets$

Algorithm 1: Collecting Tweets

#	Tweet
1	Some of the rules that #Preterm is challenging is the state's transfer agreement requirement, which #CapitalCare also challenged.
2	One more thing to keep me up at night. Who needs sleep anyway? #insomnia #NorthKorea
3	How a card with an imprint of a foot is saving babies in Uganda. #lowbirthweight #preterm #neonate https://t.co/XPVLb0Iwh2
4	Jane Norman and Phillip Bennett argue the need for alternatives to #progesterone for preventing #preterm births. https://t.co/9ELp6GNbpM
5	A secondary part of #Preterm's case against the state is to challenge the pats of the "Heartbeat Bill" added to the budget. #StopTheBans
6	Hey, me too! #sleepapnea #FirstWorldProblems https://t.co/H1EwGRoVyB
7	RT @CochraneUK: #CochraneEvidence - Cervical cerclage for preventing preterm birth #EEMidwifery @WeMidwives https://t.co/bbA5NQt6yg
8	I don't take my job for granted. #NICU #nurselife
9	The little-known strategies to manage #postpartum #anxiety in just 5 minutes. #birthtrauma #preemie #micropreemie https://t.co/h4WPPUEfOe
10	I begin to love this little creature, and to anticipate his #birth as a fresh twist to a knot which I do not wish to untie.
11	RT @dad2summit: "Breastfeeding isn't just a women's issue, it's a human issue." - Brian Wallace #parenting #infants #rights
12	Easter Egg Hunt Canceled Due To Aggressive Parents #NICU #Preemie https://t.co/v4QqcLnqrm
13	#SCSL facilities support #KangarooMotherCare for #preterm #newborns https://t.co/uqtXUfzhsI https://t.co/ACWq0Eb4vU
14	India is silently promoting of cutting woman's body, making them weak throughout their life. #CSECTION https://t.co/QWyBDY8vnm
15	RT @joylawn: 15mill #preterm +1mill deaths/yr. Richcountries invest NICU Prevention little impact so far AfricanNewborns LeftBehind

Table 4.1: Example of Collected Tweets for the *Seed #Preterm*

For undisclosed reasons, some of the posts collected through the Twitter Search API are not presented in their entirety, missing a few final characters or words. While this means that some tweets of the data set may be lacking hashtags, we decided to keep them for two reasons: since the classifier will be incapable of differentiating hashtags from other words, these posts are still acceptable for training; and these tweets without hashtags represent a very small portion of the complete database.

4.1.1 Preprocessing

After collecting the totality of the train set, we perform some preprocessing steps to prepare the corpus for training the topic model. The procedures are the following:

- all non ASCII characters are removed;
- the text is converted to UTF-8 for better compatibility with the packages;
- urls are removed;
- all non-alphabetical characters are removed;
- letters are converted to lower case;
- stopwords are removed;
- words with less than three characters are removed;
- word stemming is applied;

We decided to not remove retweets, since research has proven them to be useful as an *indicator of interestingness* and suited to mitigate sparseness in micro-blog posts [53], facilitating information retrieval. Table 4.2 discloses the results of preprocessing the examples from Table 4.1.

#	**Preprocessed Tweet**
1	rule preterm challeng state s transfer agreement requir capitalcar also challeng
2	one thing keep night need sleep anyway insomnia northkorea
3	card imprint foot save babi uganda lowbirthweight preterm neonat
4	jane norman phillip bennett argu need altern progesteron prevent preterm birth
5	secondari part preterm s case state challeng pat heartbeat bill ad budget stoptheban
6	hey sleepapnea firstworldproblem
7	cochraneuk cochraneevid cervic cerclag prevent preterm birth eemidwiferi wemidw
8	dont take job grant nicu nurselif
9	littl known strategi manag postpartum anxieti just minut birthtrauma preemi micropreemi
10	begin love littl creatur anticip birth fresh twist knot wish unti
11	dadsummit breastfeed isnt just women issu human issu brian wallac parent infant right
12	easter egg hunt cancel due aggress parent nicu preemi
13	scsl facil support kangaroomothercar preterm newborn
14	india silent promot cut woman s bodi make weak throughout life csection
15	joylawn mill preterm mill death yr richcountri invest nicu prevent littl impact far africannewborn leftbehind

Table 4.2: Preprocessed Tweets from Table 4.1

4.2 Classifier

If we were to scan the tweets collected by Algorithm 1, without any further restraints, we would be able to retrieve a large assortment of hashtags. However, only a small minority of those would actually be relevant to the intended topic. Due to the unrestrained expansion of the queue of hashtags, the network will always reach nodes whose descendents' hashtags cease to have a sufficient degree of relation to the original topic, undermining the final set. Through the analysis of the queues of hashtags we came to the following conclusion:

Every time a hashtags is searched on Twitter, the recovered tweets may contain hashtags with four types of relations to the original: they may be completely unrelated; they may be synonyms or related to the same topic; they may represent a subtopic; or they may be a supertopic. While collecting subtopics generally doesn't entail any issues, when the algorithm harvests a supertopic it may result in the consequent collection of its subtopics. When the retrieved supertopic is vague enough, its subtopics have a great probability of not being relevant enough for the *seed*.

For that reason, it becomes necessary to find a method to appropriately separate the relevant hashtags we actually want, from the extra ones we collect unintentionally. While correctly evaluating the relevance of a hahstag to another may be a challenging task even for a human being, the tweets containing them provide additional context and are adequate for training a classifier. For this reason, instead of attempting to classify the hashtags themselves, we automatically associate each one with the classifications of the respective tweets. In this manner, even ambiguous words, concatenations of words and acronyms should have a better chance of being classified, specially since the topic model employed does not take into consideration the semantics of the terms.

Our approach divides the classification problem in three principal steps: first, we discover the latent topics of the various tweets of the train set; next, we automatically aggregate the combinations of those topics as relevant or irrelevant to the *seed*; and finally, we train a binary classifier that considers the labels of the tweets and their topic distributions to classify new ones as *relevant* or *irrelevant*.

For the first step, we consider that every preprocessed tweet from our data set is a single document and build a document term matrix based on the collected corpus. An example of a document term matrix generated in this manner is depicted in Table 4.3. This DTM is then used to determine the most effective topic model as possible, as well as the most adequate topic distribution associated with each document, through Latent Dirichlet Allocation. The hidden topic distribution of a tweet can be seen as a summary of its content and can be used to detect similarities with other documents within the corpus.

	Terms								
Document	babi	preterm	nicu	parent	birth	preemi	infant	pregnant	neonat
1423	0	0	2	0	0	0	0	0	0
1457	0	0	2	0	0	1	0	0	0
1476	4	0	0	0	0	2	0	0	0
1502	0	0	2	0	0	1	0	0	0
1898	0	0	0	0	0	0	0	0	0
277	0	0	2	0	0	1	0	0	0
314	1	0	1	0	0	0	0	0	0
759	0	0	2	0	0	0	0	0	0
874	0	0	2	0	0	1	0	0	0
898	0	0	0	0	0	0	0	1	0

Table 4.3: Example of a Document Term Matrix

It is important to notice, however, that despite the most frequent terms of the determined topics may be indicative of a specific subject, these are not necessarily equivalent to real life topics. An example of a topic model trained with this method is depicted in Figure 4.1.

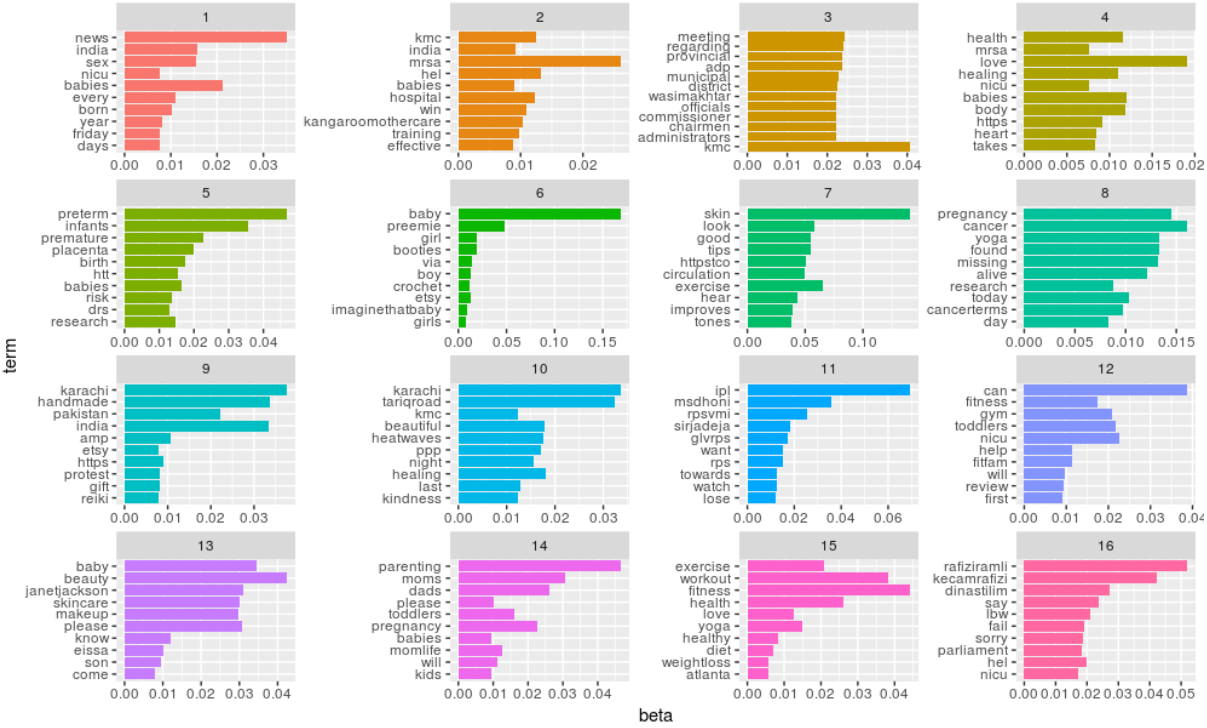


Figure 4.1: Example of a Topic Model

In the second step, we need to divide the tweets in two segments according to their relevance to the original topic. Since we want TORHID to be a completely unsupervised method, we decided to inspect the DTM and label each document of the data set as *relevant* if it contains the *seed* (preceded or not by the octothorp) or *irrelevant* otherwise. This classification is not perfect in the sense that some documents related to the topic may be incorrectly labeled for not containing the exact *seed*. However, from our experience, these tweets usually represent a negligible percentage of the data set.

The final step consolidates the previous two into training data. Due to its flexibility in finding good margins for data separation, we chose to use a support vector machine for our classifier. We associate the topic distributions of each tweet with its corresponding label, thus aggregating all the combinations of latent topics available in the train set in two different classes. After collecting a train sample, the support vector machine is trained to classify new tweets as *relevant* or *irrelevant* based on their corresponding topic distributions.

The pseudo-code representation of the three steps comprising the method described above is described in Algorithm 2.

We implemented the proposed process for training the classifier in the `R` programming language, employing the package `topicmodels` [27], which provides straightforward functions

Input: $AllTweets$, $seed$, k
$Dtm \leftarrow$ document term matrix of $AllTweets$
Generate the LDA model of Dtm with k topics
$RelevantDistributions \leftarrow$ all topic distributions from $Tweets$ containing $seed$
$RelevantDistributions\$Class \leftarrow$ relevant
$IrrelevantDistributions \leftarrow$ all topic distributions from $Tweets$ not containing $seed$
$IrrelevantDistributions\$Class \leftarrow$ irrelevant
$AllDistributions \leftarrow RelevantDistributions \cup IrrelevantDistributions$
$Classifier \leftarrow$ SVM model to predict $class$ from $AllDistributions$
return $Classifier$, $LdaModel$

Algorithm 2: Classifier

for building topic models based in Latent Dirichlet Allocation, as well as the package `e1071` [49] which provides methods to implement the support vector machine classifier. Table 4.4 describes an example of the classifications assigned by this method to a set of tweets and Table 4.5 exemplifies a train set for the support vector machine.

#	Tweet	Classification
1	Some of the rules that #Preterm is challenging is the state's transfer agreement requirement, which #CapitalCare also challenged.	Irrelevant
2	One more thing to keep me up at night. Who needs sleep anyway? #insomnia #NorthKorea https://t.co/m2J92HBLjL	Irrelevant
3	How a card with an imprint of a foot is saving babies in Uganda. #lowbirthweight #preterm #neonate https://t.co/XPVLb0Iwh2	Relevant
4	Jane Norman and Phillip Bennett argue the need for alternatives to #progesterone for preventing #preterm births. https://t.co/9ELp6GNbpM	Relevant
5	A secondary part of #Preterm's case against the state is to challenge the pats of the "Heartbeat Bill" added to the budget. #StopTheBans	Irrelevant
6	Hey, me too! #sleepapnea #FirstWorldProblems https://t.co/H1EwGRoVyB	Irrelevant
7	RT @CochraneUK: #CochraneEvidence - Cervical cerclage for preventing preterm birth #EEMidwifery @WeMidwives https://t.co/bbA5NQt6yg	Relevant
8	I don't take my job for granted. #NICU #nurselife	Irrelevant
9	The little-known strategies to manage #postpartum #anxiety in just 5 minutes. #birthtrauma #preemie #micropreemie https://t.co/h4WPPUEfOe	Irrelevant
10	I begin to love this little creature, and to anticipate his #birth as a fresh twist to a knot which I do not wish to untie.	Irrelevant
11	RT @dad2summit: "Breastfeeding isn't just a women's issue, it's a human issue." - Brian Wallace #parenting #infants #rights	Irrelevant
12	Easter Egg Hunt Canceled Due To Aggressive Parents #NICU #Preemie https://t.co/v4QqcLnqrm	Irrelevant
13	#SCSL facilities support #KangarooMotherCare for #preterm #newborns https://t.co/uqtXUfzhsI https://t.co/ACWq0Eb4vU	Relevant
14	India is silently promoting of cutting woman's body, making them weak throughout their life. #CSECTION https://t.co/QWyBDY8vnm	Irrelevant
15	RT @joylawn: 15mill #preterm +1mill deaths/yr. Richcountries invest NICU Prevention little impact so far AfricanNewborns LeftBehind	Relevant

Table 4.4: Classification attributed to Tweets from Table 4.1

#	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Classification
1	0.201350365938869	0.200895288433061	0.201600018454252	0.198076575220882	0.198077751952935	Irrelevant
2	0.200432163206433	0.19820056709395	0.199218848202671	0.206223401701195	0.19592501979575	Irrelevant
3	0.201091455094151	0.201728371159158	0.198920222015087	0.199650569927917	0.198609381803687	Relevant
4	0.200762838859385	0.206843021265519	0.198554641414959	0.199159327648504	0.194680170811633	Relevant
5	0.197722395540538	0.204629801607133	0.197456382657291	0.200273024415732	0.199918395779307	Irrelevant
6	0.199546915601299	0.200813189420656	0.200770426811711	0.198748884989713	0.200120583176621	Irrelevant
7	0.202588391613233	0.203853243651978	0.198378894730528	0.198336817264618	0.196842652739643	Relevant
8	0.203445281166697	0.199630705000399	0.19839356584914	0.199277579970908	0.199252868012857	Irrelevant
9	0.197067443147154	0.198533445886119	0.210096872007458	0.199320500564699	0.19498173839457	Irrelevant
10	0.199087709610595	0.196442571804194	0.205846565419953	0.20028864883948	0.198334504325778	Irrelevant
11	0.19972006148384	0.196405338594467	0.203528968304008	0.202546359263627	0.197799272354058	Irrelevant
12	0.205446662409353	0.202722916628599	0.201021472750483	0.195382718857437	0.195426229354128	Irrelevant
13	0.200683780578818	0.197744899841737	0.199643091345472	0.200761997914563	0.20116623031941	Relevant
14	0.199287743327576	0.199733503283252	0.20248116164658	0.195179784016463	0.203317807726128	Irrelevant
15	0.203101470982174	0.198661284383724	0.197108221025688	0.203567156432395	0.19756186717602	Relevant

Table 4.5: Example of Train Set for the Support Vector Machine

4.3 Collecting the Final Recommendation

The final stage of the process for recommending hashtags relevant to a topic is similar to the initial step for data collection. However, we now have a classifier capable of categorizing each of the newly collected tweets as relevant or irrelevant to the subject of interest. In this manner, we can restrain the expansion of the hashtag queue, discarding the irrelevant tweets in place.

After training the classifier, we start by employing the Twitter Search API to query Twitter and collect a combination of the most recent and the most popular tweets containing the *seed*. However, this time we want to remove the irrelevant tweets from the set before inspecting them for hashtag retrieval.

We consider the retrieved assortment of tweets as a corpus of documents and use it to generate a new Document Term Matrix with exactly the same terms as the DTM used to train the classifier. Through this new DTM, the posterior probabilities of the topics for each document are calculated, that is, the topic distribution of each tweet is determined according to the topic model previously used for training the classifier. While the process of calculating these posterior probabilities may seem complex at first, software packages for topic modeling already include an implementation of the required algorithms based on Gibbs sampling.

After calculating the topic distributions of the corpus, the classifier is employed for labeling each tweet as *relevant* or *irrelevant* and subsequently discard all the irrelevant ones from the group. The hashtags from the tweets identified as *relevant*, whose frequency is above a threshold, are added to a queue and stored in an array. The process is repeated with the new search terms in the queue until it becomes empty or a limit of iterations is reached. At the end of the execution, the array contains a list of hashtags currently relevant to the explored *seed*.

The pseudo-code representation of the method described above to collect the final relevant hashtags is shown in Algorithm 3.

In addition to the ones introduced in previous steps, the function used to implement this method for collecting hashtags relevant to a *seed* in `R` was *Posterior* from the package `topicmodels` [27], to calculate the topic distributions of the new tweets according to the provided topic model based on Latent Dirichlet Allocation.

An example of the set of hashtags returned from executing our program with the *seed* *#preterm* is depicted in Table 4.6.

To improve the results, every new tweet collected during the execution of this step is also stored and added to the corpus used to train the classifier in the next executions for that *seed*. The idea is the classifier should provide more accurate results each time we use TORHID to collect hashtags, since we add more data from the previous iteration.

Input: *seed*, *max.searches*, *threshold*, *LdaModel*, *Classifier*
HashtagQueue ← initiate queue with *seed*
AllTweets ← ∅
RelevantHashtags ← ∅
while *HashtagQueue not empty* ***and*** *max.searches* > 0 **do**
  CurrentHT ← dequeue *HashtagQueue*
  NewTweets ← set of retrieved tweets containing *CurrentHT*
  AllTweets ← *AllTweets* ∪ *NewTweets*
  NewTweets ← preprocessed *NewTweets*
  TopicTweets ← topic distributions of *NewTweets* (with topics from *LdaModel*)
  RelevantTweets ← tweets labeled *relevant* by Classifier
  NewHashtags ← hashtags in *RelevantTweets* more than *threshold* times
  HashtagQueue ← *HashtagQueue* ∪ *NewHashtags*
  RelevantHashtags ← *RelevantHashtags* ∪ *NewHashtags*
  max.searches ← *max.searches* − 1
end
store *AllTweets*
return *RelevantHashtags*

Algorithm 3: Collecting the Relevant Hashtags

Hashtags			
#babies	#kangaroomothercare	#preemies	#prematurity
#countrychildmag	#newborn	#prematurebaby	#preterm

Table 4.6: Example of Hashtags Returned for the *Seed #Preterm*.

