POLITECNICO DI TORINO

Master degree course in Computer Engineering

Master Degree Thesis

Enhancing topic modeling through Latent Dirichlet Allocation with self-tuning strategies

Supervisor
prof. Tania Cerquitelli

Candidate
Stefano Proto

April 2018

Summary

With modern applications and technologies, an ever increasing quantity of data is produced every day in our lives and society. This applies also to textual data, being generated and collected from social networks to digital libraries. However, analysing and gather useful information from this huge amount of collected data is challenging and requires a lot of efforts, because both field expertise and computational cutting-edge technologies are needed to approach the issue. Specifically, text mining and topic modeling study algorithms to find previously unknown but potentially high-quality information from large document collections. Given the huge amount of data available, making the mining activity automatic is the natural subsequent step in information retrieval. Indeed, exploiting the data mining techniques to extract the information hidden in the collected data is not effective if a constant human supervision of the activity is needed. Since text mining is a multi-step process requiring specific configurations and parameters for every algorithm involved in the analysis, the presence of texts-field expertise and analysts should be required to guide the retrieving process. To overcome this problem, innovative solutions are needed to make the analysis of large data scalable and not supervised by data experts.
This thesis aims to implement a framework able to automatically cluster texts documents into cohesive and well separated groups, based on the content of the data. The framework should be able to relieve the analysts from selecting proper values for the analysis and the clustering processes, and scalable, to be able to address the dimensionality of datasets that are currently collected and so being a tool usable in the Big Data context.
The proposed solution is ToPIC (T*uning* o*f* P*arameters for* I*nference of* C*oncepts*), a distributed self-tuning engine able to describe corpora of text documents and divide them into correlated groups of documents with a similar topic. To purse this goal, ToPIC uses a probabilistic model of the data (i.e., Latent Dirichlet Allocation). Intuitively, in the LDA topic modeling, documents can be associated with a particular topic or can be seen as a mixture of topics in different proportions, and certain words can be expected to appear in a document more or less frequently. This topic modeling algorithm is then able to describe topics (and so documents) by means of similar word clusters. ToPIC includes different suitable data weighting functions, based on local and global weights, and a newly proposed approach

to automatically select optimal values for the number of topics, and so relieve the end-user of the burden of setting the model parameters. This approach, named ToPIC-Similarity, does not only consider the quantitative probabilistic indices of the whole model, but it considers the topics content and description, specifically evaluating their semantic similarity. ToPIC's current implementation runs on Apache Spark, a state-of-the-art distributed computing framework able to support large scale analytics, in order to overcome the bottleneck of the computational costs and so addressing the scalability requirements due to large dimensions of the datasets under analysis. ToPIC has been validated over different real data collections, having different textual characteristics and properties: two Wikipedia article collections and the Reuters-21578 dataset.
From the obtained experimental results, ToPIC turns out to be outperforming the current state-of-the-art approaches aiming to automatically select optimal number of clusters, both in the quality of the mining activity results and in the computational costs of the algorithms. Moreover, ToPIC resulted to be efficient in describing and clustering the given datasets, being an effective tool to streamline the analytics process and off-load the parameter tuning from end-user.

Acknowledgements

I would like to thank all the people who supported me writing this thesis in the last months.
First of all, I want to thank my supervisor professor Tania Cerquitelli, who gave me the chance to work on this interesting and challenging project, and mentored me with her helpful comments and observations. I would like then to express my gratitude to Evelina and Francesco, who guided me during this time with their knowledge and advices, always leading and pushing me to do my best. I thank them also for their friendship, and all the other people met in Lab5 for making the working environment welcoming, stimulating, and friendly. Moreover, I thank my family and my friends with which I shared this journey and who sustained me in these months.
To finish this preface, I would like to thank my assessors for reading this text.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

An ever increasing quantity of data is produced every day in our lives and society. However, analysing and gather useful information from this huge amount of collected data is challenging and requires a lot of efforts, because both field expertise and computational cutting-edge technologies are needed to approach the issue.
Data mining is the process of extracting high-quality information from row datasets. Mining information means extracting implicit and hidden knowledge from the data, analysing the structures of the datasets in order to find significative patterns or correlations among the data items. Data mining has become more and more relevant because of the great amount of data that is constantly produced.
The ever increasing production of data applies also to texts. From the social networks to the literary and scientific productions, from the e-learning platforms to the digital libraries, the amount of text documents that is daily produced grows continuously. Once collected, extracting information from the data is not a trivial task. It becomes natural to apply data mining processes and techniques to textual data: text mining is the branch of data mining that aims to gather useful and previously unknown information from textual data.
Given the pervasiveness of textual data, its wide range of applications in our lives and the difficulty of interpret poorly structured natural language data, text mining is one of the most challenging activities in the data mining universe. Because of these reasons, text mining and specifically topic modeling are currently areas of great interest and research for the scientific community. Text mining activities include building classification models, text categorization, grouping documents based on similar characteristics and contents, concepts/topics detection and extraction, sentiment analysis and texts summarization.
Since collecting textual data means gather the text productions of people, and since the human language and the text production is very diverse from person to person, using text documents as data to analyse is complicated and requires a lot of processing. Mining textual data is though made possible through natural language processing techniques and analytical methods, that aim to model and structure the

textual sources in order to make them accessible for the data analysis and investigation. These involve several steps and phases, such as the syntax and the semantic analysis.
In the scientific research, several approaches and solutions have been attempted and proposed in order to firstly represent and then mine and retrieve information from the text sources. Depending on the modeling of the text data and the used techniques, different models have been proposed in the scientific literature: *set-theoretic* (such as the Boolean models, representing documents as sets of words or phrases), *algebraic* (representing documents as vectors or matrices, such as the Vector Space models and the Latent Semantic Analysis) and *probabilistic* (such as the Latent Dirichlet Allocation, representing documents as probabilities of words).
However, besides the used approach to analyse the text documents, given the huge amount of data available, making the mining activity automatic is the natural subsequent step in information retrieval. Indeed, exploiting the data mining techniques to extract the information hidden in the collected data is not effective if a constant human supervision of the activity is needed. This happens also in the case of text mining. Indeed, being text mining a multi-step process requiring specific configurations and parameters for every algorithm involved in the analysis, the presence of texts-field expertise and analysts should be required to guide the retrieving process. To overcome this problem, innovative solutions are needed to make the analysis of large data scalable and not supervised by human analysts and data experts.

Cutting edge technologies and approaches to make text analysis and mining automatic are hot topics in scientific research. Some solutions to target the issue have already been proposed, but many other efforts are needed to improve and make automatic text mining more effective.
The goal of the thesis is to implement a framework able to automatically cluster texts documents into cohesive and well separated groups, based on the content of the data. The framework should be able to relieve the analysts from selecting proper values for the analysis and the clustering processes. Moreover, the framework should be scalable and distributed to be able to address the dimensionality of the datasets that are currently collected, so being a tool usable in the Big Data context.
In detail, a probabilistic model will be used in order to retrieve information from the data collections, and a novel approach to determine the optimal number of topics in a corpus of documents will be newly proposed.
After designing and implementing the framework and its building components, some experiments will be carried out to evaluate the effectiveness of the produced solution.

The thesis text is structured as follows. Chapter 2 presents a discussion about textual data analysis. It will describe the background and the related works of this study, specifically the PASTA analytics engine and the state-of-the-art techniques

used to determine the optimal number topics for the considered probabilistic topic modeling. Chapter 3 briefly introduces the readers to the Big Data platforms, and specifically to the Apache Spark framework, base of the architecture of the implemented framework ToPIC. Chapter 4 describes the Latent Dirichlet Allocation, the topic modeling algorithm used in ToPIC, and the design of the framework, together with its building parts. The Chapter shows the used approach, describing in detail the analytic flow of ToPIC; it also presents and defines the ToPIC-Similarity index, the newly proposed approach to determine how well the LDA modeled topics represents cohesive and well separated concepts. In the same Chapter the methodologies used to evaluate the obtained results are explained. Chapter 5 presents the datasets the framework has been tested with, characterizing them by means of statistical indices and textual characteristics. Continuing on in the Chapter, the results obtained for a representative running example are shown to the readers. Then, the impacts of the weighting schemas are illustrated, together with a comparison with the state-of-the-art techniques. The Chapter ends with a general evaluation of the framework and some final observations on the ToPIC performance. Chapter 6 concludes the study, with a look into the key contributions of the thesis and a discussion about possible future research works in this area.

Chapter 2

Textual data analysis

This Chapter introduces the background and related works explored during this study in order to achieve the stated goals, being them the starting point of this thesis and essential components of the produced work. Section 2.1 illustrates PASTA, an engine developed to do text analysis in a self-tuning manner. This engine has been taken as the starting point from which develop further methodologies to expand and contribute the current text analysis cutting-edge. Section 2.2 introduces the probabilistic topic modeling algorithms, describing the goals and the approach used to group textual documents based on their argument and contents. Specifically, the Latent Dirichlet Allocation is briefly introduced to the readers. The state-of-the-art techniques used in literature to find optimal parameters for the modeling and make the LDA processing automatic are presented in Section 2.3. A gap analysis about the lack of stable, valid and self-tuning methods to determine the LDA parameters and configuration concludes the Chapter in Section 2.4. In this Section, an additional consideration about the presented engine and its possible enhancements is also given, thus proposing this study as the natural development of PASTA.

2.1 PASTA framework

PArameter-free Solutions for Textual Analysis (PASTA) is a distributed data analysis engine, which aim is to make the analysis of huge text datasets carried out without the intervention of data-field experts nor data analysts [7].
The engine automatically sets parameters and tunes the text clustering algorithms extracting the knowledge in behalf of the end-user, still achieving optimal quality results. PASTA is then capable to suggest to the users the optimal weighting schema and reduction functions for the dataset under analysis, together with the more suitable clustering configuration in order to describe as good as possible the given dataset [7].
Figure 2.1 shows the PASTA architecture. In the following part of Section, the

PASTA main characteristics and activities are described in detail.

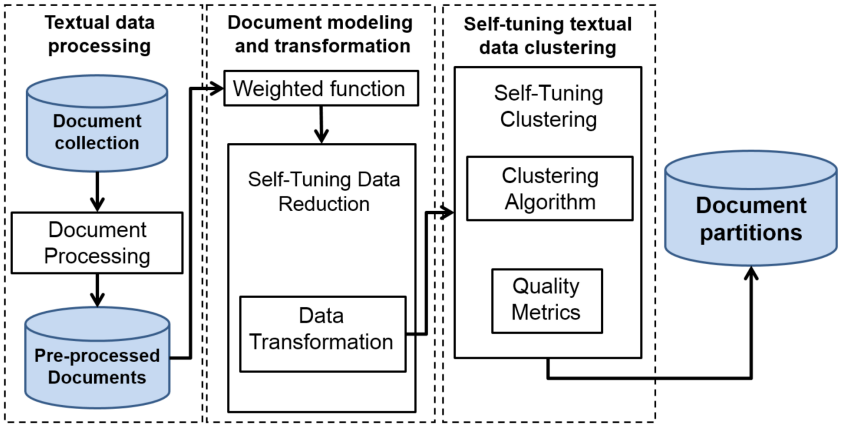


Figure 2.1: PASTA architecture © *Self-tuning techniques for large scale cluster analysis on textual data collections* [7].