4.3.1 Weighting the Relevance

After collecting the final selection of hashtags, we can try to measure their relevance to the original topic. However, unlike the current approaches for hashtag recommendation, we developed TORHID to allow the graphical representation of the collected data, including the illustration of how relevant the hashtags are to each other. In this manner, instead of weighting and ranking the hashtags themselves, we direct and weight the edges of the network.

Our algorithm works in the following manner: we begin by creating an adjacency matrix with every recommended hashtag and zeroing the values. Next, we use the Twitter Search API again to search for the hashtags in Twitter, one by one, and fill in the adjacency matrix the number of times that each one of the other hashtags is contained in the returned tweets. An example of an adjacency matrix created through this process is depicted in Table 4.7.

Finally, we remove the cycles from the matrix through its conversion into an upper/right triangular matrix. In this manner, the adjacency matrix always represents a tree whose root is the *seed*. An example of an upper/right triangular matrix constructed from the adjacency matrix in Table 4.7 is depicted in Table 4.8.

Hashtag	preterm	prematurity	prematurebaby	babies	newborn	countrychildmag	preemies	kangaroomothercare
preterm	97	6	5	15	8	5	0	4
prematurity	10	95	6	3	6	5	0	0
prematurebaby	9	7	78	0	6	6	0	0
babies	3	1	0	64	6	0	0	0
newborn	4	3	2	9	49	0	0	0
countrychildmag	3	0	1	0	0	31	3	0
preemies	5	6	5	3	4	0	24	0
kangaroomothercare	8	0	0	2	0	0	6	12

Table 4.7: Example of an Adjacency Matrix for the Hashtags from Table 4.6

Hashtag	preterm	prematurity	prematurebaby	babies	newborn	countrychildmag	preemies	kangaroomothercare
preterm	97	6	5	15	8	5	0	4
prematurity	0	95	6	3	6	5	0	0
prematurebaby	0	0	78	0	6	6	0	0
babies	0	0	0	64	6	0	0	0
newborn	0	0	0	0	49	0	0	0
countrychildmag	0	0	0	0	0	31	3	0
preemies	0	0	0	0	0	0	24	0
kangaroomothercare	0	0	0	0	0	0	0	12

Table 4.8: Converted Upper/Right Adjacency Matrix of Table 4.7.

The pseudo-code representation of the method described above to determine and weight the relations among the retrieved hashtags is shown in Algorithm 4.

Input: $HashtagQueue$
$Weights \leftarrow n * x$ matrix (where n is the size of $HashtagQueue$)
while $HashtagQueue$ *not empty* **do**
  $CurrentHT \leftarrow$ dequeue $HashtagQueue$
  $NewTweets \leftarrow$ set of retrieved tweets containing $CurrentHT$
  $NewHashtags \leftarrow$ hashtags in $NewTweets$
  $index \leftarrow 1$
  while $index <$ *number of* $HashtagQueue$ **do**
    $Weights[CurrentHT][index] \leftarrow$ number of times the hashtag in index appears in $NewTweets$
    $index \leftarrow index + 1$
  end
end
return $Weights$

Algorithm 4: Weighting the Relations of the Hashtags

We implemented the proposed process for directing and weighting the relations of hashtags in the `R` programming language, employing the package `igraph` [13], which provides a straightforward interface in `R` for representing networks through the drawing of graphs. The package allows the use of adjacency matrices and includes basic functions for easy handling of the data.

The results obtained by applying the described method are exemplified in Figure 4.2 which represents the network of relations of the collected hashtags for the *seed #Preterm*.

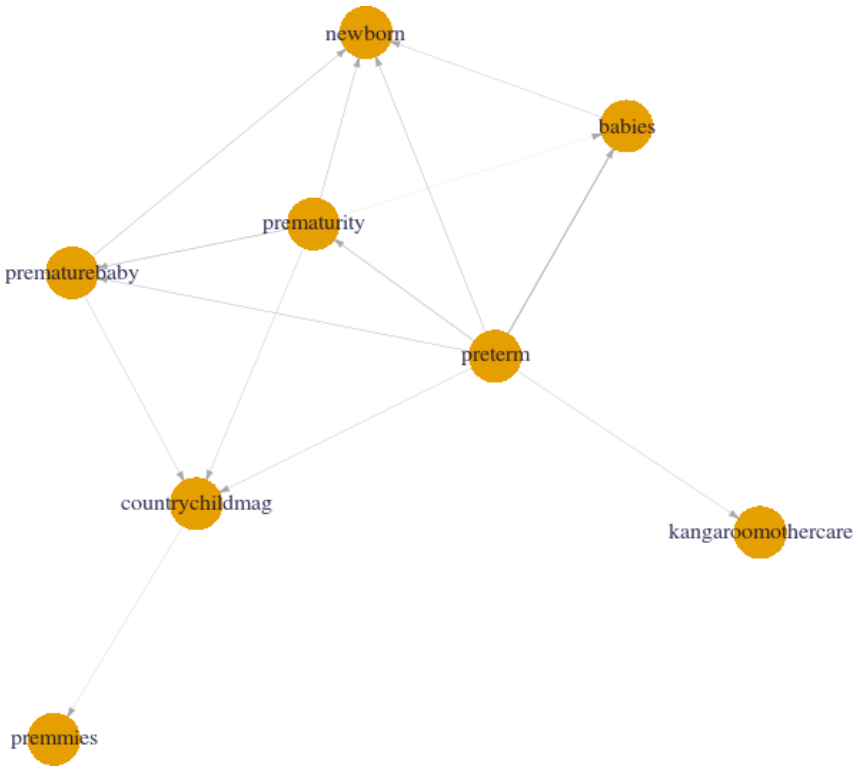


Figure 4.2: Representation of a Network of Hashtags Related to *#Preterm*

Chapter 5

Evaluation

The evaluation of a new method for hashtag recommendation is always a challenging task. As far as we know, there are no works in the current literature proposing a standard data set or establishing a general method of topic model evaluation and, even if there were any, our approaches for both data collection and hashtag recommendation are radically different from the previous methods, so achieving an appropriate comparison would be troublesome.

On the other hand, there are other complications that need to be considered when evaluating the performance of an hashtag recommendation system, due to its subjective character. Considering that English words are often ambiguous and that some hashtags can be simply acronyms or abbreviations, it is a non-trivial task to accurately measure how much a hashtag is relevant or not to a given topic, especially without providing any context.

Therefore, we focus on our objective of defining relevance to a topic as recognized by human beings to develop an appropriate mechanism for testing our approach. We decided to divide the evaluation process in two different stages: first, we test the performance of the classifier by experimenting with multiple combinations of the parameters and by comparing the results to a manual labeling, thus obtaining a quantitative evaluation; later, we examine the network of hashtags resultant from running our tool and analyze it in a qualitative way, comparing its performance with a commercial tool.

5.1 Quantitative Evaluation

Considering that the classifier is one of the distinctive components of TORHID, we need to evaluate its efficiency and estimate the best values for the possible variables. On the other hand, our algorithm for searching and collecting tweets can also accept different parameters and thresholds that may influence the final hashtags. In this section we will be experimenting with those different processes, subdividing them when possible.

Since our algorithm needs to identify relevant hashtags for multiple *seeds* in Twitter, we

decided to not limit our evaluation to one single topic. However, the addition of too many subjects exponentially increases the labor required for manual classification. For those reasons, we have chosen three *seeds* for our quantitative evaluation: *#preterm*, *#cerebralPalsy* and *#gunControl*.

5.1.1 Data set

While one of the advantages of TORHID is the collection of tweets in real time directly from Twitter, and consequentially not requiring the maintenance of large data sets, this approach becomes inconvenient when experimenting with different parameters and thresholds since the results naturally variate over time. As such, for this analysis we employed our data collection tool from Algorithm 1 to construct three data sets capable of representing segments of Twitter somewhat relevant to our *seeds* at a particular time, with the intention of using them to later simulate the direct searches.

We collected the initial data sets in the following manner: For each *seed*, over the course of two days, we employed Algorithm 1 to search 1000 hashtags (without minimum frequency restrictions) and collected up to 1000 tweets for each of those hashtags. Since the Twitter Search API imposes limitations for the age of the retrieved tweets, fetching exactly 1000 posts is unlikely for most hashtags. For this reason the three final data sets contained slightly more than 500.000 tweets each. Table 5.1 describes a summary of the data sets collected for each *seed*.

	Seed		
	#Preterm	#Cerebralpalsy	#Guncontrol
Number of Tweets	650555	977105	707329
Number of Terms	361071	482641	339443
Non-/Sparce Entries	8331441/234888212964	12591034/471578343271	9268436/240088609311
Sparcity	100%	100%	100%
Maximal term length	66	118	115
Number of Individual Hashtags	51827	66685	44710
Percentage of Tweets with Hashtags	80,90%	80,54%	78,07%
Percentage of Tweets with more than one Hashtag	61,69%	61,36%	56,61%
Percentage of Tweets without Hashtags	19,10%	19,45%	21,93%

Table 5.1: Summary of the Data Sets used for Evaluation

Due to our process of data collection, it was expected that all tweets would contain at least one hashtag. However, for undisclosed reasons, some of the posts collected through the Twitter Search API were not presented in their entirety, missing a few final characters or words. Nonetheless, TORHID consistently increased the percentage of collected tweets with hashtags from 20% or less in previous approaches [85][37][54] to approximately 80%. Furthermore, over half of the tweets ($\approx 59,89\%$) contained more than two hashtags, compared to less than 5% in previous methods [55][47]. The percentages of tweets per number of hashtags in the collected data sets are represented in Figure 5.1.

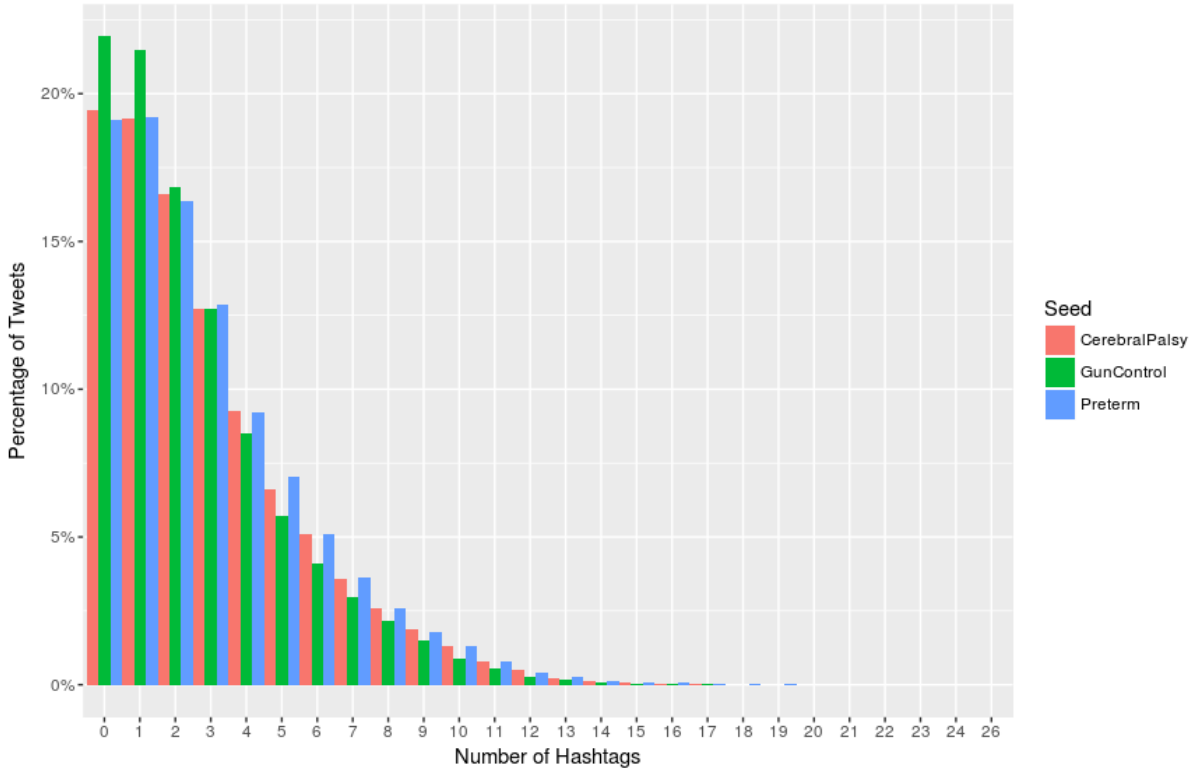


Figure 5.1: Amount of Tweets per number of Hashtags

5.1.1.1 SVM Classifier

The effectiveness and versatility of classifiers based on support vector machines is dependent of applying the correct kernels and respective variables for each circumstance. However, the conventional method of determining the best parameters for training a SVM is still trial and error. In this subsection, we experiment with combinations of different values for the kernel, the γ and the cost, as well as with the removal of the *seed* terms from the respective document term matrices.

We started by sampling the simulation data sets to generate train sets and test sets, with 30.000 tweets and 1.500 tweets respectively, for each topic. We proceed by manually labeling the tweets of the test sets into *relevant* or *irrelevant* to the corresponding *seeds*. Since analyzing tweets according to the topics is subjective, this task was performed by two persons independently. The classifications were subsequently compared and when the two persons did not agree, they discussed their interpretations until a consensus was reached. Despite manual labelling being a burdensome and tedious task, it is unquestionably the most accurate categorization we can produce for evaluating our classifier.

We selected four different kernels (*linear*, *polynomial*, *radial* and *sigmoid*) to test the support vector machine and three different values for experimenting with the γ (1, 10, 100) and the cost (1, 10, 100). Considering that γ and cost could be mutually dependent, we decided to test all combinations of the values. For each *seed*, we used LDA to build a model according

to the corpus, with 25 hidden topics, and used it to train the SVM for every combination of kernels and variables (except for the linear type of kernel that doesn't require γ and cost). The classifications of the tweets were later compared with the manual labeling done in previous stages for calculating the percentages of correct and incorrect classifications.

For a comparison metric, we developed a simple classifier that works in the following manner: every tweet containing the *seed* is labeled as *relevant* and every tweet not containing the *seed* is labeled as *irrelevant*. This classifications were also compared with the manual labels for calculating the percentages of right and wrong classifications. The graphical summary of the results for the experiments with the different variables are described in Figure 5.2 for the *seed* *#Preterm*, in Figure 5.3 for the *seed* *#CerebralPalsy* and in Figure 5.4 for the *seed* *#GunControl*, with the dashed black lines representing the threshold of the alternative algorithm and the colors green, red and blue representing the percentage of correct classifications, false negatives and false positives respectively.

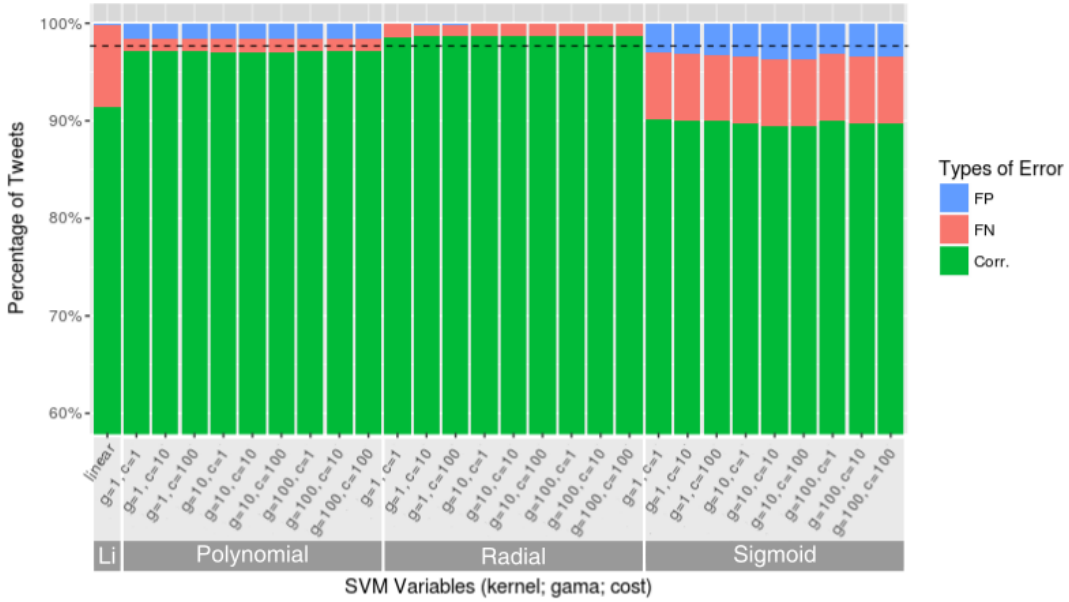


Figure 5.2: Percentage of Types of Error with Different SVM Variables for the Hashtag *#Preterm*.

According to our tests, it seems apparent that a radial type kernel performs better for our use case of finding relevant tweets than the other types of kernel, since it generally provides more accurate and more consistent results, always surpassing the alternative of only considering the presence of the *seed* in the text. On the other hand, the variations in the results of experimenting with different values for γ and cost are usually negligible, particularly for the radial kernel where using a combination of $\gamma = 1$ and $cost = 100$ consistently improved the number of correct classifications, but by an insignificant margin. Finally, the percentage of false positives is consistently very low, which results in great precision values. This is important for the overall reliability of the algorithm since it is expected to translate into a smaller probability of returning hashtags unrelated to the seed.

A shortcoming of TORHID that is not expressed in this experiment is the amount of time

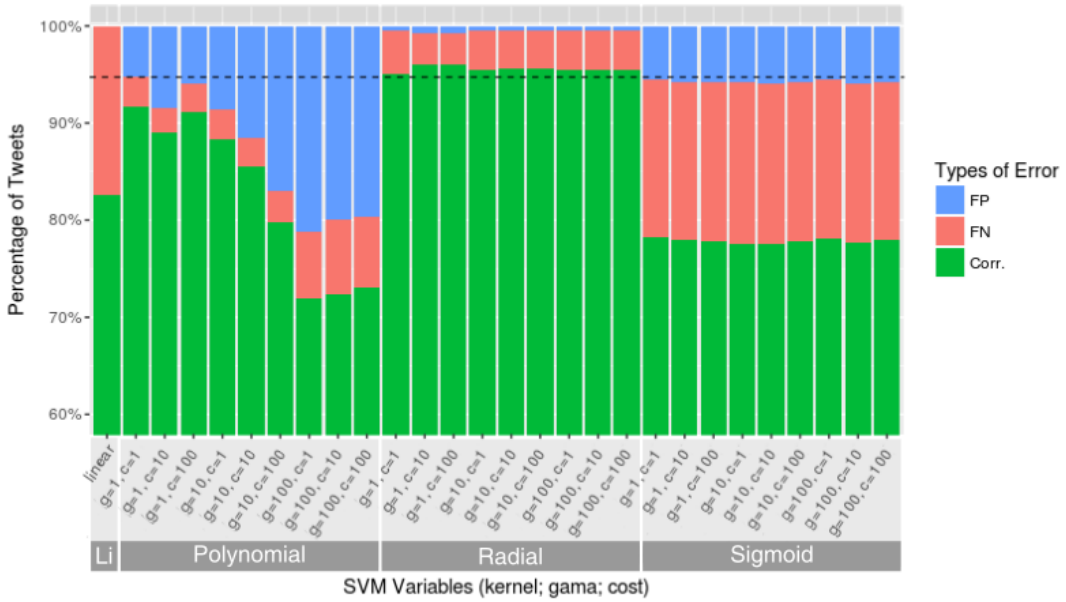


Figure 5.3: Percentage of Types of Error with Different SVM Variables for the Hashtag *#CerebralPalsy*.