2.1.1 PASTA data preprocessing and weighting

Text data are by their nature very variable and dirty, differing a lot based on the typology, the source, the target and the field of expertise. Because of this, before analysing the documents several preprocessing steps are needed:

1. **Document splitting**: depending on the next tasks, documents can be analysed in their entirety or split in paragraphs or sections;

2. **Tokenization**: the texts are split in a set of words, named *tokens*;

3. **Stopwords removal**: the meaningless words (e.g. words that appear often in the texts, such as articles or prepositions) are removed, since they do not add any information to the analysis;

4. **Stemming**: the words are reduced to their base form (*stem*), removing prefixes and suffixes. This reduces the dictionary size and groups words with the same root;

5. **Case normalization**: letters are all transformed in lower case characters;

6. **Weighting**: words may have different weights within a document or within the corpus. This affects the final clustering in topics; more details about this procedure will be given later in this Section.

After the preprocessing, the documents are represented in the *Bag-Of-Works* form, that describes texts disregarding the terms order and the grammar rules, but however representing the main themes [18].
Besides these steps, the framework also computes several statistical indices in order to characterizes the documents dataset [4]. These indices are specific for the textual analysis context; they are:

- *Number of categories*: if known, it represents the original number of different topics in the dataset that is going to be analysed;
- *Average frequency of the terms*: the average frequency of the terms occurrences among all the words in the corpus;
- *Maximum frequency*: the greatest frequency of occurrences of a term in the corpus;
- *Minimum frequency*: the lowest frequency of occurrences of a term in the corpus;
- *Number of documents*: the number of documents in the given corpus;
- *Number of terms*: the number of words in the corpus, considering the repetitions;
- *Dictionary*: the number of words in the corpus, not considering the repetitions;
- *Type-Token Ratio (TTR)*: the ratio between the *Dictionary* cardinality and the total *Number of terms* in the corpus. It represents the vocabulary variation within a text [1];
- *Hapax %*: the ratio between the number of terms with only one occurrence in the corpus (called *hapax*) and the *Dictionary* (dictionary variety);
- *Guiraud Index*: the ratio between the *Dictionary* cardinality and the square root of the *Number of documents*, in order to highlight the lexical richness of the corpus.

In order to better identify the correct topic of a document and help the clustering process to group similar documents together, weights can be assigned to all the terms in the corpus. The weights measure the relevance the terms have in the documents, and they are computed as the product of a local and a global weight. The weight a word has within the document is called **local weight**, while the weight it has with respect to the whole corpus is called **global weight**.
The weights are stored in a matrix X where the rows are associated to the documents and the columns to the terms: $x_{i,j} = l_{i,j} \times g_{i,j}$ represents the weight of the term j

in document i. PASTA explores different weighting strategies, combining weights from both the categories: *Term Frequency* (TF) and *Logarithmic term frequency* (LogTF) are the local weights, while *Inverse Document Frequency* (IDF) is the considered global one. The weighting functions used in PASTA, along with their definitions, are shown in table 2.1.

Weight	**Definition**
Local	$\mathrm{TF} = \mathrm{tf}_{ij}$ $\mathrm{LogTF} = \log_2(\mathrm{tf}_{ij} + 1)$
Global	$\mathrm{IDF} = \log\left(\frac{\lvert D\rvert}{df_i}\right)$

Table 2.1: PASTA weighting functions.

2.1.2 PASTA data reduction and clustering

In order to reduce the dimensionality of the corpus and focus the computation only on the most relevant concepts of the documents, a data transformation is needed. The weighted matrix X representing the documents in the corpus is reduced in size using LSI (*Latent Semantic Indexing*). For a given reduction factor, LSI allows to reduce the dimensionality of the matrix without losing significant information. However, choosing the most proper reduction factor is not a trivial task.
To reduce the dimensionality of the matrix and uncover the hidden concepts in the dataset, LSI applies the SVD transformation (*Singular Value Decomposition*) to X, in order to decompose the matrix into the product of its component U, S and V^T. The matrix S represents the singular values of the dataset under analysis, one for each dimension (term). Based on the magnitudes of the singular values in S it is possible to approximate the matrix X.
Keeping only the most K_r significant terms corresponding to the most significant singular values, and ignoring the other dimensions in S, the original matrix X can be reduced by reducing all the components to rank K_r: $X_{K_r} = U_{K_r}\ S_{K_r}\ {V^T}_{K_r}$ [7].
As mentioned before, finding the correct dimensionality reduction is not a trivial task, since too few dimensions would lead to a poor data representation, while too many dimensions would lead to a noisy dataset. This open research issue is solved in PASTA by ST-DaRe (*Self-Tuning Data Reduction*). The ST-DaRe algorithm automatically chooses three reduction parameters that will be used by LSI or PCA to decrease the dimensionality of the dataset under analysis. This algorithm, enhanced in e-PASTA, firstly considers only the first greatest 100 singular values, then computes their mean value and the standard deviation [8]. ST-DaRe selects as reduction values the following ones:

- the singular value corresponding to the mean;

- the singular value corresponding to the mean plus the standard deviation;
- the singular value corresponding to the mean of the two previously chosen parameters.