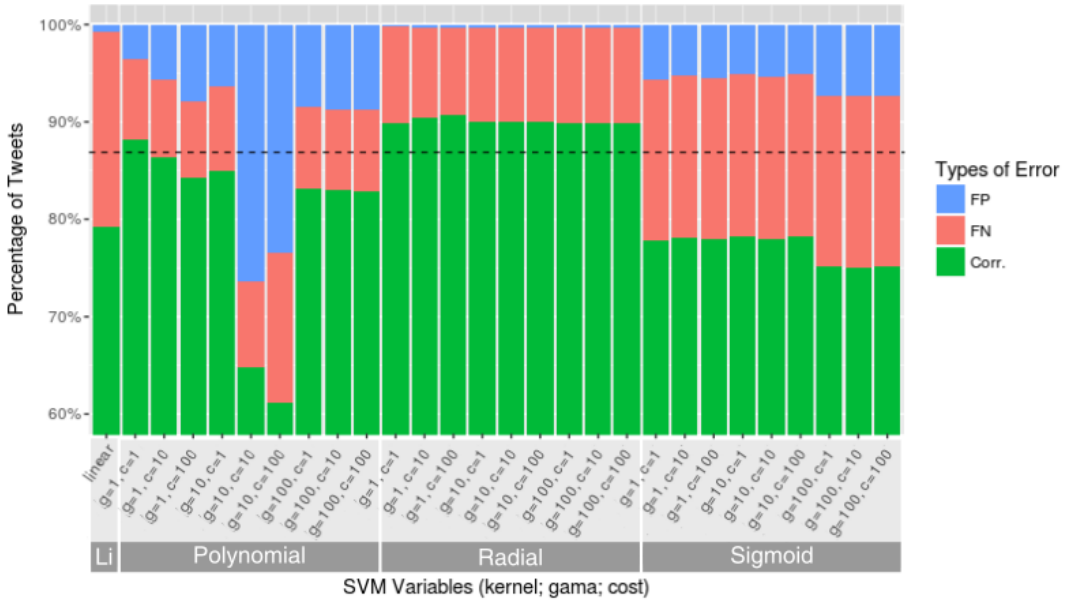


Figure 5.4: Percentage of Types of Error with Different SVM Variables for the Hashtag *#GunControl*.

required for training. Depending on the size of the training data, TORHID may need up to an hour to train the topic model and the support vector machine. When using the right kernel, the classifications provided by TORHID were consistently superior to the alternative of only considering the presence of the *seed* in the text. However, this alternative method was always ready for classifying the tweets faster than our tool, since it does not require a previous step for model training. Furthermore, since the use of a train set would be pointless for this method,

the procedure of data collection can be ignored, additionally saving up to several days. For this reason, employing the simple classifier method could be sufficient in cases where a faster execution time would be prioritized over the better results achieved by TORHID.

In the next step, we test the dependency of the classification of each tweet on the presence or not of the *seed*. That is, if removing the *seed* term from a relevant tweet would make the classifier incorrectly label it as *irrelevant* instead. It is expected that, by removing the *seed*, the classifier may become less biased towards it. It is important to notice that, since the corpus was preprocessed, all the octothorps were removed, rendering the hashtags indistinguishable from the regular words which comprise them.

For this task, we first edited the document term matrices of the train and the test sets of each topic, removed the term corresponding to the stemmed *seeds* and proceeded to repeat the previous tests with exactly the same kernels and variables combinations. The graphical summary of these results are described in Figure 5.5 for the *seed #Preterm*, in Figure 5.6 for the *seed #CerebralPalsy* and in Figure 5.7 for the *seed #GunControl*, once again with the dashed black lines representing the threshold of the alternative algorithm that only considers the presence of the *seed* (obviously not removed) and the colors green, red and blue representing the percentage of correct classifications, false negatives and false positives respectively.

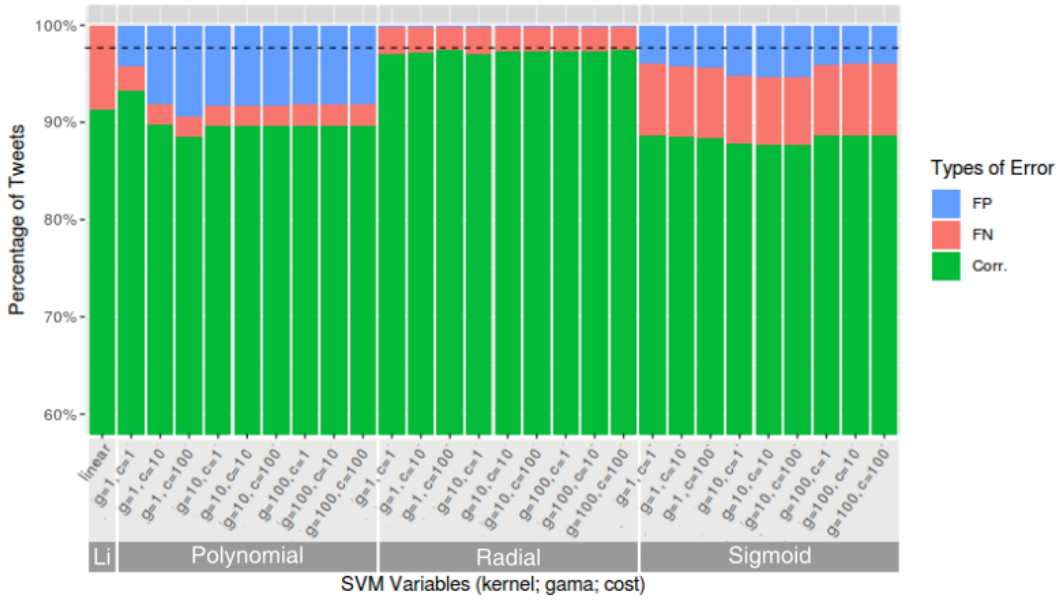


Figure 5.5: Results for Different Combinations of SVM Variables Without *Seed* for the Hashtag *#Preterm*.

It is immediately noticeable that the support vector machines without access to the *seeds* provided inferior classifications, not only when compared to previous results where the *seeds* were maintained in the document term matrices, but also when compared to the alternative algorithm that only considers the presence of the *seed*. This fact is probably due to the higher chance of the model to classify most tweets with the seed hashtag as positive. Despite the lower percentage of correct classifications, the results were consistent with the previous experiments,

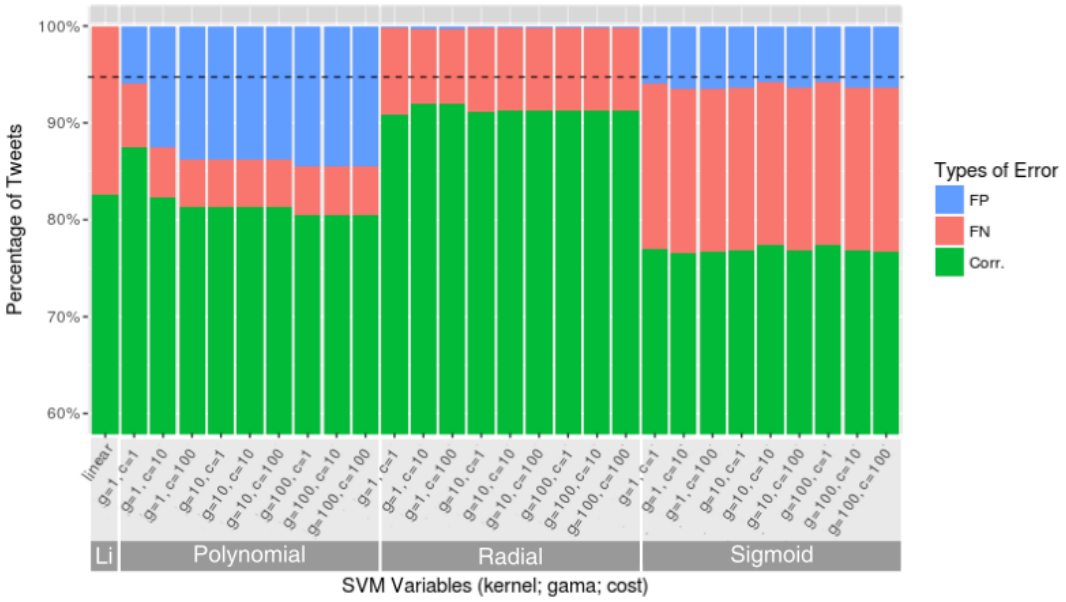


Figure 5.6: Results for Different Combinations of SVM Variables Without *Seed* for the Hashtag *#CerebralPalsy*.

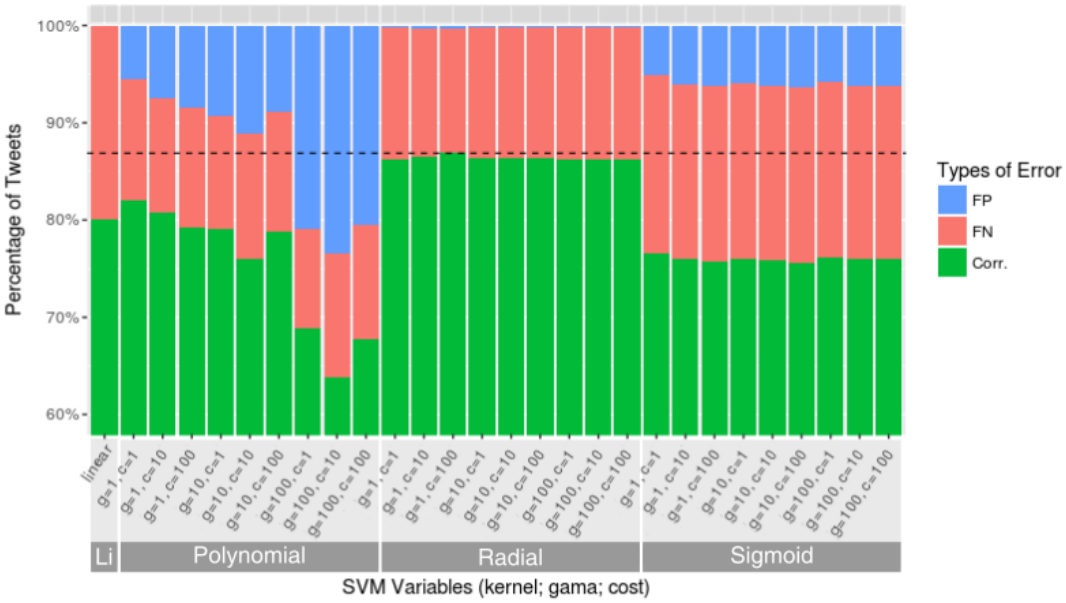


Figure 5.7: Results for Different Combinations of SVM Variables Without *Seed* for the Hashtag *#GunControl*.

with the radial type kernel proving to be more accurate than the alternatives and the different γ-cost combinations failed to significantly impact the classifications for most cases.

While the removal of the *seed* from the classifier resulted in worse classifications, the results were still comparable to the baseline, specially for *#Preterm* and *#Guncontrol*. This shows that while TORHID definitely benefits from the presence of the *seed* in the tweets, it is not entirely

dependent on that information to determine the relevance for the topic of interest.

Overall, the results from these experiments show that a radial type kernel with $\gamma = 1$ and $cost = 100$ will generally achieve the best classifications and that removing the *seed* from the document term matrix results in a deterioration of the results.

5.1.1.2 LDA Model

Since the generated topic model based on Latent Dirichlet Allocation influences the train data of the support vector machine, it is also important to be tested. In this subsection, we experiment with different values for the number of hidden topics and with different sizes of samples for the training sets.

We started by sampling the simulation data sets to generate different train sets with 5.000, 10.000, 30.000 and 50.000 tweets for each *seed*. We also decided to use the same test sets as used in the previous tests to maintain consistency among the experiments.

For each train set, we employed Latent Dirichlet Allocation to build multiple topic models according to the respective corpus, using different numbers of latent topics (5, 10, 25, 50, 75). Next, the topic models were used to train the support vector machines, with a radial kernel and associated $\gamma = 1$ and $cost = 100$, and classify the tweets from the test set. As before, the classifications of the tweets were compared with the manual labels and the percentages of correct and incorrect classifications were calculated.

Next, the results were grouped according to the size of the train sets (5.000, 10.000, 30.000 and 50.000 tweets) and the averages of the values were calculated accordingly. The graphical summary of the results of modeling the LDA with different sized samples is described in Figure 5.8 for the *seeds #Preterm*, *#CerebralPalsy* and *#GunControl* with the colors green, red and blue representing the percentage of correct classifications, false negatives and false positives respectively.

The results are consistent across the three different *seeds*: the proportion of correct classifications increases with the size of the train sets. However, the increase of performance seems to be logarithmic instead of linear, being significantly lower when expanding the train set from 30.000 tweets to 50.000, compared to expanding the train set from 5.000 tweets to 10.000. Nonetheless, this may be due to the closed approach of our tests and bigger train sets may prove to be beneficial in long term solutions where an higher ratio of relevant tweets could be maintained and older tweets would be mixed with new ones over time.

The results were then regrouped according to the number of hidden topics (5, 10, 25, 50, 75) and the averages of the values were again calculated accordingly. The number of topics is usualy a trade-off between a too general model that has few topics and a very specific model which needs a lot of training data to avoid overfitting. The graphical summary of the results of modeling the LDA with different sized samples is depicted in Figure 5.9 for the *seeds #Preterm*,

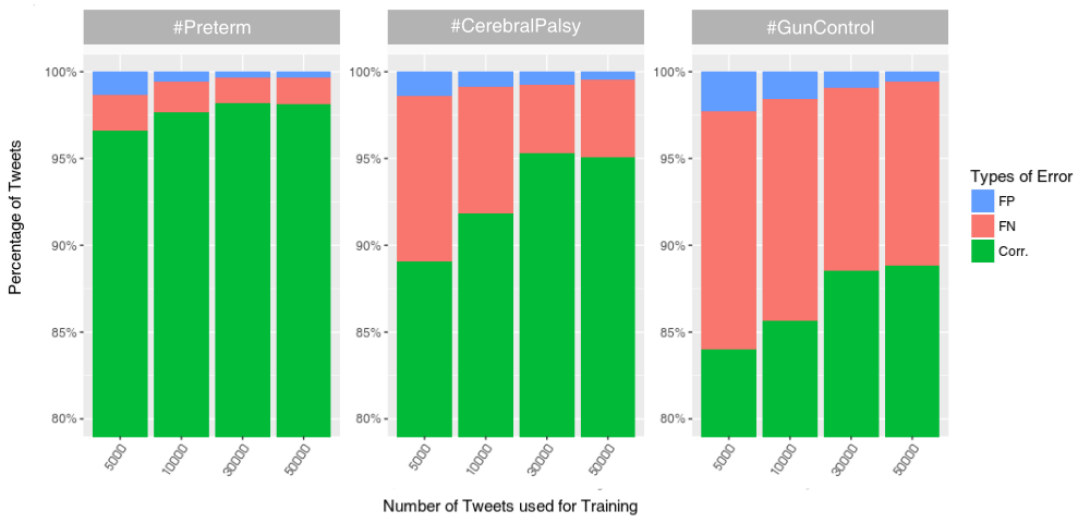


Figure 5.8: Average of the Percentages of the Types of Error for Different Sizes of Train Sets and Different *Seeds*.

#CerebralPalsy and *#GunControl* with the colors green, red and blue representing the percentage of correct classifications, false negatives and false positives respectively.

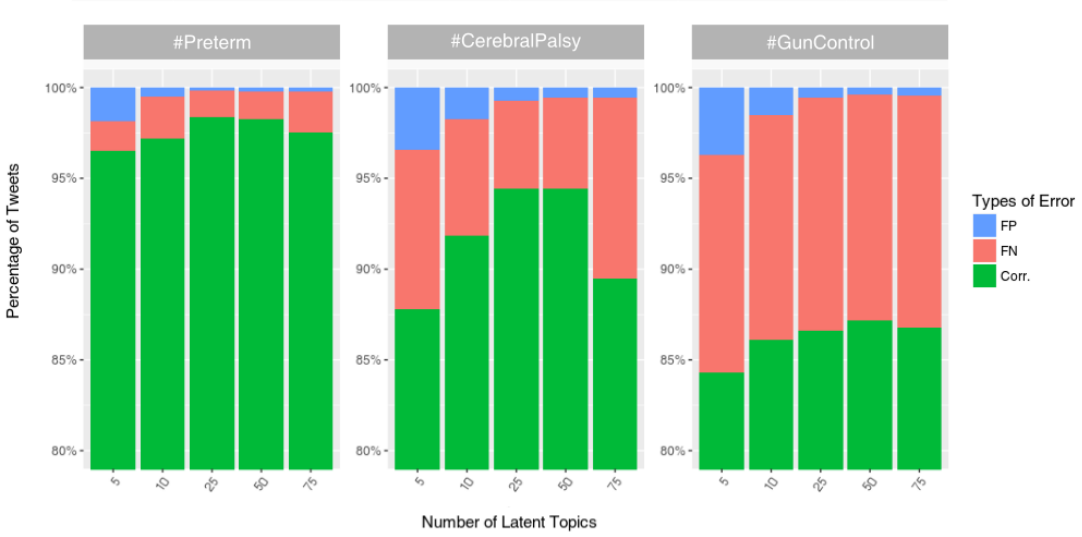


Figure 5.9: Average of the Percentages of the Types of Error for Different Numbers of Latent Topics and Different *Seeds*.

Again, the results are consistent across the three different *seeds*: the proportion of correct classifications increases with the number of latent topics until 25, decreasing in again at 75 hidden topics. In this manner, the results suggest an ideal number of topics between 25 and 50 to generate the LDA model. It is expected that smaller train sets would benefit from less latent topics and larger train sets from more.

Generally, the results from these experiments show that the performance of the classifier

increases with the amount of training data and, in this particular case, a train set with 30.000 is already enough for achieving positive results. On the other hand, the best number of latent topics seems to be approximately between 25 and 50 in every case.

5.1.1.3 Search Variables

After determining the best parameter for the classifier, we proceed to test the variables of another key component of our solution: the search mechanism. While the classifier is important to filter unrelated hashtags, the search element of the algorithm is responsible for the actual collection of the tweets, as well as a simple selection based on frequency thresholds. In this subsection, we experiment with different values for the number of search iterations, the quantity of tweets collected in each search and the minimum frequency for hashtags being considered relevant.

We started by training the classifiers with the best performing parameters according to the previous tests: for each *seed*, the same train sets with 30.000 tweets were employed to generate the respective topic models with 25 hidden topics and the support vector machines were trained with a radial kernel, $\gamma = 1$ and $cost = 100$.

Next, we simulated the execution of TORHID in a controlled scenario by running multiple searches for each *seed* in the respective data sets. The tool was configured to collect a maximum of 250 tweets for each search and the threshold for minimum hashtag frequency was set to 3, while we experimented with different numbers of iterations: 1, 5, 15, 25 and 35. The hashtags retrieved in each execution were later manually labeled as relevant or irrelevant to the respective *seed* by four different people independently and compared to reach a consensus. It was established that, this time, too general hashtags would be labeled *irrelevant* (despite generality also being somewhat subjective). The results of searching with a different number of iterations are summarized in Figure 5.10 for the *seeds #Preterm* represented in blue, *#CerebralPalsy* in red and *#GunControl* in green, while the solid lines represent the relevant hashtags and the dashed lines represent the irrelevant ones.

While the total number varies, the amount of relevant hashtags is always superior to the number of irrelevant hashtags. Also, raising the number of iterations in some cases may increase the number of hashtags retrieved. However, this pattern occurs at a very small rate and stops increasing at a peak with 25 iterations. This is probably due to the most relevant hashtags being closer to the *seed* in the relations network and the algorithm never collected more than 14 hashtags in total. In this case, we consider that 15 iterations provided the most satisfactory results, being the lower number to collect the maximum relevant hashtags in total.

We proceeded to investigate the impact of the number of tweets collected by each search. Using the same classifiers and data sets as in the previous test, we repeated the simulations with 15 search iterations and a fixed threshold of a minimum frequency of 3 times for each hashtags, while experimenting with the number of collected tweets: 50, 150, 250 and 350. The hashtags retrieved in each execution were again manually labeled and compared through the same process

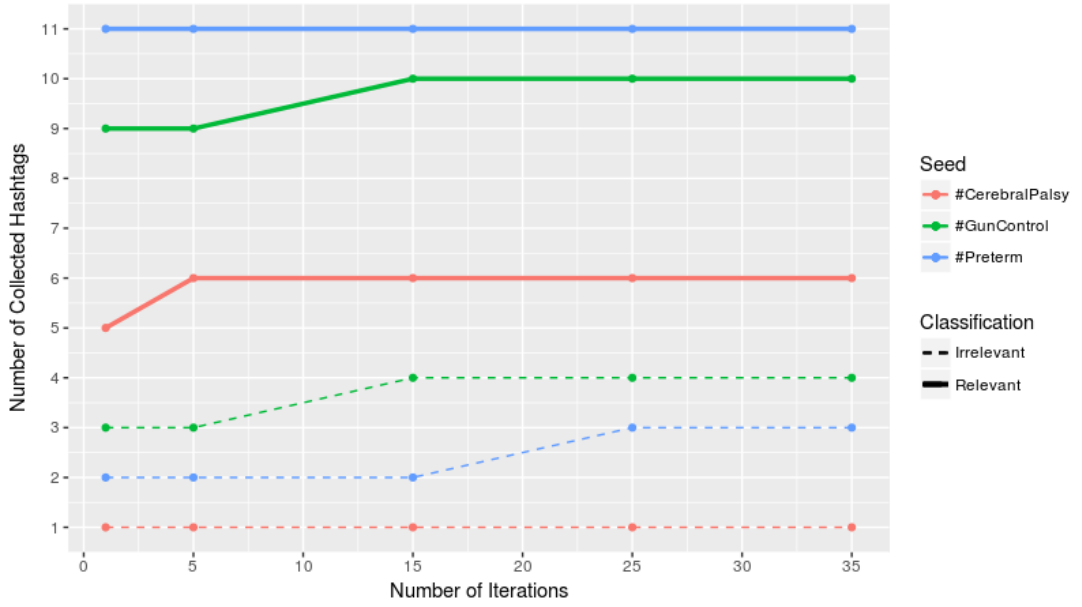


Figure 5.10: Number of Hashtags Collected with Different Number of Iterations for Different *Seeds*.

as before. The results of searching different numbers of tweets per iteration are summarized in Figure 5.11 for the *seeds #Preterm* represented in blue, *#CerebralPalsy* in red and *#GunControl* in green, while the solid lines represent the relevant hashtags and the dashed lines represent the irrelevant ones.

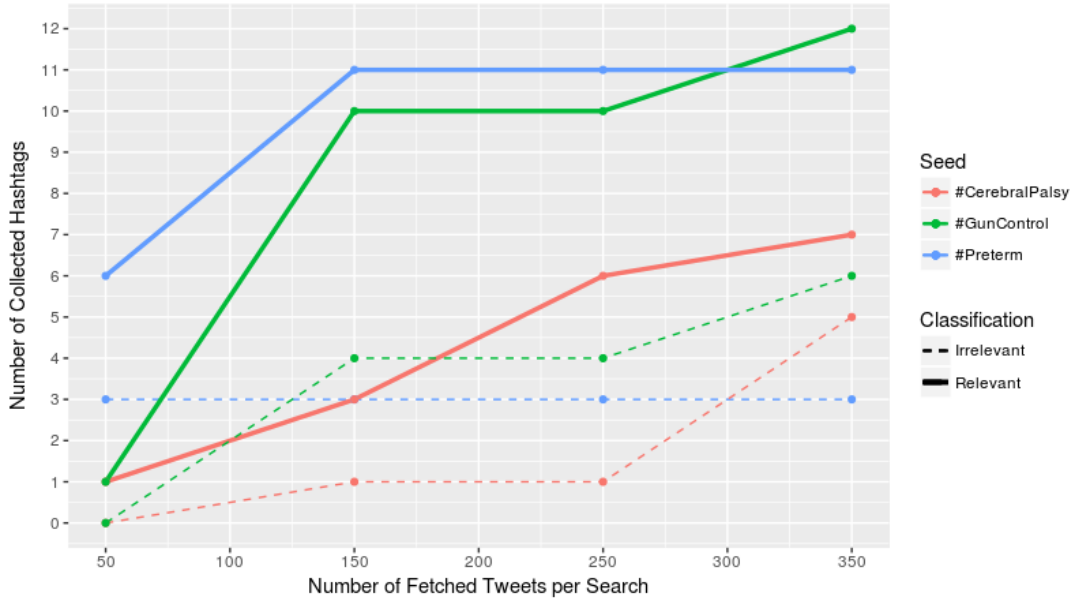


Figure 5.11: Number of Hashtags Collected with Different Number of Tweets per Iterations for Different *Seeds*.

As before, the number of relevant hashtags remains superior to the number of irrelevant ones. Nonetheless, it is noticeable that the number of collected hashtags regularly increases with

the number of collected tweets. This increase is expected since the hashtags are contained in the tweets and thus collecting more different tweets corresponds naturally to collecting more different hashtags. While the number of hashtags for *#CerebralPalsy* and *#GunControl* keeps increasing almost linearly, for *#Preterm* it stops when it reaches 150 tweets, what may be a sign of scarceness or high similarity in the tweets collected for this *seed*.

Finally, we examine the influence of the hashtag frequency threshold. Once again, we employ the same classifiers and data sets and repeat the simulations, this time with 15 search iterations collecting 250 tweets, while experimenting different frequency thresholds: 1, 3, 5, 7, 9 and 11. The hashtags retrieved in each execution were also manually labeled and compared through the same process as before. The results of varying the minimum frequency for considering hashtags relevant are summarized in Figure 5.12 for the *seeds #Preterm* represented in blue, *#CerebralPalsy* in red and *#GunControl* in green, while the solid lines represent the relevant hashtags and the dashed lines represent the irrelevant ones.

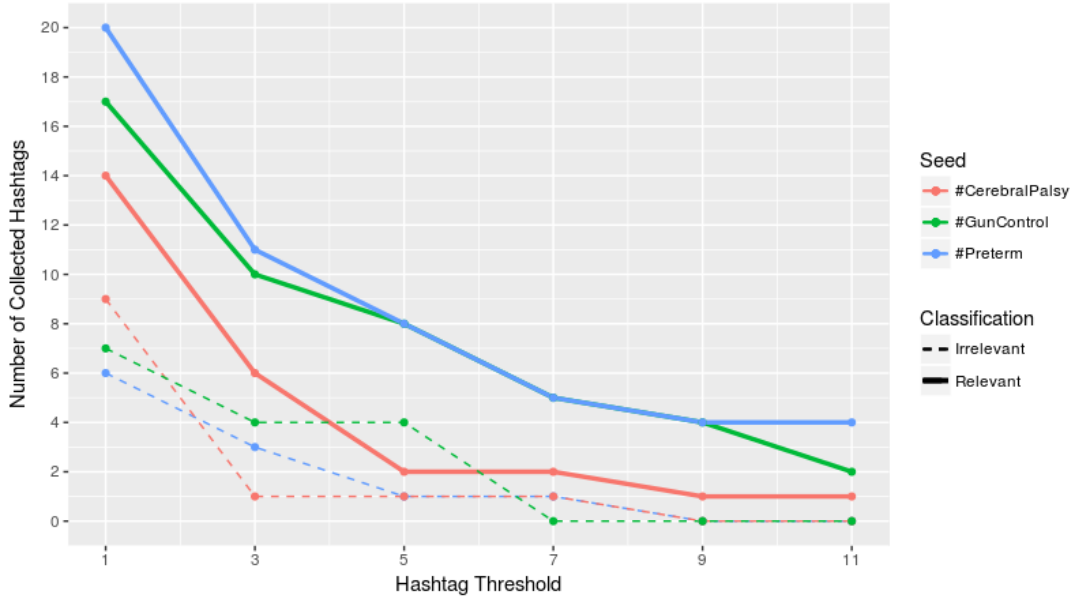


Figure 5.12: Number of Hashtags Collected with Different Frequency Thresholds for Different *Seeds*.

Considering the proportions of relevant and irrelevant hashtags, the results remain consistent with the previous ones. However, this time we experience an exponential decay while increasing the frequency threshold. This results are expected since tweets are limited to 280 characters, forcing users to select fewer hashtags. Furthermore, our tests suggest that increasing the frequency threshold may filter the hashtags by degree of relevance to the *seed* since setting its value to 9 resulted in the retrieval of only relevant hashtags, albeit in smaller quantity. With a threshold large enough, only the *seed* would be returned.

In addition, commonly to all experiments in these sections, the number of hashtags collected for *#Preterm* and *#GunControl* is always higher than for the hashtag *#CerebralPalsy*. This may be due to cerebral palsy being a sensible theme, what may result in less discussions and

shares on Twitter, leading to the tweets containing the hashtag being scarcer and similar to each other. On the other hand, gun control is currently a major discussion topic in the United States of America and preterm births, while not a popular conversation theme for the average person, performed well in our tests due to charity campaigns ongoing when the tweets were collected.

5.2 Qualitative Evaluation

The method to qualitatively evaluate TORHID consists in the comparison of its returned hashtags with the alternatives depicted in the related work section. Ideally, we would use similar data sets to train our classifier and implementations of different approaches of the current literature, in order to compare the results. However, this arrangement is impractical in this case, since the existing methods that recommend hashtags for a topic or a keyword are very scarce. Furthermore, as far as we know, no other method approaches data collection in the same way as we do, thus making the use of the same data set for training and comparing both systems inaccurate. Therefore, we decided to follow a different approach and perform a qualitative comparison with a commercial grade recommendation system.

Despite most of the current algorithms for hashtag recommendation differing from ours in their purpose and operation, the main goal of *Hashtagify.me* resembles ours and the hashtags collected by both approaches can be conveniently compared. As previously mentioned in Chapter 3, *Hashtagify.me* is a commercial web service, with some features available for free, whose purpose is to allow its clients to search for related hashtags and influencers in order to improve their social media strategy. The free version currently limits the user to search for a hashtag and retrieve a word cloud with the 10 most related hashtags. Other features are advertised in beta for clients with a payed subscription, but are currently unavailable.

Since *Hashtagify.me* constitutes a business, its source code and algorithms are closed source and so, unlike with the previously proposed approach, we were forced to employ a black-box testing method to evaluate and compare their service with TORHID.

Once again, since our algorithm needs to identify relevant hashtags for diversified topics in Twitter, we opted to not limit our evaluation to one single *seed*. This time, however, comparing both methods was less burdensome and we decided to expand our previous selection of topics to twenty different hashtags from both trending and more obscure subjects. From this new listing, fourteen hashtags were single words, three were concatenations of words and three were acronyms. The full list of topics is disclosed in Table 5.2.

Alzheimer	Eurovision	Lisbon	NBA	Sunset
Apple	Gaza	Microsoft	Nokia	Tumor
Bitcoin	Guncontrol	Moneyconf	Preterm	WWE
Cerebralpalsy	InfinityWar	MLB	Spring	Workout

Table 5.2: *Seeds* Explored for the Qualitative Evaluation

5.2.1 Data Collection

The data collection was performed in the following manner: for each topic, we used our tool to collect the corresponding relevant hashtags and represent their relations in a graph. At the same time, we searched the same hashtag in *Hashtagify.me* and stored the retrieved word cloud. The hashtags collected by both approaches were collected for later categorization. Figure 5.13 compares a word cloud returned by *Hashtagify.me* and a collected by TORHID.

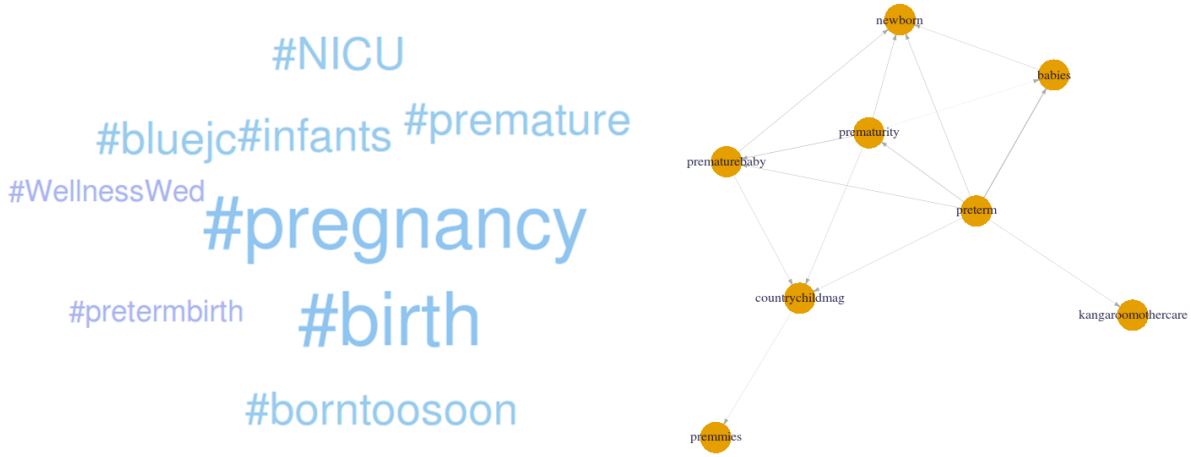


Figure 5.13: Examples from *Hashtagify.me* (left) and from TORHID (right) for the *seed #Preterm*

5.2.2 Comparison and Results

Since the relevance of a hashtag to a given topic is subjective, specially without providing context, the task was divided by multiple individuals. Instructions were given to four people to independently classify each hashtag of the obtained network and word cloud as *relevant*, *irrelevant* or *neutral* to the main topic. Searching the meaning of a hashtag was not forbidden or discouraged and, in total, 449 hashtags were manually classified by each individual. The summaries of these evaluations are described in Table 5.3, comparing the means and standard deviations of the results from both approaches.

Despite the percentage of *neutral* hashtags being very proximate for both tools, our hashtags were classified as 7.35% more *relevant* and 9.89% less *irrelevant* than the ones from *Hashtagify.me*. Usually, those percentage differences would not translate to a tremendous improvement on themselves. However, *Hashtagify.me* rarely succeeded in retrieving 10 hashtags as selected (only accomplished that goal three times out of twenty) while TORHID collected more than 10 hashtags in the majority of times.

On average, our technique collected 15 hashtags for each topic, out of which 10.09 were considered relevant, while *Hashtagify.me* retrieved on average 7.45 hashtags, out of which 4.46 were considered relevant. Ultimately, both results were satisfactory, particularly when considering

	TORHID		Hashtagify.me	
	Arithmetic Mean	σ	Arithmetic Mean	σ
Relevant	201,75	9,639	89,25	4,573
Irrelevant	33,75	11,529	31,5	11,091
Neutral	64,5	12,557	28,25	7,805
Total	300	N/A	149	N/A
Relevant (%)	67,25%	0,032	59,90%	0,031
Irrelevant (%)	11,25%	0,038	21,14%	0,074
Neutral (%)	21,50%	0,042	18,96%	0,052

Table 5.3: Comparison of the Results Collected by TORHID and *Hashtagify.me*

the low ratio of false positives, that is, irrelevant hashtags. However, due to the disparity in the average number of hashtags in these tests, it was considered that TORHID outperformed *Hashtagify.me*.

5.2.3 Temporal Relevance

One objective of TORHID not directly examined with the evaluation process described above is the temporal relevance of the hashtags for the topic. While some hashtags remain relevant and popular indefinitely, others, specially ones referring to particular or annual events, are only relevant during a short period of time. For this reason, unless the tweet data set is properly maintained and frequently updated, hashtag recommendation systems may suggest hashtags that have been appropriate in the past but are currently unrelated to the *seed*.

The temporal character of the results is important since the recency of the collected hashtags may be decisive for some applications like in information retrieval. Therefore, we followed by experimenting a variation of the previous evaluation with some extra considerations but in much lower extent because of the added difficulty in accurate classification.

The adjustments to the evaluation procedure described above were the following: this time, only the three topics used for the quantitative evaluation were chosen and only two people independently classified each hashtag. Also, the hashtags were labeled in the following manner: *relevant* if they immediately had direct bearing on the topic (or were was a subtopic); *neutral* if they were connected to the topic but too general (or were a supertopic); *irrelevant* if they were completely unrelated to the given topic; or *contextual* in cases when the hashtags seemed irrelevant at first sight, but after searching for the subject they were found to be relevant (at least for that particular given time). When the two people did not agree, they discussed the classifications until a consensus was reached. The hashtags collected by each method, as well as their classifications are described side by side in Table 5.4 for the topic *#Preterm*, in Table 5.5 for the topic *#CerebralPalsy* and Table 5.6 for the topic *#GunControl*.

#Preterm		
Classification	**TORHID**	**Hashtagify.me**
Relevant	prematurebaby, prematurity, premmies	borntoosoon, premature, pretermbirth
Contextual	countrychildmag, kangaroomothercare	nicu
Neutral	babies, newborn	birth, infants, pregnancy
Irrelevant	—	bluecj, wellnesswed
Total	7	9

Table 5.4: Comparison of the Hashtags Collected for the Topic *#Preterm*

#CerebralPalsy		
Classification	**TORHID**	**Hashtagify.me**
Relevant	cerebralpalsyawerenessmonth, disabilityrights	cp, disability, disabled
Contextual	westham	hyperbaric
Neutral	autism	autism
Irrelevant	writer	inspiration
Total	5	6

Table 5.5: Comparison of the Hashtags Collected for the Topic *#CerebralPalsy*

At first glance, both approaches seemed to retrieve similar hashtags. However, after a closer inspection, it is noticeable that only two pairs, from a total of the forty three hashtags returned (considering the three topics combined), were exactly the same for the two methods. This phenomenon is probably due to the way our tool works with more recent data compared to *Hashtagify.me* which presumably relies in a large longstanding database maintained by the company.