Once the reduction has been done, PASTA clusters the documents in the corpus via K-Means. This clustering algorithm uses a partitional strategy to divide the corpus in K_{cl} different clusters. Initially, K_{cl} random document are chosen to be the centroids (the points representing the whole cluster, i.e. the mean of the objects in the cluster itself). At every iteration, all the documents are assigned to same cluster of the centroid they are nearer. The centroids are then recomputed for each cluster, and so relocated. This routine continues until the centroids do not change anymore. K-Means well performs, being computationally fast and producing tight clusters. Nevertheless, the number of clusters K_{cl} has to be set a priori. In PASTA this issue is addressed by the ST-C (*Self-Tuning Clustering*) algorithm, that automatically chooses a good number of cluster for the dataset under analysis, using quality metrics (such as Purified-Silhouette and Weighted-Silhouette) and agreement measure (such as Rand-index). For each reduction value K_r, ST-C iterates over a set [min_{cl} - max_{cl}] of possible number of clusters K_{cl} given by the analyst: the clustering over K_r is performed by the K-Means algorithm and the result is stored. The qualities of the clustering results are then evaluated by means of the mentioned indices, and the partition with the highest Purified and Weighted Silhouette indices is chosen among the others. Partitions with Rand-index higher than 0.9 are considered equivalent to one picked by ST-C, and then proposed to the user.

2.2 Probabilistic topic modeling

A completely different approach from the one presented in the previous Section, is the probabilistic topics modeling approach. This technique represents textual documents as probabilities of words and aims to discover and annotate large archives of texts with thematic information.
Probabilistic topic modeling algorithms are based on statistical methods that analyse the original texts and their words in order to discover the arguments they go through, and to which other documents they are related. These algorithms do not require any prior annotation or labeling of the documents, but they are able to describe corpora of documents without previous knowledge of the datasets.
The *Latent Dirichelt Allocation* (LDA) is one the most famous and most used probabilistic topic modeling algorithm. The intuition behind LDA is that documents are mixtures of multiple topics [2]. Topics are defined to be distributions over a fixed vocabulary. Documents, instead, are seen as a distribution over the set of different topics, thus showing multiple topics in different proportions. Finally, the LDA

algorithm models the given textual dataset with document-topics and topic-terms probabilities distributions.
LDA can be used to infer the topic hidden in a textual dataset. However, as most of the topic modeling algorithms, LDA requires the number to topics to be previously known and fixed. However, finding the optimal values for the number of topics that have to be discovered by the LDA is not trivial, and it is instead an open issue in the scientific community. In the following Section, two state-of-the-art approaches to automatically determine the optimal number of clusters needed in the topic modeling will be presented.

2.3 LDA number of topics selection

In many clustering algorithms finding the optimal value for the number of topics K is not trivial. This is also the case for the Latent Dirichlet Allocation algorithm. The number of topics has great influence on the results of the clustering process, but often the evaluation of the results is subjective, difficult to be interpreted and time-consuming.
This model parameter has to be set carefully, since based on it the number of clusters, and so the final clustering result, will drastically change. Indeed, too low K values would lead LDA to be too coarse to be able to identify proper clusters, while K values that are too big would lead to a very complex model, difficult to be interpreted and difficult to be validated.
In this research, three different approaches to find the most suitable value K have been studied and explored. This Section will present two approaches considered as the state-of-the-art techniques, while a third method, that aims to improve the results obtained with the following ones, will be newly proposed in Chapter 4.

2.3.1 RPC approach

The *Rate of Perplexity Change* (*RPC*), proposed by Zhao et al. [23], is a heuristic approach aiming to estimate the best value for the number of topics K for the LDA model.
Proposing this strategy, the authors wanted to overcome the problems of perplexity, the methodology originally proposed to evaluate the LDA models and determine which clustering statistically better describes a given dataset. According to [3], the lower the perplexity of a model, the better it performs describing the data collection (a detailed description of perplexity as evaluation index is given in Section 4.6).
Finding this approach not stable and too varying, even for the same dataset, the RPC strategy aims to outperform the perplexity approach.
This method, claimed to be stable and effective, considers how the variation of the average perplexities P_i for K candidate number of topics (P_1, ..., P_i, ... P_K) changes

with respect to the topics numbers K_i $(1 < i \leq K)$ [23]. The equation of the RPC index is:

$$RPC(i) = \left| \frac{P_i - P_{i\text{-}1}}{K_i - K_{i\text{-}1}} \right| \tag{2.1}$$

Given the definition of the RPC function, the first change point of the RPC curve, i.e. the first i that satisfies the equation $RPC(i) < RPC(i+1)$, is chosen to be the most suitable value for the number of topics K.
Figure 2.2 shows an example of the RPC curve, obtained for dataset D1, TF-IDF weighting schema. In this case, the K value proposed by the RPC strategy turns out to be 3.

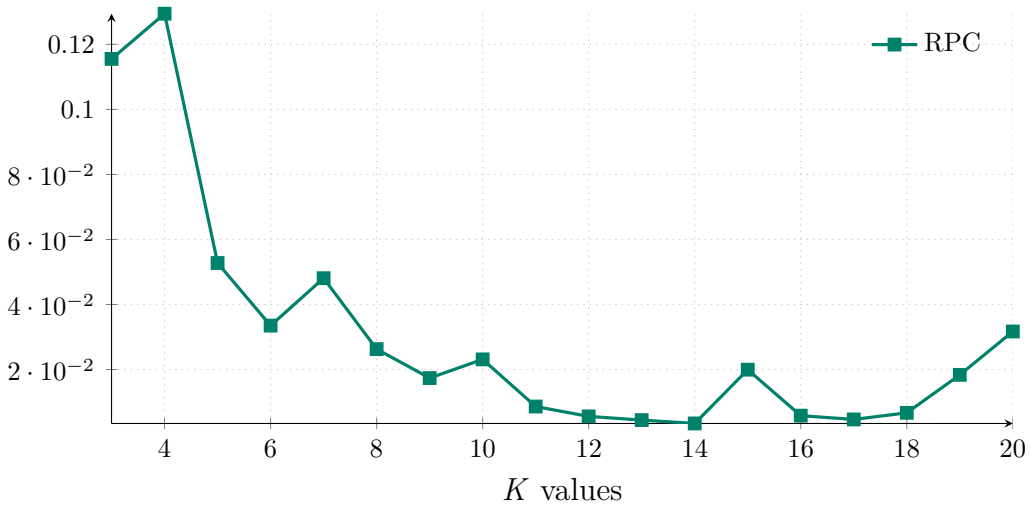


Figure 2.2: Example of RPC chart by different K values, dataset D1, TF-IDF weighting schema.