When searching the *seeds #Preterm* and *#CerebralPalsy*, TORHID returned less hashtags than *Hashtagify.me*. This may be due to subjects related to medical conditions being too intimate for being shared (personal experiences) or too specialized for becoming popular (medical findings), thus benefiting from long term data. However, when discarding the irrelevant hashtags, TORHID stays on pair with *Hashtagify.me*. Hashtags like *#wellnesswed*, *#writer* and *#inspiration* expressed contexts too general for being considered related to the topics and *#bluecj* was an obscure hashtag, to the point of a web search on the term getting almost no results.

#GunControl		
Classification	**TORHID**	**Hashtagify.me**
Relevant	assaultweaponsban, boycottthenra, guncontrolnever, gunsense	floridaschoolshooting, nra
Contextual	boycott, marchforo, neveragain, oregon, parkland, parklandstudents, wecallbs	neveragain, parklandstudentsspeak
Neutral	—	—
Irrelevant	—	—
Total	11	4

Table 5.6: Comparison of the Hashtags Collected for the Topic *#GunControl*

On the other hand, when searching for *#GunControl* our results were significantly better than *Hashtagify*'s. While none of the approaches retrieved any irrelevant or even neutral hashtags, TORHID collected more than the double of the hashtags retrieved by *Hashtagify.me*, although *#neveragain* was a common hashtag to both systems. This is probably due to the character of the topic being currently very popular and publicly debated in the United States of America (the hashtags were collected a few days after the Stoneman Douglas High School shooting[39]), what benefits a short term and more embracing search.

Contextual results are particularly important for TORHID, since they are composed of hashtags which either are not immediately obvious to the average person or whose relevancy to the topic is variable within time and context, i.e. hashtags that are recent. This means that, while *#Westham* is an association football team, at that point in time they were relevant to the topic *#CerebralPalsy* because of their campaign during the Cerebral Palsy Awareness Month to gather funds to help support people with the condition. Meanwhile, *#Parkland* became relevant to the topic *#GunControl* because of the infamous school shooting which toke place in the city days before we searched for relevant hashtags. Both of these hashtags, as well as the majority of the ones classified as contextual, will probably cease to be relevant to these topics in the future but are pertinent right now, so they are entitled to a different classification. Considering that our tool searches Twitter for the latest tweets every time, it has an advantage in returning contextual hashtags when compared to methods that rely on longstanding databases, like *Hashtagify* which take longer to be updated.

Overall, we consider that our results outperform the ones provided by *Hashtagify.me*, with a similar number of relevant hashtags and a higher number of contextual ones. TORHID also has the advantages of being free to implement, of requiring a smaller database and of collecting more up to date hashtags, while also returning a network with the relations among each other. On the other hand, the dependency of *Hashtagify.me* in a extensive and longstanding database brings some advantages, particularly in the short time required to answer a query to their database (few seconds), while TORHID may need up to an hour to train the classifier, in addition to the necessary time to collect the train set. Since the hashtags retrieved by both approaches were generally different, using both methods when searching Twitter for hashtags relevant to a topic could be advantageous for some people.

Chapter 6

The TORHID Application

In Chapter 5 we observed that TORHID can successfully recommend relevant hashtags to a given topic, generally outperforming or at least matching the capabilities of commercially available platforms like *Hashtagify.me*. However, the tool we implemented carries some usability drawbacks: the variety of parameters for data collection and training, as well as the need of installing the `R` software environment for running the program through a script may be overwhelming for novice users; also, the process of collecting new tweets for every execution, while carrying its advantages, makes the overall operation slower. This shortcomings make the mentioned commercial platforms, that can be simply accessed through a website, much more convenient.

In this chapter, we will focus on improving the intuitiveness of our tool through the addition of an interactive web application. We will also adjust the variables and attempt to reduce the running time of the process and analyze the impact of the modifications.

6.1 Web Application

The first step in improving the usability of our tool is providing a graphical interface to the user, in order to assist regular people to operate it without having to learn how to code as well as understanding the ins and outs of our method.

Since TORHID was developed and tested within the `R` software environment, it would be burdensome to port it to a different programming language and repeat all the tests in order to add a user interface. Instead, we used the *Shiny* software package.

`Shiny` is an open source `R` package developed by RStudio that provides a straightforward but capable web framework for developing web applications based on `R` code. Using this package, we were able to convert TORHID into an interactive app with a familiar design and uncomplicated operation, conveniently accessible through a browser environment.

With simplicity in mind, by default the user interface of our application only provides a field for typing the *seed* and a search button to initialize the process. A navigation bar was

added for providing further information about the application, like the motivation behind it and some details about the different options. Considering the limitations imposed by Twitter in the Search API, we decided to add the option for using different authentication tokens, as well as instructions to acquire them. Figure 6.1 depicts the default user interface of our web application running in Mozilla Firefox.

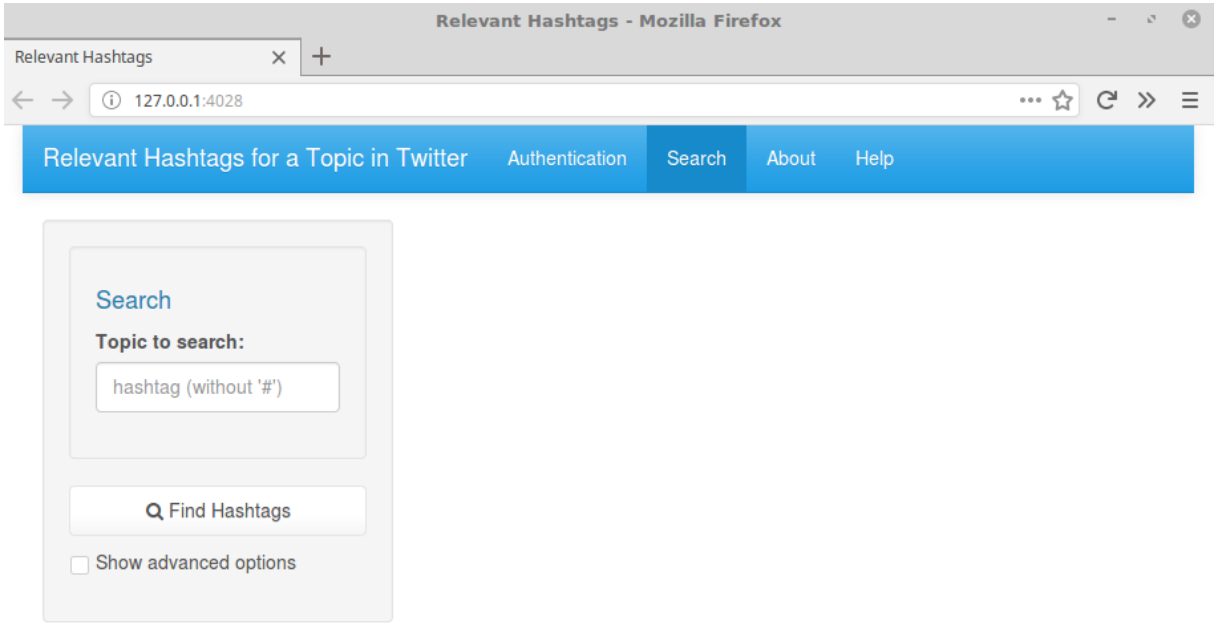


Figure 6.1: User Interface of the Shiny Application Running in Mozilla Firefox

While in Chapter 5 we estimated the best parameters for training the classifier, these values may change over time due to factors like the popularity of the *seeds*. Also, other variables, like the size of the train set or the number of collected tweets, may be tweaked in order to achieve more accurate or faster results. For these reasons, all the different variables and options described through this thesis are available by checking the box for *showing advanced options*, providing the tweaks required by a skillful user to achieve the best results. Figure 6.2 illustrates the advanced options of the application, as well as the resultant network of hashtags collected for the *seed* *#Preterm*.

Since the installation of the `R` software environment may also be burdensome for less experienced people, distributing TORHID in the cloud could also be advantageous. While we did not follow this path yet, RStudio offers a few possibilities for easily publishing *Shiny* web applications online, allowing developers to deploy their own *Shiny* server or host the website in the service *shinyapps.io*.

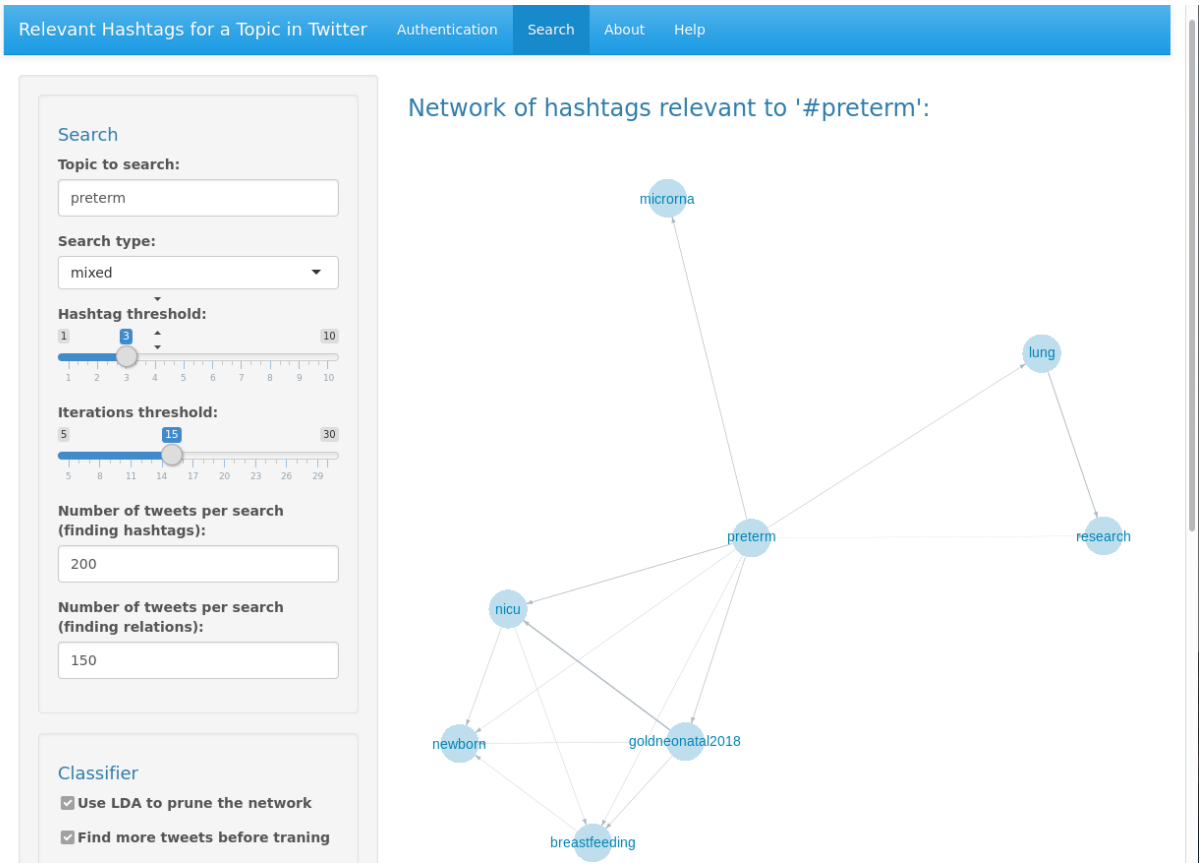


Figure 6.2: User Interface of the Shiny Application with Advanced Options and Results

6.2 F-TORHID

While the supplementary user interface presented in the previous section uncomplicates the use of TORHID and we have already determined that the hashtags returned by our tool are at least on pair with the available alternatives, better results and good usability are not always a guarantee of being the best solution. Since social media is constantly in motion, rapidness may in some cases outdo accuracy and, unfortunately, our process is slower than the presented alternatives. Therefore, in this section, we propose F-TORHID (Faster Topic Relevant Hashtag Identification) a variation of our algorithm with the purpose of achieving better temporal performances, at a possible expense of precision.

After carefully analyzing our method, we concluded that the most negative impacts in temporal performance are caused by the following processes:

Initial Data Collection - despite requiring the collection of significantly less training data than the alternative approaches, this step is still too prolonged and may take up to days for being completed; while when doing research on the topic of interest it is important to collect a comprehensive train set, for general cases it may be unnecessary;

Training the Classifier - topic models and support vector machines require heavy calculations

and plenty of time to be properly trained; our classifier relies on Latent Dirichlet Allocation, whose training time increases with the size of the provided corpus and the number of hidden topics;

Querying Twitter - despite the collection of posts through the Twitter Search API decreasing the number of tweets without hashtags and enabling TORHID to collect less irrelevant tweets (ultimately saving some time) the limitations imposed by Twitter force TORHID to pause several times for intervals of 15 minutes; during these pauses the resources are being entirely wasted;

Considering that the combination of support vector machines with Latent Dirichlet Allocation is the groundwork of our approach and this type of classifiers needs labeled data to be trained, we cannot simply attempt to discard any of the processes presented above. Instead, admitting the possibility of lowering the accuracy of the classifier to some degree, we will attempt to accelerate its training.

We suggest the collection of a smaller train set, as well as the execution of the tool from start to finish (including the data collection process) in one go. Instead of collecting a comprehensive corpus over several days for training the classifier, we collect a more limited data set and immediately proceed to the next step of model training. In this manner, we tackle the first two issues: the data collection tool will only run once, decreasing the runtime from up to several days to possibly just a few minutes (depending on the iteration threshold); and a smaller corpus results in a quicker modeling of the LDA and consequently in a faster training of the SVM, decreasing the execution time even further.

Nonetheless, even with a smaller data set, our tool still wastes 15 minutes for every pause imposed by the Twitter Search API. With all possible interactions with Twitter locked, the only step of TORHID that may be anticipated is the training of the classifier with the data collected so far. In order to provide the classifier with the largest train set possible, the tweets need to be all collected first, instead of pruned in place like done before.

For easier understanding, we divide the explanation of F-TORHID in three essential steps: data collection, classifier training and pruning.

The data collection process is similar to the regular method: we employ the Twitter Search API to query Twitter and collect a combination of the most recent and the most popular tweets containing the *seed*; this selection of tweets is stored and inspected, looking for new hashtags to be added to a search queue; and finally the queue is expanded through the repetition of the process, searching for the collected hashtags instead of the *seed*. However, in this implementation we count the calls to the function `Get()` of the Twitter Search API and, right before the limit is reached, the collection of tweets is paused and the classifier trained. When the 15 minutes limitation ends, the process is retaken.

Training the classifier is also analogous to the regular method: a corpus is created with the available tweets and appropriately preprocessed; the latent topics are calculated and their

different combinations are then labeled as *relevant* or *irrelevant* according to the presence or not of the *seed*; finally, the support vector machine is trained with the labeled train set. If all tweets are collected before reaching the limit, a sample of the data is collected for training. On the other hand, since now we are dealing with significantly less tweets (only a few thousands), the training time is usually less than 15 minutes and the classifier can be retrained with additional data if the limit is reached again after resuming data collection.

The pruning process is where the method differs the most: now that we have a trained classifier and a complete data set, we create and preprocess a corpus containing every tweet; the topic distributions are then calculated through the posterior function and used by the classifier to label the tweets as *relevant* or *irrelevant*; finally, the relevant tweets are parsed in order to collect the relevant hashtags. By calculating the co-occurrence of the collected hashtags, we can also create an adjacency matrix and use it to draw the network of relations as with the regular method.

A simplified pseudo-code representation of the method described above to quickly recommend relevant hashtags for a topic is shown in Algorithm 5.

Input: *seed*, *max.searches*, *threshold*, *limit*
$HashtagQueue \leftarrow$ initiate queue with *seed*
$AllTweets \leftarrow \emptyset$
while *HashtagQueue not empty* ***and*** $max.searches > 0$ **do**
  $CurrentHT \leftarrow$ dequeue $HashtagQueue$
  $NewTweets \leftarrow$ set of retrieved tweets containing $CurrentHT$
  $AllTweets \leftarrow AllTweets \cup NewTweets$
  if $counter + 1 = limit$ **then**
    $counter \leftarrow 0$
    $StartTime \leftarrow$ current time
    train the classifier with $AllTweets$
    $TrainTime \leftarrow$ current time $- StartTime$
    $PauseTime \leftarrow 15minutes - TrainTime$
    pause for $PauseTime$
  end
  else
    $counter \leftarrow counter + 1$
  end
  $max.searches \leftarrow max.searches - 1$
end
$RelevantTweets \leftarrow$ tweets from $AllTweets$ classified as relevant
$RelevantHashtags \leftarrow$ hashtags appearing in $RelevantTweets$ more than *threshold* times
return $RelevantHashtags$

Algorithm 5: Collecting Relevant Hashtags using only One Data Set

6.2.1 Comparison with the Regular Method

In this subsection we will be evaluating the performance of the F-TORHID, through a comparison with the regular method presented in previous chapters. While the general concept was not altered much, the classifier of the new tool is supported by a much smaller train set, that is essentially a sample of the full data set that is later classified. It is important to determine the impact this smaller train set will have in the overall accuracy of the labeling process, as well as in the temporal efficiency.

In order to keep our evaluation consistent with the previous tests, we selected the three usual *seeds* for our experiment: *#preterm*, *#cerebralPalsy* and *#gunControl*.

Considering that the answers to search queries on Twitter variate over time, in order to keep our evaluation impartial we decided to employ the same data sets collected in Subsection 5.1.1. While this means that our experiments do not consider the time required to collect the train sets (in the case of the original tool), this is intended since the runtime of the data collection process highly fluctuates with the number of tweets, making our tests unreliable.

Over the course of our experiments, for the original tool, the classifiers were trained with the best performing parameters according to the previous tests: for each *seed*, train sets with 30.000 tweets and 50.000 tweets were used to generate topic models with 25 hidden topics, while the support vector machines were trained with a radial kernel, $\gamma = 1$ and $cost = 100$. The iteration threshold was set at 15, while the number of requested tweets per search was 250 and the hashtag threshold was set at 3. For the F-TORHID tool, the parameters were similar, except for experimenting with the number of tweets collected on each query (250 and 150) and the number of hidden topics (25 and 5).

All tests were performed on a Dell XPS 15 9560 with an Intel®Core^{TM}i7-7700HQ CPU (2.80GHz) and 16GB of memory, running the 64-bit version of Linux Mint 18.1 Serena Cinamon Edition. The R version installed was 3.4.3.

We started by comparing the temporal efficiency of both tools. While the majority of the execution is almost instantaneous, three processes have a noticeable impact in the runtime: collecting the tweets for hashtag extraction, training the classifier and waiting for the access to the Search API to be restored. The programs were executed one at the time for each *seed* and the execution times of these processes were recorded.