2.3.2 En-LDA approach

The *Entropy optimized Latent Dirichlet Allocation* (*En-LDA*) [22], is an entropy-based measure to optimally select the number of topics needed to properly describe a corpus using the LDA topic modeling.
Entropy measures the average information contained in an event. Generally, information can also be seen as the uncertainty characterizing the probabilistic event itself: in our case, considering the probabilistic model generated by the LDA, entropy represents the uncertainty of the model when describing the dataset under analysis.
The idea of the En-LDA authors is to measure the entropy of different LDA models

obtained with different configurations, to assess which of them is likely to be the better one. To do that, they measured the entropy of each term given a document d_m using topics as the probabilistic labels of the word. Considering all the documents in a corpus, the entropy of all the words are then aggregated to estimate the overall entropy of the terms given the distribution of words with respect to the topics and the distribution of the topics with respect to the documents.
The overall entropy of the clustering is then measured as:

$$Entropy(K) = \sum_{m=1}^{M} \sum_{k=1}^{K} p(z_m = k|d = d_m) \left(\sum_{n=1}^{N_m} -p_{t,k,m} \; ln(p_{t,k,m}) \right) \tag{2.2}$$

where M is the number of documents in the corpus, K is the number of topics in the clustering, N_m is the number of terms in document d_m, z_m the document's topic and z_n the topic of the word w_n. $p_{t,k,m}$ is instead the normalized probability of the word $w_n = t$ with respect to the topic $z_n = k$. In other words, $p_{t,k,m}$ represents the probability that the term w_n is t under the kth topic. In formula, this is expressed by: $p_{t,k,m} = \frac{p(w_n = t|z_n = k)}{\sum_{n=1}^{V_m} p(w_n = t|z_n = k)}$, having V_m equal to the size of the vocabulary for the document d_m.
To choose the most proper K number of topics, several LDA models have to be computed for different values of K, and then the one with the minimum value of entropy is selected to be the optimal number of partitions to describe the data collection.
The approach is considered to be stable and able to handle very low and very high number of clusters. Indeed, very low K values lead to very large entropy values, as well as very large number of clusters.
Figure 2.3 shows an example of the En-LDA measure, obtained for dataset D1, TF-IDF weighting schema. In this case, the K value proposed by the En-LDA approach results to be 19.

2.4 Gap analysis

In this study two main objective are pursued. Firstly, this study improves and enhances the PASTA engine, implementing an alternative to the K-Means clustering technique. The proposed solution, that exploits the LDA modeling, potentially is also usable as a reduction technique, becoming a further alternative to the LSI reduction. LDA, being probabilistic and a state-of-the-art technique for text based information retrieval, perfectly suits in the considered data analysis engine as a complementary approach for the text mining activity.
Moreover, since the LDA model and the related literature do not provide a stable and a resilient method to find a proper number of topics for the dataset under analysis, the second goal of this study is to propose a novel method to determine

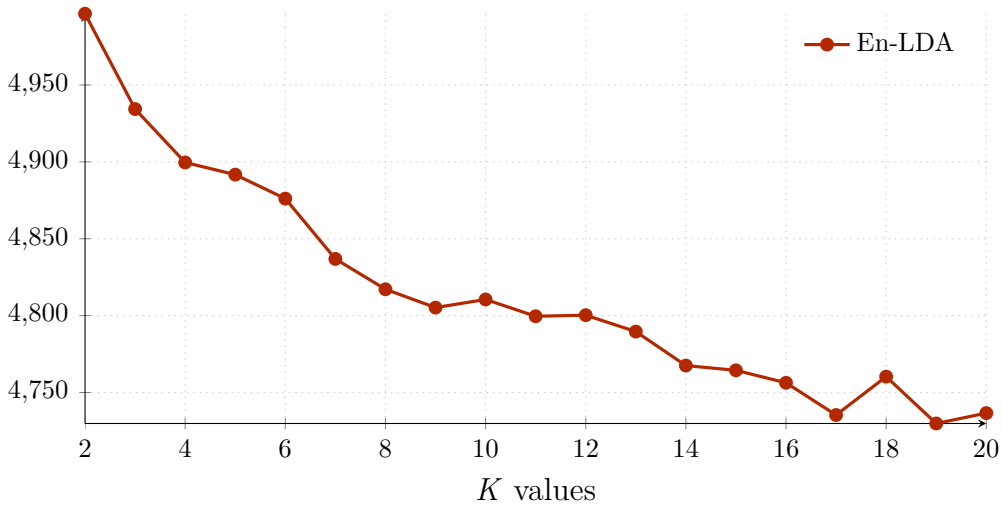


Figure 2.3: Example of En-LDA chart by different K values, dataset D1, TF-IDF weighting schema.

an optimal number of topics (clusters) for the LDA topic modeling. Particularly, the state-of-the-art techniques only consider the probabilistic properties of the LDA model. This study aims instead to consider the semantic representation of the topics in order to find cohesive and well separated groups cluster that well represent the documents content.

Chapter 3

Big Data platforms

In the current days, the data production and the knowledge digitilazion rates increase continuously. To overcome the critical bottleneck of the computational costs, innovative algorithms and methodologies able to support large scale analytics are needed. Several studies had proposed many innovative solutions. Among them, *MapReduce* [6] and *Apache Spark* [21] are the most used and famous frameworks able to support large scale analytics.
As initially said, ToPIC has been thought to be distributed and scalable. In order to achieve these goals and being an effective tool usable in the Big Data context, ToPIC's current implementation runs on Apache Spark, the current state-of-the-art distributed framework to make the storage and the analysis of the data parallel and distributed.
This Chapter describes the Apache Spark framework and its main characteristics.

3.1 Spark framework

Apache Spark is an open source cluster computing platform, created at the University of California and then donated to the Apache Software Foundation. The Spark framework is designed to be fast and general-purpose [10]. These characteristics made it a valid alternative to *Hadoop MapReduce* (Spark results to be 10 to 20 times faster than MapReduce[1]) and one of the most used framework for Machine Learning algorithms and Big Data analysis.
The Spark architecture is based on a distributed storage system, maintained in a fault-tolerant way. Furthermore, the Spark framework uses a cluster manager and a computational engine that allow scheduling, distribution and monitoring of the applications, making the computation fast and optimized. Being general purpose,

[1] `https://spark.apache.org/`

Spark allows the users to cover and combine several different types of workloads, from batch programming to streaming and real-time data processing applications, from interactive SQL queries to machine learning tasks, all within the same project. Spark offers its functionalities through API for different programming languages (Java, Scala, Python, R).

3.1.1 Spark Core

As depicted by its name, the Spark Core is the basis of the Spark projects. It is the *computational engine* in charge of the distributed aspects of the framework, also providing the I/O functionalities, memory management, the dispatching and the scheduling of the tasks on the cluster machines. The Spark Core exposes these functionalities above the abstraction of the *Resilient Distributed Dataset* (RDD), offering APIs to build and manipulate these collections and making easy to write parallel applications.
Applications are executed on Spark as a set of independent processes, coordinated by the `SparkContext` in the main program (called *driver* program). Specifically, the SparkContext connects to a Cluster Manager (supporting Spark to be deployed in a distributed manner over the *Apache Mesos* or the *Apache Hadoop YARN* platforms, or to be deployed in a Standalone fashion), that assigns the resources to the applications. The processes actually performing the computation and the operations on the cluster nodes are called *executor*; they receive the tasks from a driver program, execute the required transformations and actions on the data, and eventually return the results to the driver, that needs to be listening to inbound connections and reachable for the whole execution of the application. The application executors remain alive for the whole application life time, and they can execute tasks by means of different threads.

3.1.2 Spark RDDs

The *Resilient Distributed Dataset* (RDD), is the main and fundamental abstraction of Spark, by means of which it offers its functionalities and APIs. An RDD is an immutable collection of objects, that may be distributed over different nodes by means of several partitions. The RDD abstraction allows the data structures and their partitioning to be transparent to the developers and lets the programming aspects of the project remaining at a high level. Abstracting the data storage and partitioning, the RDDs allow Spark to be deployed over different file systems, such as *HDFS* (*Hadoop Distributed File System*), *Cassandra*, *HBase* and *S3*. Among the several properties of the RDDs, the following ones are the most important:

- **Resilient**: RRDs are fault tolerant, they can be recomputed in case of damages or failures of the nodes of the cluster;

- **Distributed**: the data collection can reside over different nodes of the cluster;
- **Dataset**: the RDD is a collection of partitioned data, that can be represented by primitive values, tuple of values or any type of other objects.

RDDs have many other properties. Among the others, they are:

- **Immutable and read-only**: the data collection stored in the RDD can not be modified, and transformations on the objects will produce a new RDD;
- **In-memory**: data inside RDD is stored in memory as much (size) and long (time) as possible;
- **Lazy Evaluated**: the data inside RDD is not available or transformed until an action is executed that triggers the execution;
- **Cacheable**: data can be hold in a persistent *storage* like memory (default and the most preferred) or disk (less preferred due to access speed);
- **Parallel**: data stored in the RDDs can be processed in parallel;
- **Typed**: data in RDDs must have a type, such as `Long` in `RDD[Long]` or `(Int, String)` in `RDD[(Int, String)]`;
- **Partitioned**: records are partitioned (split into logical partitions) and distributed across nodes in a cluster;

From Spark 2.x, the RDD is not the primary API anymore, but the use of Dataset API is encouraged, having the RDD technology still underlying the Dataset.

3.1.3 Spark transformations and actions

The RDD abstraction basically offers two kinds of operations: *transformations* and *actions*.
Transformations build a new RDD from a previous one, applying functions to the elements of the RDD. Examples of transformations are the `rdd.filter()` and the `rdd.map()` functions. Spark keeps track of all the dependencies between the RDDs, building a *lineage graph* (basically a directed acyclic graph of the transformations performed over the data). The lineage graph is used to achieve the fault-tolerance, since by means of the transformations history it is possible to recompute the RDDs in case of errors.
The *actions*, instead, start from an RDD to compute a result, that may be returned to the driver program or saved in an external storage system (such as HDFS). Actions force the evaluation of the transformations, since to perform the actions the

RDD is actually necessary. Examples of actions are `rdd.count()` and `rdd.take()`. Transformations and actions differ in their execution, since Spark computes and creates new RDDs only when they are used in an action for the first time (*lazy fashion*), even if the user can define RDDs anytime in the program.
The lineage graph built by Spark is indeed used in the lazy evaluation of the transformations when creating the new RDDs. The lazy evaluation approach makes sense in the Big Data context: Spark computes only the final results, effectively needed by the user, optimizing the computation to save operations, readings and writings of intermediate unnecessary states. Another advantage introduced by the lineage graph is that the directed acyclic graphs allow a way more flexible approach to the data transformation with respect to more rigid *map-then-reduce* approach.