Since our searches were simulated in a controlled environment, the Twitter limitations were not properly imposed. For this reason, we counted the number of tweets used on each test and increased their respective runtimes by 15 minutes for every 12000 tweets, which is approximately the number of posts that can be requested with 180 calls to the Search API. Also, because the data sets were already stored in the memory, we reproduced the delay of the connection to Twitter servers by increasing the search times by 1.5 seconds for each 200 tweets. The temporal performances of the different tools with different variables are compared in Figure 6.3.

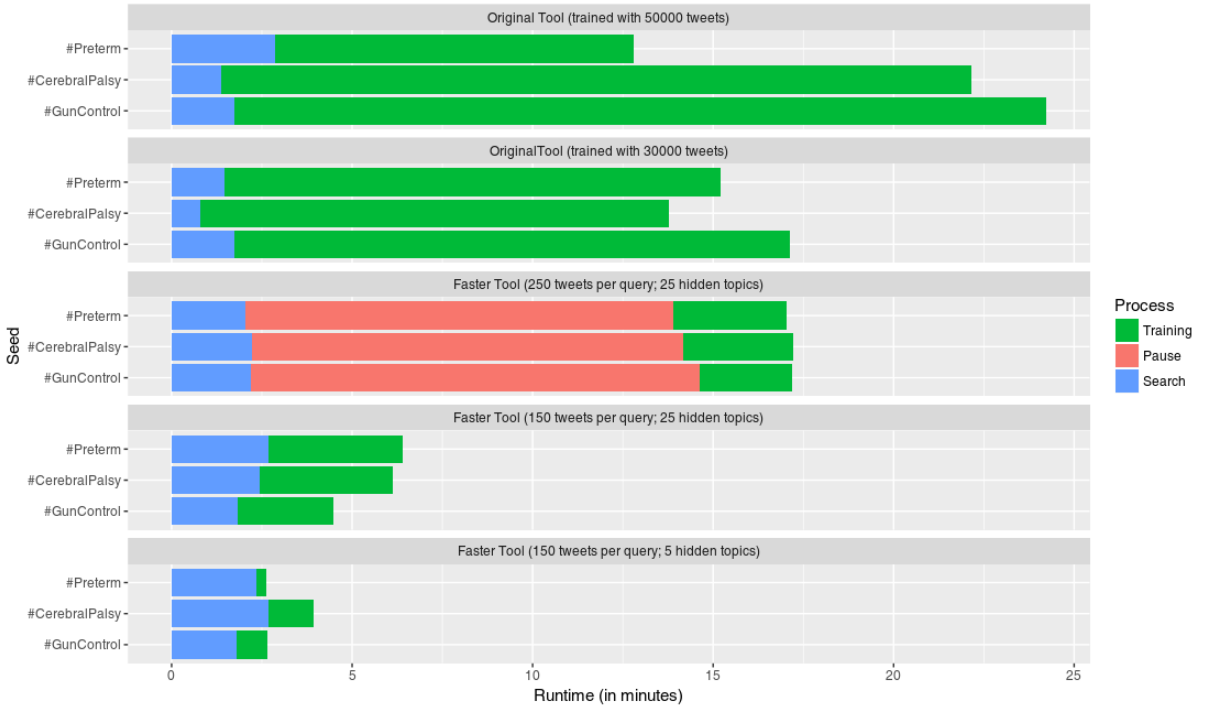


Figure 6.3: Comparison of the Runtimes of the Different Tools

According to our tests, it is apparent that, even discarding the required time for collecting the train set, F-TORHID was capable of performing much more rapidly than the original method. While the search times were slightly smaller when the tweets were pruned in place, the time for training the classifier was enough to undermine its benefits.

The experiments where we specified F-TORHID to collect 250 tweets for each query were the only ones where the limitations on the number of searches was reached. In these cases, even with the time for training the classifier subtracted from the pause time, the impact of the delay was significative, representing over half of the entire execution time. For this reason, these results were comparable to the ones from the original tool trained with 30.000 tweets. On the other hand, in tests where the limit was not reached, F-TORHID was at least twice as fast as the original TORHID.

Apart from the outlier in the first test, the temporal performance was consistent across the three different *seeds*: decreasing the size of the train set usually equals to faster runtimes. Also, our solution with only 5 hidden topics was capable of returning the final set of hashtags in less than 5 minutes, which while is worst than *Hashtagify.me*, is still a justifiable execution time.

We proceeded to evaluate the quality of the results. Since the classifier of the F-TORHID is trained with a data set much smaller than usual, we needed to determine its impact in the returned hashtags, through a comparison with the original implementation. Once again, in order to keep our evaluation impartial we decided to employ the same data sets collected in Subsection 5.1.1 instead of querying Twitter in real time.

We executed our programs again one at the time for each *seed* and for each combination of variables. For the sake of consistency, we made the experiments with the same parameters of the previous tests. The returned hashtags were then stored for manually classification and comparison. Since the relevance of a hashtag to a given topic is subjective, specially without providing context, instructions were given to two persons to independently classify each hashtag of the obtained network as *relevant* or *irrelevant* to the main topic. Searching the meaning of a hashtag was not forbidden or discouraged. The classifications were subsequently compared and when the two persons did not agree, they discussed their interpretations until a consensus was reached. The summaries of the results are described in Figure 6.4.

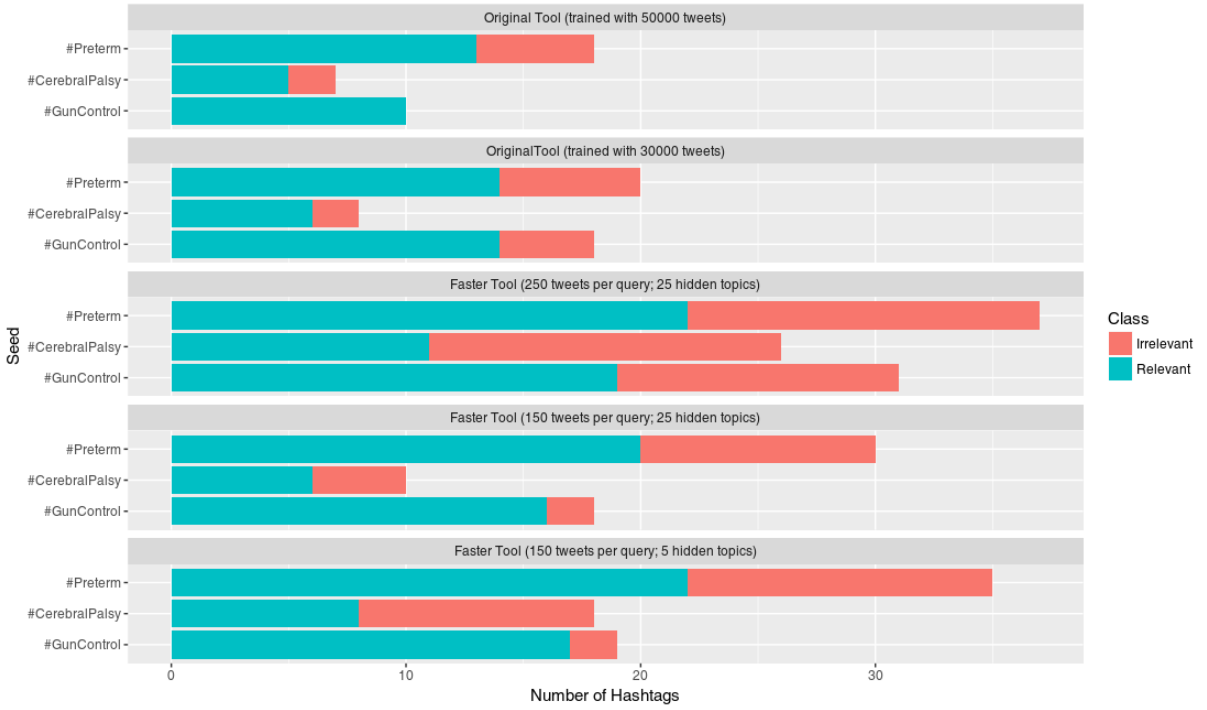


Figure 6.4: Comparison of the Hashtags Returned by the Different Tools

Again, the results are consistent across the three different *seeds*: while the original method returned less hashtags than the new alternative, the percentage of irrelevant ones was usually much lower, to the extent of all the hashtags selected for the *seed #GunControl* in the first test being relevant to the subject. This results indicate that the tolerance of the classifier is probably connected to the size and variety of the train set provided.

While the hashtags returned by the original method were comparable to each other, the results of F-TORHID were more irregular. While returning the largest number of hashtags, the experiment with worst accuracy for every *seed* was the one that collected 250 tweets in each query. This is probably due to our approach of training the classifier when the search mechanism is forced to pause, since at this point the number of tweets in the train set may still be reduced.

In general, the most balanced approach for the F-TORHID was the collection of 250 tweets per query and the training of the classifier with 25 latent topics. This proposal not only provided more

accurate results with less irrelevant hashtags, but was also a close second in temporal performance, being capable of recommending hashtags relevant for a topic in Twitter in approximately 10 minutes, without requiring previous data.

We consider the results of F-TORHID satisfactory, despite not providing as accurate hashtags as the original method, considering that was not the purpose of its implementation. Since the method we proposed requires data specific to each *seed* explored, in order to train the classifier, and the collection of that data can take up to several days depending on factors such as the popularity of the *seed*, the goal of this faster approach was to provide quicker but acceptable results in cases where a faster execution time would be prioritized over accuracy. In this manner, we proved that TORHID was capable of recommending hashtags relevant to a topic, at the moment of the search, with acceptable execution times; even if at the expense of precision to some extent.

Chapter 7

Case Study

As previously stated, the main purpose of this thesis is the development of methods and tools to identify and collect hashtags relevant to a topic in twitter, in order to promote communication and research. In this chapter, we will focus on describing an example of a common task in data science and how it could benefit from the application of TORHID to improve its results.

Sentiment analysis (or opinion mining) is a field of text mining whose goal is to employ natural language processes to retrieve and study information about the emotional state behind a sequence of words. This techniques are frequently used in commercial applications to examine people's opinions, evaluations and emotions in relation to services, events, products and organizations [86]. One of the advantages of sentiment analysis is the quick and automatic overview of the opinion of the public over a certain topic. For this reason, these methods have already been applied to Twitter data in multiple instances and other works cover the subject in more detail [1][56][5][59][11].

When analyzing trends in Twitter, it is common to follow and study only a specific hashtag or person of interest. We believe that this is a good example where the application of TORHID to explore different hashtags could be advantageous.

Since this chapter is merely demonstrative and sentiment analysis is not in the scope of our work, we will be focusing on simplicity. We will be exploring one of the most basic tasks of this field: the calculation of polarity, which essentially consists in the classification of a given piece of writing as positive, negative or neutral, taking into account the words composing it.

Furthermore, instead of simply choosing a random *seed* for experimentation as usual, we wanted to explore a different route. Since the calculation of polarity basically evaluates a document as positive or negative and gives it a score, we decided to compare the performance of two different *seeds* in social media over a period of time. More specifically, we will be comparing the social media performance of two competitor companies during an annual event.

7.1 Scenario

The Electronic Entertainment Expo, commonly referred to as E3, is one of the major annual events of the video game industry [4], with 15,000 tickets sold to the public in 2018 [41]. The event consists essentially in an exhibition floor for companies and game publishers to demonstrate and advertise the games and related products to be released in the upcoming year.

During the event, the most prominent companies of the industry organize press conferences to introduce their new products and offerings to the public. This press conferences are broadcasted live over the internet with an associated hashtag so that people feel encouraged to engage with them on social media platforms like Twitter. On this case study, we will be comparing the performance of two major competitors in the business of gaming console manufacturing, over the period of the E3 2018: the Xbox and the Playstation.

7.2 Data Collection

Since the objective of the E3 is to bring exposure to the attending publishers, the keynotes are scheduled to not occur at the same time. In 2018, the Xbox keynote toke place on a Sunday (June 10) and the Playstation press conference toke place on the following day (June 11). For this reason, in an attempt to achieve fairness, we decided to collect and compare two data sets for each brand: one immediately at the end of the respective keynote and another at the end of the event on Thursday (June 14).

Each data set was collected in the following manner: we selected the official hashtags for each presentation (*#XboxE3* for Xbox and *#PlaystationE3* for the Playstation) as the *seeds*; since we were analyzing a live event and speed was to be prioritized, we employed our faster tool to retrieve relevant hashtags for each one of the *seeds*; finally, we queried Twitter and collected a combination of the 200 most recent and most popular tweets containing those hashtags, storing them for the experimentation process. The topic models were generated with 25 hidden topics, while the support vector machines were trained with a radial kernel, $\gamma = 1$ and $cost = 100$; the iteration threshold was set at 15, while the number of requested tweets per search was 200 and the hashtag threshold was set to 3. Figure7.1 depicts the network of relations of the hashtags collected for the *seed #XboxE3* and Figure7.2 depicts the equivalent network for the *seed #PlaystationE3*.

We proceeded to convert the sets of tweets into corpora and some preprocessing steps were performed to prepare the data: all non-ASCII characters were removed; urls, mentions and other metadata were removed; numbers and punctuation were removed; letters were converted to lower case; stopwords were removed; and word stemming was applied.

In addition, we inspected the networks of relations and removed some of the returned hashtags. For example, the hashtag *#XboxE3* was returned for both networks of the *seed #PlaystationE3*

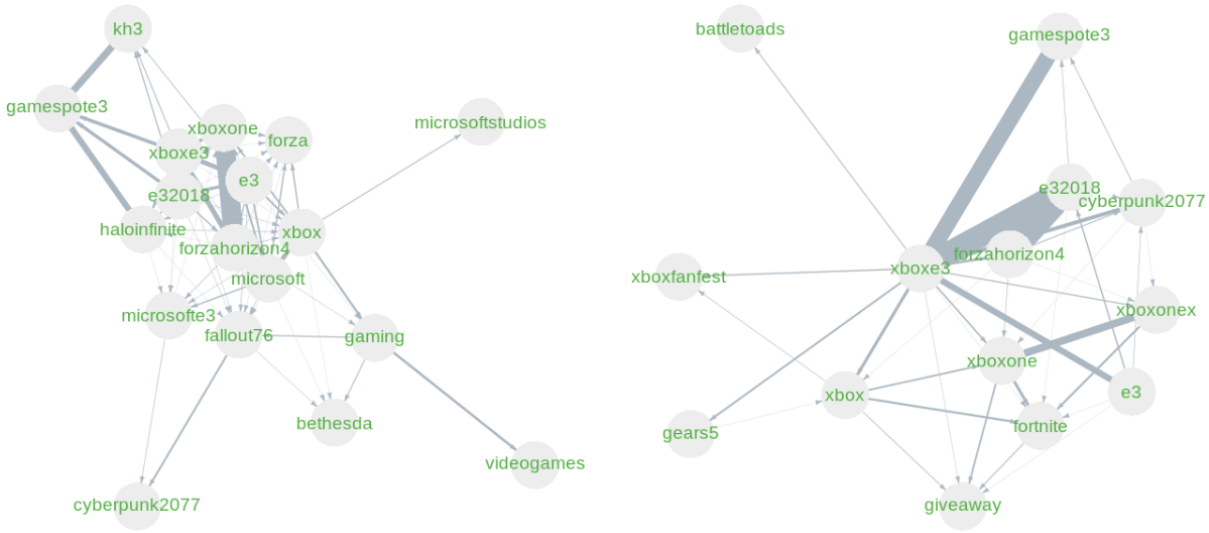


Figure 7.1: Network of Relations of the Hashtags collected for the *seed #XboxE3* on June 10 (left) and June 14 (right)

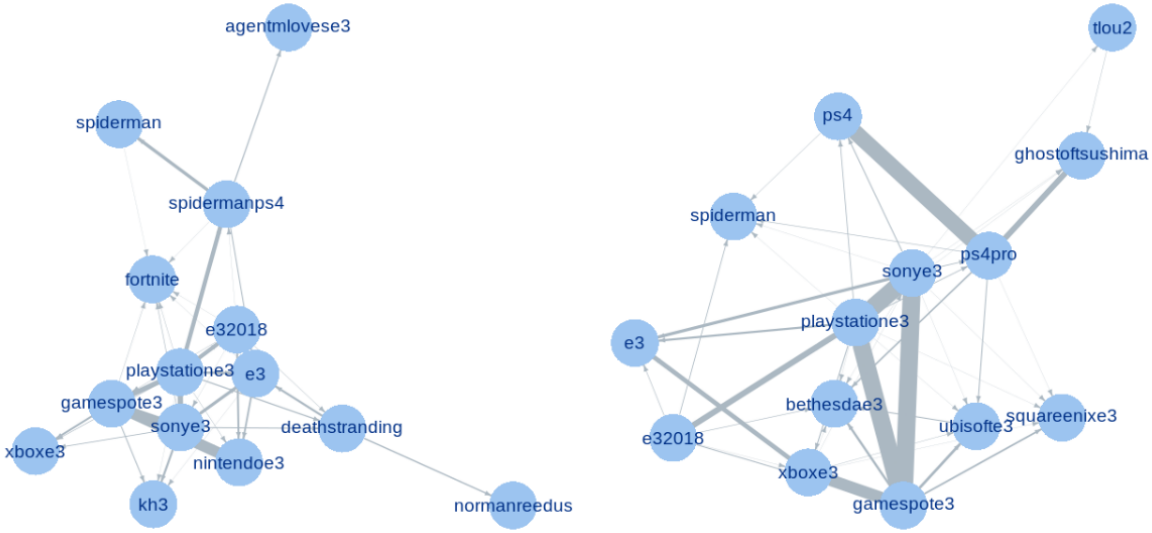


Figure 7.2: Network of Relations of the Hashtags collected for the *seed #PlaystationE3* on June 11 (left) and June 14 (right)

and, since both hashtags were used to promote keynotes on the same event, they are undoubtedly related. However, the tweets containing this hashtag, as well as hashtags referring to other press conferences, would contaminate the results of opinion mining for this *seed* and, as a consequence, must be removed for more accurate results.

Word clouds based on the remaining tweets of the data sets were generated and are depicted in Figure 7.3 for the *seed #XboxE3*, while Figure 7.4 illustrates the equivalent word cloud for the *seed #PlaystationE3*.