3.1.4 Spark libraries

Spark is designed to be highly accessible, offering simple APIs in Python, Java, Scala, and SQL, and rich built-in libraries.
The Spark libraries are tightly integrated in the framework; being high-level components, the Spark stack allows the libraries to benefit of all the functionalities and improvements of the lower levels. Furthermore, the libraries are independent of the structure and the architecture beneath them. The Spark stack, showed in picture 3.1, allows Spark to be flexible, usable in a wide range of applications and context, deployable in many different systems and in continuous evolution.

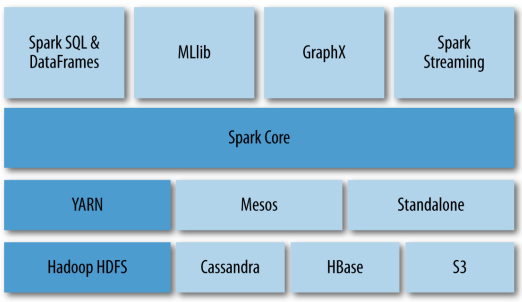


Figure 3.1: Spark stack.

Spark SQL

Spark SQL is the Spark interface that allows the users to work with structured data. It introduces the data abstraction called *DataFrame*, that allows developers to use SQL queries together with the standard RDD manipulations, all within the same application. DataFrames are distributed sets of data organized in columns, conceptually similar to the tables of the relational databases.

Spark Streaming

Spark Streaming is the component allowing Spark to process live streams of data and perform streaming analytics. Spark Streaming introduces the *DStreams* (Discretized Streams), namely sequences of RDDs arriving over time, at each time step. Programmers can easily manipulate data in memory, on disk, or arriving in real time, having the same benefits of tolerance, throughput and scalability of all the Spark applications. Spark Streaming allows batch and analytics functionalities to be merged together in the same system.

Spark MLlib

Spark Mlib (Machine Leaning library) is a distributed framework offering to the Spark users the most used and well-known machine learning functionalities and algorithms. Data import, classification, regression, clustering, collaborative filtering and model evaluation are some of the functionalities contained in MLlib, furthermore designed to be scalable and run in parallel across the cluster.

Spark GraphX

Spark GraphX is the package used in Spark to manipulate graphs and perform graph-parallel computations. It extends the Spark RDD APIs to allow the user directly creating graphs with custom properties attached to the edges or to the vertices. The library also proposes some basic graph manipulation algorithms.

Chapter 4

ToPIC implementation

This chapter introduces the ToPIC (T*uning* o*f* P*arameters for* I*nference of* C*oncepts*) framework, its implementation and its novel approach to automatically determine the most suitable number of topics for the LDA algorithm. Section 4.1 gives a detailed description of the Latent Dirichlet Allocation. Being LDA the specific topic modeling approach in this study, a more detailed observation of the model is required in order to understand the underlying bases of the presented work. Section 4.2 gives an overview of the whole system implementation and its execution flow. The next Sections provide a more detailed description of building blocks of the system, namely the preprocessing of the corpus and the weighting of the terms (described in Section 4.3), and the application of the Latent Dirichlet Allocation modeling to cluster the documents based on their topics distribution and content (Section 4.4). ToPIC-Similarity, the new evaluation index used to determine the most proper number of topics in which the documents have to divided in, is proposed and illustrated in Section 4.5. The Chapter ends with Section 4.6 that proposes several techniques (quantitative and qualitative) to compare and evaluate the results of the LDA models, visualize the obtained outcomes and assess the effectiveness and the efficiency of the framework.

4.1 Latent Dirichlet Allocation

The *Latent Dirichlet Allocation* (LDA) is an unsupervised generative probabilistic model for collections of discrete data such as text corpora [3]. Willing to generalize and improve the existing and known approaches for text modeling, D. Blei, A. Ng and M. Jordan created LDA, a model able to shortly describe large collections of data without resorting to data dimensionality reductions.
In the following part of the Section, the LDA document generative model and its application to inferential problems are explained.

Hereafter, a *document* is defined as a sequence of N words (called *terms* in the text modeling context) and denoted by $\mathbf{w} = (w_1,\ w_2,\ ...\ w_{Nd})$; a *corpus*, instead, is a collection of M documents and it is denoted by $\mathcal{D} = \{\ \mathbf{w}_1,\ \mathbf{w}_2,\ ...\ \mathbf{w}_M\ \}$.

4.1.1 LDA generative model

Before showing and giving the details of the LDA topic modeling, taking into account the assumptions behind the modeling algorithm will help the comprehension. The LDA authors, as it was done for all the other text modeling approaches known so far, based their considerations on the *Bag-of-Words* representation and assumption. The *BoW* representation allows documents to be described as sparse vectors containing the number of the tokens occurrences. This document representation assumes that the order of the terms in a document is irrelevant. At the same time, the *Bag-of-Words* assumption entails that also the documents order in the corpus can be neglected. On this basis, the LDA modeling algorithm takes in consideration the *exchangeability* property of both terms and documents. This assumption does not necessarily lead to strategies that are restricted to simple frequency counts or linear operations, but it is still possible to capture significant intra-document statistical structures via mixing distributions [3].
LDA is a generative model able to create corpora of documents. In order to create them, topics and words have to be characterized as probabilistic distributions, since the model draws the elements of the documents based on these probabilities.
The probabilities that are used in the model are the following:

- Poisson(λ), represents the distribution of the documents lengths.
- $\boldsymbol{\theta}$, describes the document-topics distributions, that are the probabilities that a given document d belongs to a certain topic k: documents are then seen as a distribution over the latent topics. This distribution is described by the Dirichlet distribution Dir($\alpha_1,\ \alpha_2,\ ...\ \alpha_k$).
- $\boldsymbol{\phi}$, represents the topic-words distribution. It describes the probabilities that words have to be drawn for each specific topic.