We implemented the process for visualizing the represented data in the `R` programming

Figure 7.3: Word cloud based on tweets collected for the *seed #XboxE3* on June 10 (left) and June 14 (right)

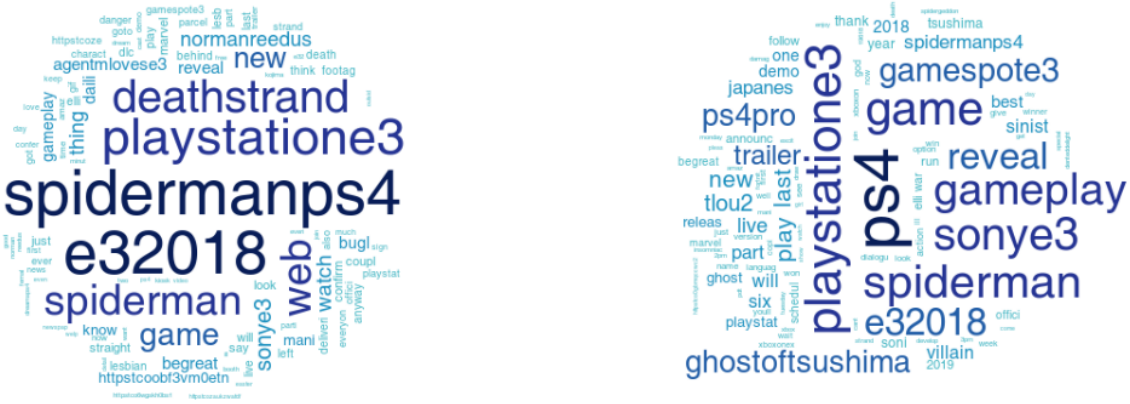


Figure 7.4: Word cloud based on tweets collected for the *seed #XboxE3* on June 10 (left) and June 14 (right)

language, employing the package `wordcloud` [19], which provides straightforward functions for creating word clouds based on a corpus or a vector of words, with options for controlling the scales and colors of the text.

Through a quick analysis of the word clouds, it is noticeable that the most preeminent words are directly related to the event (*#E32018*), to the gaming platform promoted by the *seed* (*#ps4* and *#xbox*) and to the games presented for those platforms (*#forza* and *#spiderman*). Furthermore, in both cases, the word cloud for the end of the event is different from the one of the press event, indicating that we were successful at capturing hashtags temporally relevant to the *seeds*.

7.3 Sentiment Comparison

Instead of immediately dealing with the comparison of sentiments behind the texts from both *seeds* and drawing conclusions, we started by investigating the temporal differences of the tweets, that is, if our decision to perform data collection at different points in time was beneficial and resulted in the collection of more useful data. For this task, we performed sentiment analysis of both data sets collected for each *seed* and compared the sentiments during the keynotes with the sentiments in the last day. For the sake of consistency, for the remaining of this chapter, we will refer to the data collected from the press conferences as *live* (as in *live tweets*) and to the data collected from the final day of the conference as *last day*.

As previously stated, the method employed for the analysis was polarity calculation, which works by finding the words in the tweets that represent positive sentiments and attribute them 1 point and find the words in the tweets that represent negative sentiments and attribute them -1 point. The total polarity score of the tweet is the result of adding together the scores of all the individual words that compose it.

For the task of classifying individual words, a dictionary is required. We used the function `get_sentiments` from `tydytext` [73] to retrieve the *bing* sentiment lexicon, since this dictionary was the one with the simplest implementation (only mapped words as positive or negative), whilst being the most complete (had the most entries). Other avaliable lexicons contain more classes of words like *anger*, *fear* and *trust*, but we chose the *bing* dictionary for the sake of simplicity.

The next step was the inspection of the sentiment lexicon and the removal of some words. This was a required step since the polarity score is not always very accurate. This method usually struggles with taking into account the overall context of the text, since it focuses only on individual words, and is incapable of understanding sarcasm, for example. Most of the words removed were simply names of video games that have positive or negative connotations, like *Devil*, from the video game series *Devil may Cry*, and *Fallout*[1], from the series with the same name.

The process of performing sentiment analysis was the following for each data set: first, a document term matrix was generated based on the corpus with the tweets; next the classifications of the words from the lexicon were mapped with the terms from the DTM and its values were summed to calculate the polarity score of every tweet; we summed all the positive polarity scores and all the negative polarity scores of the tweets independently, thus creating a summary of their positive/negative balance. In order to make the inspection of the results easier, the neutral values were discarded. Figure 7.5 depicts the comparison of the live summaries and the last day summaries for each *seed*.

Through the analysis of Figure 7.5, not only is visible that exists a difference between the

[1] Fallout: the radioactive debris that fall after a nuclear explosion.

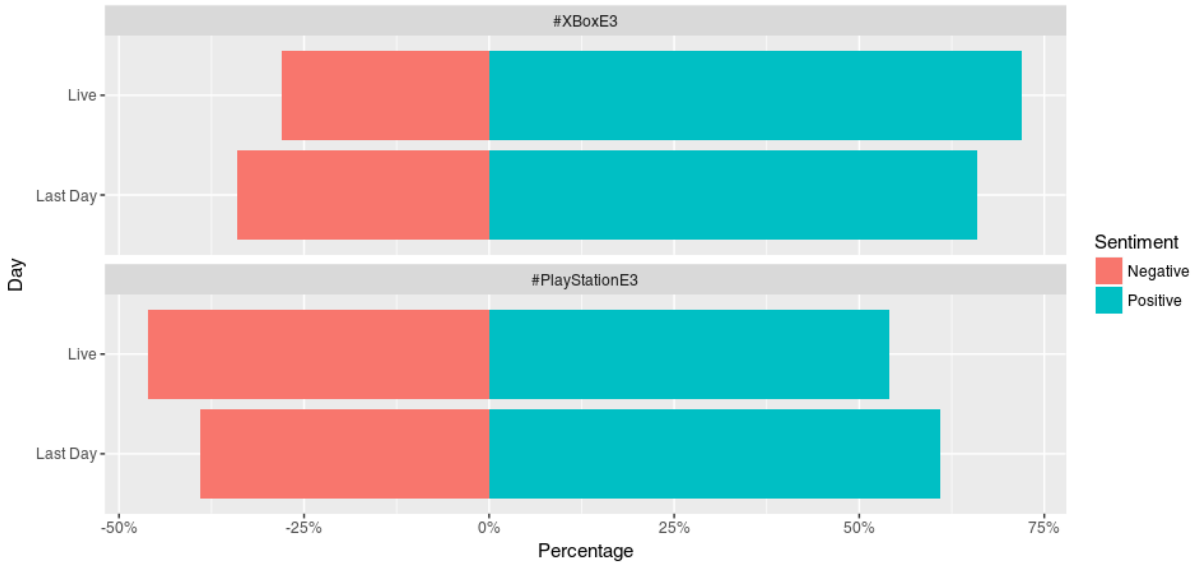


Figure 7.5: Comparison of the Live and the Last Day Summaries of the Sentiments Regarding *#XboxE3* and *#PlaystationE3*

live sentiments and the last day sentiments, but also that they stabilized over time. Right after the live presentations, the sentiments related to the xbox press conference were better than the ones related to the playstation keynote. However, it seems that after a few days, the approval of the xbox and the contempt to the playstation presentations started to decrease, maybe due to new information later presented from the exhibition floor.

We proceeded to perform the final comparison: the live data sets were mixed with the respective last day data sets and the process of calculating the polarity scores and the positive/negative balance was repeated. Figure 7.6 depicts the comparison of the live and the last day summaries for each *seed*.

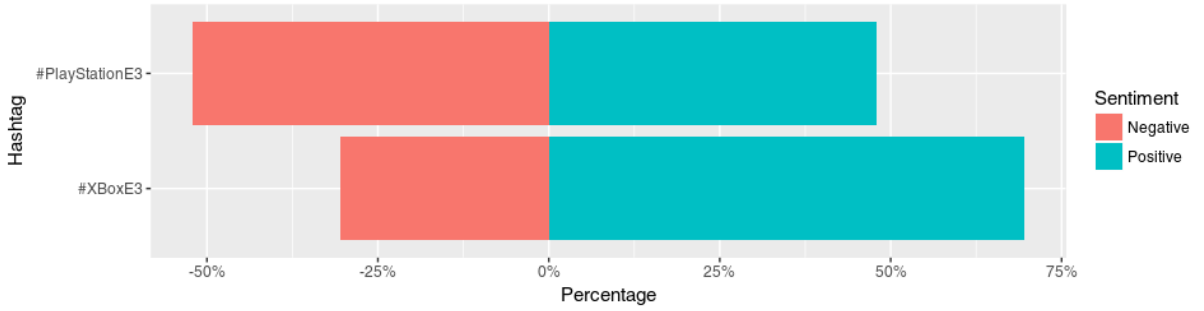


Figure 7.6: Comparison of the Sentiments from the Tweets Collected for *#XboxE3* and *#PlaystationE3*

It is immediately noticeable that the sentiments of the tweets with hashtags relevant to *#XboxE3* are more positive than the ones for *#PlaystationE3*, even considering the period of reflection after the presentations. This indicates that the audience was probably overall more pleased with the announcements from Microsoft than with the ones from Sony regarding their

new lineups of games and plans for the future. The implication of a better keynote from xbox was later acknowledged by some media outlets dedicated to the gaming industry [68][72], suggesting that our results were indeed accurate.

7.4 Impact of the Tool

After determining that our sentiment analysis was accurate, we needed to understand the impact that our hashtag recommendation tool had in the final results, that is, if the additional information gathered by the new tweets and hashtags was helpful or not. We started by comparing the sentiments of the tweets with the *seeds* with the hashtags with the best and the worst polarity scores from the respective data sets. Later, we also compared the sentiments of the tweets with the *seed* alone with the entire network collected by TORHID.

We thus proceeded in the following manner for each data set: the polarity score of every tweet was calculated as before; next, we inspected the document term matrix and divided the tweets from the data set into groups according to the hashtags contained in them; through the sum of the polarity scores of these groups of tweets, we calculated the summary of the sentiments for each hashtag; the hashtag with the best and the worst summaries were selected for comparison. Figure 7.7 depicts the comparison of the best and worst summaries of each data set for each *seed*.

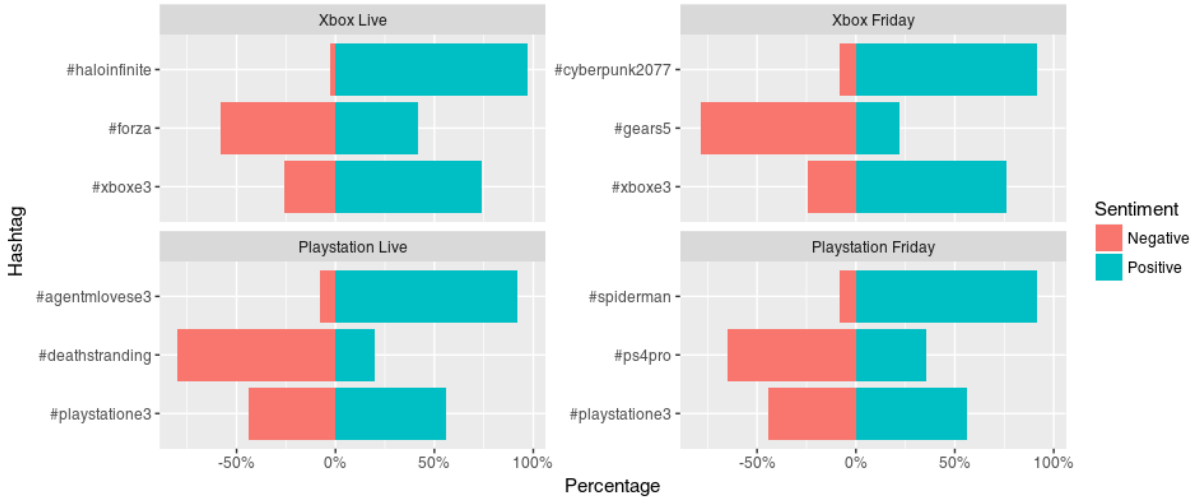


Figure 7.7: Comparison of the Best and Worst Summaries from the Tweets Collected for *#XboxE3* and *#PlaystationE3* with the Respective *Seeds*

It is noticeable that the hashtags with best associated sentiments are related to the best received video games showcased during the event [20][38], while the hashtag with worst sentiments for playstation on the last day was related to hardware.

Overall, the sentiments of the tweets with *#XboxE3* have some resemblance with the sentiments of posts from the corresponding best hashtag, while the sentiments of *#PlaystationE3* are more balanced, always approximating a 50/50 ratio. Also, while the hashtags with the worst

sentiments had achieved better results by the end of the event or disappeared from the network, the emotions related to the *seeds* alone were almost identical from start to finish, showing no evolution unlike what happened with the entire networks.

Finally, we evaluated the difference between the sentiments of the tweets with only the *seed* and the sentiments of the tweets from the full data sets. In this particular case, instead of comparing the proportion of positive/negative sentiments, we compared the actual percentage of positive and negative tweets, thus indirectly including the neutral sentiments. Figure 7.8 depicts the comparison of the sentiments towards the *seeds* with the sentiments of the entire data sets for *#XboxE3* and *#PlaystationE3*.

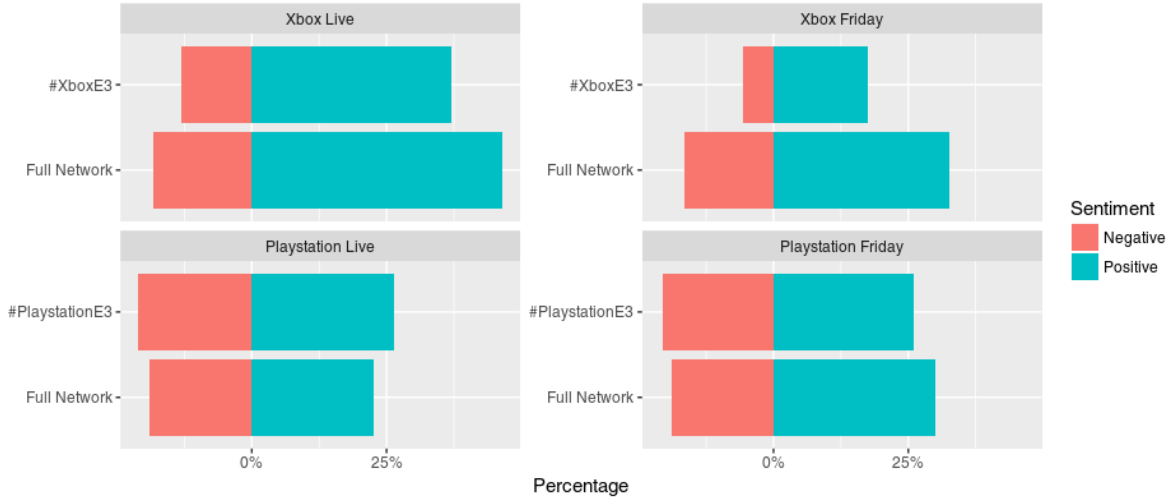


Figure 7.8: Comparison of the Sentiments of *seeds* with the entire Networks Collected for *#XboxE3* and *#PlaystationE3*

When inspecting the tweets related to the xbox, the collection of posts from relevant hashtags successfully increased the proportion of tweets with positive and negative sentiments. On the other hand, the percentage of positive and negative tweets for *#PlaystationE3* alone does not change much when compared to the entire data set collected. Either way, the collection of the data sets always resulted in the availability of more positive and negative tweets, since the overall number of tweets increased significantly.

Overall, we consider that the case study described was successful since we achieved results somewhat corroborated by media outlets. Also, we were capable of considering more data than what would normally be available due to the restrictions imposed by Twitter, since we not only explored tweets with the *seeds* but also other posts considered relevant to them. While we do not aim to establish this process as essential for sentiment analysis tasks on Twitter, we successfully proved that TORHID is capable of easily being applied to it, as well as other use cases where more information would be even more useful.

Chapter 8

Conclusion

In this paper, we propose TORHID (Topic Relevant Hashtag Identification), method based on LDA and SVM for identifying topic relevant hashtags in Twitter streams. Unlike most proposed methods, our approach relies on a small database only for training the classifier and searches for new tweets, every time. This way, the hashtag network is always as updated as possible. However, it was very rare for the collected hashtags to have more that one degree of distance to the seed hashtag.

We started by searching a small group of tweets from a seed hashtag that represents the topic and stored them in a database. This tweets were later used to model an LDA and train a SVM to classify new tweets as relevant or irrelevant to the same given topic. Every time we collect a new tweet, we can keep it and harvest its hashtags or discard it based on the classifier. In this manner, we are capable of creating a network of hashtags relevant to the topic.

According to our tests where data was manually labeled for comparison, our classifier has shown promising results in detecting and removing the unrelated hashtags from the networks of hashtags, surpassing the results of only considering if the *seed* hashtag is avalible or not. One important aspect of TORHID was the very high precision rates, close to 100%, meaning that once the classifier categorizes a tweet as relevant, that classification can be trusted almost every time.

In addition, we compared TORHID with *Hashtagify*, a web service whose goals are similar to ours, with satisfactory results. Our method not only collected more relevant hashtags on average, but also provided more information about the relations of those hashtags with each other and the seed. We were also more successful in collecting hashtags that are relevant depending on the context or time, that is, our relations were more updated. However, the type of evaluation we performed is susceptible of different interpretations.

The amount of hashtags returned by TORHID was usually between five and ten, which is acceptable for most use cases. There may be some alternatives to possibly perform a controlled expansion of the network like retraining the classifier with the new hashtags in the network and redo the search, combining the results of TORHID with other tools like *Hashtagify* or employing

community detection algorithms in larger networks, attempting to achieve faster results.

The major drawback with the method we propose is the very long time required for gathering a reasonable amount of tweets to train the classifier and the training itself which takes significantly longer than asking a query in *Hashtagify*. This compromise is a consequence of keeping the results as updated and tailored as possible.

We proceeded to improve the usability of the tool through the addition of a simple user interface and experimented with ways of accelerating its results, while reducing the need of a previous step of data collection for training. This experiments were successful since our new faster tool was capable of retrieving relevant hashtags for a *seed*, executing the entire process in less than 5 minutes.

Finally, we described an example use case where TORHID was used to gather additional data on a subject in order to perform sentiment analysis and compare the social media performance of two competitor companies during an event that both attended and where they introduced new products.

Nevertheless, in our opinion, there are still some improvements that could be applied to our tool. For example, while we defined the possible relations among the collected hashtags (as unrelated, related to the same topic, representing a super-topic or representing a sub-topic), we never leveraged them in our solution. Furthermore, other concepts like word-embeddings could be used in an attempt to enhance the performance of the classifier.

Overall, our proposed method to search, expand and prune a network of hashtags related to a topic in Twitter streams provided acceptable results, specially when considering the fact that it is completely unsupervised.

Bibliography

[1] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. Sentiment analysis of twitter data. In *Proceedings of the Workshop on Languages in Social Media*, LSM '11, pages 30–38, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN: 978-1-932432-96-1.

[2] Hamidreza Alvari. Twitter hashtag recommendation using matrix factorization. *CoRR*, abs/1705.10453, 2017.

[3] Dolan Antenucci, Gregory Handy, Akshay Naresh Modi, and Miller Tinkerhess. Classification of tweets via clustering of hashtags. In *EECS 545, Final Project*, 2011.

[4] Entertainment Software Association. About e3, May 2018.

[5] Albert Bifet and Eibe Frank. Sentiment knowledge discovery in twitter streaming data. In Bernhard Pfahringer, Geoffrey Holmes, and Achim Hoffmann, editors, *Discovery Science - 13th International Conference, DS 2010, Canberra, Australia, October 6-8, 2010. Proceedings*, volume 6332 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2010. ISBN: 978-3-642-16183-4. doi:10.1007/978-3-642-16184-1_1.

[6] David Blei and John Lafferty. Correlated topic models. In *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada]*, pages 147–154, 2005.

[7] David Blei and John Lafferty. Dynamic topic models. In William Cohen and Andrew Moore, editors, *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, volume 148 of *ACM International Conference Proceeding Series*, pages 113–120. ACM, 2006. ISBN: 1-59593-383-2. doi:10.1145/1143844.1143859.

[8] David Blei, Andrew Ng, and Michael Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[9] Dustin Boswell. Introduction to support vector machines. *Lectures in University of California*, 2002.

[10] Christopher Burges. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2(2):121–167, June 1998. ISSN: 1384-5810. doi:10.1023/A:1009715923555.

[11] Josemar Alves Caetano, Helder Seixas Lima, Mateus Santos, and Humberto Torres Marques-Neto. Using sentiment analysis to define twitter political users' classes and their homophily during the 2016 american presidential election. *J. Internet Services and Applications*, 9(1): 18:1–18:15, 2018. doi:10.1186/s13174-018-0089-0.

[12] Corinna Cortes and Vladimir Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

[13] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006.

[14] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. In Eric Brewer and Peter Chen, editors, *6th Symposium on Operating System Design and Implementation (OSDI 2004), San Francisco, California, USA, December 6-8, 2004*, pages 137–150. USENIX Association, 2004.

[15] James Dickey. Multiple hypergeometric functions: Probabilistic interpretations and statistical uses. *Journal of The American Statistical Association - J AMER STATIST ASSN*, 78: 628–637, 09 1983.

[16] Roman Dovgopol and Matt Nohelty. Twitter hash tag recommendation. *CoRR*, abs/1502.00094, 2015.

[17] Miles Efron. Hashtag retrieval in a microblogging environment. In Fabio Crestani, Stéphane Marchand-Maillet, Hsin-Hsi Chen, Efthimis Efthimiadis, and Jacques Savoy, editors, *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2010, Geneva, Switzerland, July 19-23, 2010*, pages 787–788. ACM, 2010. ISBN: 978-1-4503-0153-4. doi:10.1145/1835449.1835616.

[18] Ingo Feinerer, Kurt Hornik, and David Meyer. Text mining infrastructure in r. *Journal of Statistical Software*, 25(5):1–54, March 2008.

[19] Ian Fellows. *Package 'wordcloud'*, 08 2018. R package version 1.1.9.

[20] Stefanie Fogel. Xbox e3 2018: The 10 biggest announcements, June 2018.

[21] Sandra Garcia Esparaza, Michael O'Mahony, and Barry Smyth. Towards tagging and categorization for micro-blogs. *Proceeding of the 21st National Conference on Artificial Intelligence and Cognitive Science*, 01 2010.

[22] Jeff Gentry. *twitteR: R Based Twitter Client*, 07 2015. R package version 1.1.9.

[23] Mark Girolami and Ata Kabán. On an equivalence between PLSI and LDA. In Charles Clarke, Gordon Cormack, Jamie Callan, David Hawking, and Alan Smeaton, editors, *SIGIR 2003: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 28 - August 1, 2003, Toronto, Canada*, pages 433–434. ACM, 2003. ISBN: 1-58113-646-3. doi:10.1145/860435.860537.

[24] Statista GmbH. Most famous social network sites 2017, by active users, September 2017.

[25] Statista GmbH. Twitter: number of monthly active users 2010-2017, September 2017.

[26] Fréderic Godin, Viktor Slavkovikj, Wesley De Neve, Benjamin Schrauwen, and Rik Van de Walle. Using topic models for twitter hashtag recommendation. In *WWW (Companion Volume)*, pages 593–596. International World Wide Web Conferences Steering Committee / ACM, 2013.

[27] Bettina Grün and Kurt Hornik. topicmodels: An R package for fitting topic models. *Journal of Statistical Software*, 40(13):1–30, 2011. doi:10.18637/jss.v040.i13.

[28] Morgan Harvey and Fabio Crestani. Long time, no tweets! time-aware personalised hashtag suggestion. In Allan Hanbury, Gabriella Kazai, Andreas Rauber, and Norbert Fuhr, editors, *Advances in Information Retrieval - 37th European Conference on IR Research, ECIR 2015, Vienna, Austria, March 29 - April 2, 2015. Proceedings*, volume 9022 of *Lecture Notes in Computer Science*, pages 581–592, 2015. ISBN: 978-3-319-16353-6. doi:10.1007/978-3-319-16354-3_65.

[29] Liangjie Hong and Brian Davison. Empirical study of topic modeling in twitter. In Lee Giles, Prasenjit Mitra, Igor Perisic, John Yen, and Haizheng Zhang, editors, *Proceedings of the 3rd Workshop on Social Network Mining and Analysis, SNAKDD 2009, Paris, France, June 28, 2009*, pages 80–88. ACM, 2010. ISBN: 978-1-60558-676-2. doi:10.1145/1964858.1964870.

[30] Lichan Hong, Gregorio Convertino, and Ed Chi. Language matters in twitter: A large scale study. In Lada Adamic, Ricardo Baeza-Yates, and Scott Counts, editors, *Proceedings of the Fifth International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July 17-21, 2011*. The AAAI Press, 2011.

[31] Twitter inc. Twitter standar search api, June 2018.

[32] Twitter inc. Rate limiting, June 2018.

[33] Tomoharu Iwata, Kazumi Saito, Naonori Ueda, Sean Stromsten, Thomas Griffiths, and Joshua Tenenbaum. Parametric embedding for class visualization. *Neural Computation*, 19 (9):2536–2556, 2007. doi:10.1162/neco.2007.19.9.2536.

[34] Michael Jordan. Learning in graphical models. *STATISTICAL SCIENCE*, 19(1):140–155, 2004.

[35] Elham Khabiri, James Caverlee, and Krishna Yeswanth Kamath. Predicting semantic annotations on the real-time web. In Ethan Munson and Markus Strohmaier, editors, *23rd ACM Conference on Hypertext and Social Media, HT '12, Milwaukee, WI, USA, June 25-28, 2012*, pages 219–228. ACM, 2012. ISBN: 978-1-4503-1335-3. doi:10.1145/2309996.2310034.

[36] Kullback and Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 03 1951. doi:10.1214/aoms/1177729694.

[37] Su Mon Kywe, Tuan-Anh Hoang, Ee-Peng Lim, and Feida Zhu. On recommending hashtags in twitter networks. In Karl Aberer, Andreas Flache, Wander Jager, Ling Liu, Jie Tang, and Christophe Guéret, editors, *Social Informatics - 4th International Conference, SocInfo 2012, Lausanne, Switzerland, December 5-7, 2012. Proceedings*, volume 7710 of *Lecture Notes in Computer Science*, pages 337–350. Springer, 2012. ISBN: 978-3-642-35385-7. doi:10.1007/978-3-642-35386-4_25.

[38] Liz Lanier. Sony e3 2018: The six biggest announcements, June 2018.

[39] Oliver Laughland, Richard Luscombe, and Alan Yuhas. Florida school shooting: at least 17 people dead on 'horrific, horrific day'. *The Guardian*, 2018.

[40] Janette Lehmann, Bruno Gonçalves, José Ramasco, and Ciro Cattuto. Dynamical classes of collective attention in twitter. In Alain Mille, Fabien Gandon, Jacques Misselis, Michael Rabinovich, and Steffen Staab, editors, *Proceedings of the 21st World Wide Web Conference 2012, WWW 2012, Lyon, France, April 16-20, 2012*, pages 251–260. ACM, 2012. ISBN: 978-1-4503-1229-5. doi:10.1145/2187836.2187871.

[41] Marshall Lemon. E3 2018 attendance was highest since 2005, June 2018.

[42] Quanzhi Li, Sameena Shah, Rui Fang, Armineh Nourbakhsh, and Xiaomo Liu. Discovering relevant hashtags for health concepts: A case study of twitter. In *AAAI Workshop: WWW and Population Health Intelligence*, volume WS-16-15 of *AAAI Workshops*. AAAI Press, 2016.

[43] Tianxi li, Yu Wu, and Yu Zhang. Twitter hash tag prediction algorithm. *ICOMP'11 - The 2011 International Conference on Internet Computing*, 2011.

[44] Zhenhui Li, Ding Zhou, Yun-Fang Juan, and Jiawei Han. Keyword extraction for social snippets. In Michael Rappa, Paul Jones, Juliana Freire, and Soumen Chakrabarti, editors, *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 1143–1144. ACM, 2010. ISBN: 978-1-60558-799-8. doi:10.1145/1772690.1772845.

[45] Jimmy Lin and Gilad Mishne. A study of "churn" in tweets and real-time search queries. In John Breslin, Nicole Ellison, James Shanahan, and Zeynep Tufekci, editors, *Proceedings of the Sixth International Conference on Weblogs and Social Media, Dublin, Ireland, June 4-7, 2012*. The AAAI Press, 2012.

[46] Oded Maimon and Lior Rokach. *Data Mining and Knowledge Discovery Handbook*. Springer-Verlag, Berlin, Heidelberg, 2005. ISBN: 0387244352, 9780387244358.

[47] Allie Mazzia and James Juett. Suggesting hashtags on twitter. *EECS 545 Project*, 01 2010.

[48] Rishabh Mehrotra, Scott Sanner, Wray Buntine, and Lexing Xie. Improving LDA topic models for microblogs via tweet pooling and automatic labeling. In Gareth Jones, Paraic Sheridan, Diane Kelly, Maarten de Rijke, and Tetsuya Sakai, editors, *The 36th International*

ACM SIGIR conference on research and development in Information Retrieval, SIGIR '13, Dublin, Ireland - July 28 - August 01, 2013, pages 889–892. ACM, 2013. ISBN: 978-1-4503-2034-4. doi:10.1145/2484028.2484166.

[49] David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, and Friedrich Leisch. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 07 2018. R package version 1.6-8.

[50] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In Christopher Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119, 2013.

[51] George Miller. WORDNET: a lexical database for english. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*. Morgan Kaufmann, 1992. ISBN: 978-1-55860-272-4.

[52] Mor Naaman, Hila Becker, and Luis Gravano. Hip and trendy: Characterizing emerging trends on twitter. *JASIST*, 62(5):902–918, 2011. doi:10.1002/asi.21489.

[53] Nasir Naveed, Thomas Gottron, Jérôme Kunegis, and Arifah Che Alhadi. Searching microblogs: coping with sparsity and document quality. In Craig Macdonald, Iadh Ounis, and Ian Ruthven, editors, *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011*, pages 183–188. ACM, 2011. ISBN: 978-1-4503-0717-8. doi:10.1145/2063576.2063607.

[54] Eriko Otsuka, Scott Wallace, and David Chiu. Design and evaluation of a twitter hashtag recommendation system. In Bipin Desai, Ana Maria de Almeida, Jorge Bernardino, and Elsa Ferreira Gomes, editors, *18th International Database Engineering & Applications Symposium, IDEAS 2014, Porto, Portugal, July 7-9, 2014*, pages 330–333. ACM, 2014. ISBN: 978-1-4503-2627-8. doi:10.1145/2628194.2628238.

[55] Eriko Otsuka, Scott Wallace, and David Chiu. A hashtag recommendation system for twitter data streams. *Computational Social Networks*, 3(1):3, May 2016. ISSN: 2197-4314. doi:10.1186/s40649-016-0028-9.

[56] Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, 17-23 May 2010, Valletta, Malta*. European Language Resources Association, 2010. ISBN: 2-9517408-6-7.

[57] Xuan Hieu Phan, Minh Le Nguyen, and Susumu Horiguchi. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In Jinpeng

Huai, Robin Chen, Hsiao-Wuen Hon, Yunhao Liu, Wei-Ying Ma, Andrew Tomkins, and Xiaodong Zhang, editors, *Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, April 21-25, 2008*, pages 91–100. ACM, 2008. ISBN: 978-1-60558-085-2. doi:10.1145/1367497.1367510.

[58] Jan Pöschko. Exploring twitter hashtags. *CoRR*, abs/1111.6553, 2011.

[59] Evangelos Psomakelis, Konstantinos Tserpes, Dimosthenis Anagnostopoulos, and Theodora Varvarigou. Comparing methods for twitter sentiment analysis. *CoRR*, abs/1505.02973, 2015.

[60] Daniel Ramage, David Leo Wright Hall, Ramesh Nallapati, and Christopher Manning. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP 2009, 6-7 August 2009, Singapore, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 248–256. ACL, 2009. ISBN: 978-1-932432-59-6.

[61] Daniel Ramage, Susan Dumais, and Daniel Liebling. Characterizing microblogs with topic models. In William Cohen and Samuel Gosling, editors, *Proceedings of the Fourth International Conference on Weblogs and Social Media, ICWSM 2010, Washington, DC, USA, May 23-26, 2010*. The AAAI Press, 2010.

[62] RStudio, Inc. *Easy web applications in R.*, 11 2018.

[63] RStudio Team. *RStudio: Integrated Development Environment for R*. RStudio, Inc., Boston, MA, 07 2018.

[64] Bernhard Scholkopf and Alexander Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001. ISBN: 0262194759.

[65] Bernhard Schölkopf, Christopher Burges, and Vladimir Vapnik. Extracting support data for a given task. In *First International Conference on Knowledge Discovery & Data Mining (KDD-95)*, pages 252–257, Menlo Park, CA, USA, August 1995. Max-Planck-Gesellschaft, AAAI Press.

[66] Bernhard Schölkopf, Chris Burges, and Vladimir Vapnik. Extracting support data for a given task. In *Proceedings, First International Conference on Knowledge Discovery & Data Mining, Menlo Park*, pages 252–257. AAAI Press, 1995.

[67] Linchpin SEO. Twitter cheat sheet to increase engagement and followers, June 2016.

[68] Shak. E3 2018: Microsoft won e3, sony got it very wrong, June 2018.

[69] Jieying She and Lei Chen. TOMOHA: topic model-based hashtag recommendation on twitter. In Chin-Wan Chung, Andrei Broder, Kyuseok Shim, and Torsten Suel, editors, *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April*

7-11, 2014, Companion Volume, pages 371–372. ACM, 2014. ISBN: 978-1-4503-2745-9. doi:10.1145/2567948.2577292.

[70] Bichen Shi, Georgiana Ifrim, and Neil Hurley. Learning-to-rank for real-time high-precision hashtag recommendation for streaming news. In Jacqueline Bourdeau, Jim Hendler, Roger Nkambou, Ian Horrocks, and Ben Zhao, editors, *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, pages 1191–1202. ACM, 2016. ISBN: 978-1-4503-4143-1. doi:10.1145/2872427.2882982.

[71] Bichen Shi, Gevorg Poghosyan, Georgiana Ifrim, and Neil Hurley. Hashtagger+: Efficient high-coverage social tagging of streaming news. *IEEE Trans. Knowl. Data Eng.*, 30(1): 43–58, 2018. doi:10.1109/TKDE.2017.2754253.

[72] Jacob Siegal. E3 2018 press conference scorecard: Ranking microsoft, sony, nintendo and more, June 2018.

[73] Julia Silge and David Robinson. tidytext: Text mining and analysis using tidy data principles in r. *JOSS*, 1(3), 2016. doi:10.21105/joss.00037.

[74] Internet Live Stats. Twitter usage statistics, January 2018.

[75] Ingo Steinwart. On the optimal parameter choice for nu-support vector machines. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(10):1274–1284, October 2003. ISSN: 0162-8828. doi:10.1109/TPAMI.2003.1233901.

[76] Yee Whye Teh, Michael Jordan, Matthew Beal, and David Blei. Sharing clusters among related groups: Hierarchical dirichlet processes. In *Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems, NIPS 2004, December 13-18, 2004, Vancouver, British Columbia, Canada]*, pages 1385–1392, 2004.

[77] Mika Timonen, Paula Silvonen, and Melissa Kasari. Classification of short documents to categorize consumer opinions. *ADMA'11. Online proceedings*, 2011.

[78] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, Berlin, Heidelberg, 1995. ISBN: 0-387-94559-8.

[79] Vladimir Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

[80] Yang Wang, Payam Sabzmeydani, and Greg Mori. Semi-latent dirichlet allocation: A hierarchical model for human action recognition. In Ahmed Elgammal, Bodo Rosenhahn, and Reinhard Klette, editors, *Human Motion - Understanding, Modeling, Capture and Animation, Second Workshop, Human Motion 2007, Rio de Janeiro, Brazil, October 20, 2007, Proceedings*, volume 4814 of *Lecture Notes in Computer Science*, pages 240–254. Springer, 2007. ISBN: 978-3-540-75702-3. doi:10.1007/978-3-540-75703-0_17.

[81] Xing Wei and Bruce Croft. Investigating retrieval performance with manually-built topic models. In David Evans, Sadaoki Furui, and Chantal Soulé-Dupuy, editors, *Computer-Assisted Information Retrieval (Recherche d'Information et ses Applications) - RIAO 2007,*

8th International Conference, Carnegie Mellon University, Pittsburgh, PA, USA, May 30 - June 1, 2007. Proceedings, CD-ROM. CID, 2007.

[82] Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. Twitterrank: finding topic-sensitive influential twitterers. In Brian Davison, Torsten Suel, Nick Craswell, and Bing Liu, editors, *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010*, pages 261–270. ACM, 2010. ISBN: 978-1-60558-889-6. doi:10.1145/1718487.1718520.

[83] Feng Xiao, Tomoya Noro, and Takehiro Tokuda. News-topic oriented hashtag recommendation in twitter based on characteristic co-occurrence word detection. In Marco Brambilla, Takehiro Tokuda, and Robert Tolksdorf, editors, *Web Engineering - 12th International Conference, ICWE 2012, Berlin, Germany, July 23-27, 2012. Proceedings*, volume 7387 of *Lecture Notes in Computer Science*, pages 16–30. Springer, 2012. ISBN: 978-3-642-31752-1. doi:10.1007/978-3-642-31753-8_2.

[84] Wen-tau Yih, Joshua Goodman, and Vitor Carvalho. Finding advertising keywords on web pages. In Les Carr, David De Roure, Arun Iyengar, Carole Goble, and Michael Dahlin, editors, *Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, UK, May 23-26, 2006*, pages 213–222. ACM, 2006. ISBN: 1-59593-323-9. doi:10.1145/1135777.1135813.

[85] Eva Zangerle, Wolfgang Gassler, and Günther Specht. Recommending #-tags in twitter. In *Proceedings of the Workshop on Semantic Adaptive Social Web*, 2011.

[86] Lei Zhang and Bing Liu. Sentiment analysis and opinion mining. In Claude Sammut and Geoffrey Webb, editors, *Encyclopedia of Machine Learning and Data Mining*, pages 1152–1161. Springer, 2017. ISBN: 978-1-4899-7685-7. doi:10.1007/978-1-4899-7687-1_907.

[87] Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. Comparing twitter and traditional media using topic models. In Paul Clough, Colum Foley, Cathal Gurrin, Gareth Jones, Wessel Kraaij, Hyowon Lee, and Vanessa Murdock, editors, *Advances in Information Retrieval - 33rd European Conference on IR Research, ECIR 2011, Dublin, Ireland, April 18-21, 2011. Proceedings*, volume 6611 of *Lecture Notes in Computer Science*, pages 338–349. Springer, 2011. ISBN: 978-3-642-20160-8. doi:10.1007/978-3-642-20161-5_34.