Given the described distributions, the steps needed to build a document are:

1. Choose N_d, the number of words in the document, from Poisson(λ)
2. For each of the N_d words w_n:

   Choose a topic $z_n \in \{\ 1,\ 2,\ ...\ k\ \}$ from Multinomial($\boldsymbol{\theta}$)

   Choose a word w_n from Multinomial($\boldsymbol{\phi}_{z_n}$), conditioned on the topic z_n

This document generating process has to be repeated for all the documents belonging to the corpus.
In other words, for each document in the corpus and for each word w_n in the N_d number of words, a topic is chosen accordingly to the document-topics distribution (Multinomial($\boldsymbol{\theta}$)), thus generating a mixture of topics for each document. Then, a word is extracted from the vocabulary V, taking into account the terms probabilities for each given topic of the documents mixture [3].
The number of topics K, and so the dimensionality of the Dirichlet distribution, is assumed to be known and fixed in the documents creation, while the parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ have to be given to the model in order to be built.
A further consideration is that all the variables that contribute to generate the corpus ($\boldsymbol{\theta}$ and $\mathbf{z}$) are independent from the number of terms in each document N_d; this parameter is then generally not considered in the model discussions, and it will not appear in the following discussion.

Given $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, the joint multivariate distribution of the topic mixture $\boldsymbol{\theta}$, the set of N topics $\mathbf{z}$ and the set of N terms $\mathbf{w}$ is given by:

$$p(\boldsymbol{\theta}, \mathbf{z}, \mathbf{w}|\boldsymbol{\alpha}, \boldsymbol{\beta}) = p(\boldsymbol{\theta}|\boldsymbol{\alpha}) \prod_{n=1}^{N_d} p(z_n|\boldsymbol{\theta}) p(w_n|z_n, \boldsymbol{\beta}) \tag{4.1}$$

From the equation 4.1, it is possible to compute the probability of the whole corpus by integrating over $\boldsymbol{\theta}$, summing over $\mathbf{z}$ and then taking the product of the marginal probabilities of the single documents:

$$p(\mathcal{D}|\boldsymbol{\alpha}, \boldsymbol{\beta}) = \prod_{d=1}^{M} \int p(\boldsymbol{\theta}_d|\boldsymbol{\alpha}) \left(\prod_{n=1}^{N_d} \sum_{z_{dn}} p(z_n|\boldsymbol{\theta}_d) p(w_{dn}|z_{dn}, \boldsymbol{\beta}) \right) d\boldsymbol{\theta}_d \tag{4.2}$$

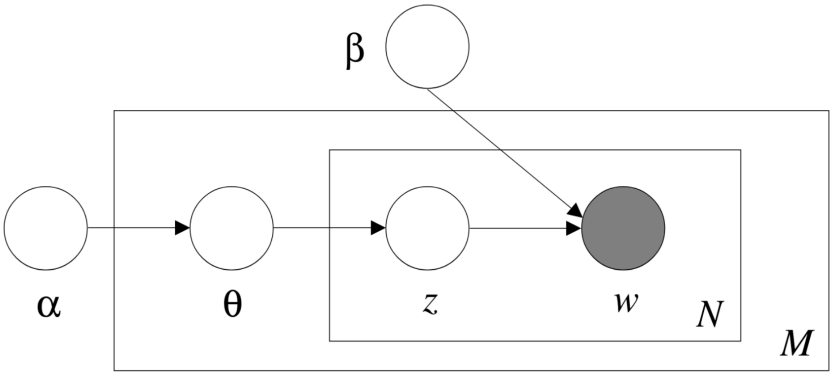


Figure 4.1: Graphical representation of the LDA © *Latent Dirichlet Allocation* [3].

The LDA model can be graphically represented with the schema reported in figure 4.1. As shown, there are three different levels in the schema, representing the corpus, the documents and the terms. In the most inner box, representing the terms, z and w are sampled for every n: this means that also the topic is sampled for every term in the document, giving as result that the same document can be described by multiple topics. The variables $\boldsymbol{\theta_d}$ are sampled once per document, while the parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are sampled once for the whole document generating process.

4.1.2 LDA for inferential problems

The Latent Dirichlet Allocation model is generative; nevertheless, it can used to do inference of the posterior distribution of the latent variables for a given corpus and then recover its structure [3]. The variables that describe the document are the distributions of the topic mixture $\boldsymbol{\theta}$ and the set of N topics $\mathbf{z}$:

$$p(\boldsymbol{\theta}, \mathbf{z}|\mathbf{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{p(\boldsymbol{\theta}, \mathbf{z}, \mathbf{w}|\boldsymbol{\alpha}, \boldsymbol{\beta})}{p(\mathbf{w}|\boldsymbol{\alpha}, \boldsymbol{\beta})} \tag{4.3}$$

However, it is generally unfeasible to compute these distributions since the integral in the expression in intractable, and so it is impossible to exactly solve this posterior Bayesian inferential problem. To overtake this issue, several approximate inference algorithms have been proposed and considered for the LDA inferential problem.
The original Latent Dirichlet Allocation paper used proposed a variational Bayes approximation, but many other alternatives, such as the Markov chain Monte Carlo approach, are available to approximate the probability distributions.

4.1.3 LDA inferential example

The following example clearly explains the LDA topic modeling for a simple corpus. Let us consider a corpus made of the following documents:

- I like to eat broccoli and bananas.
- I ate a banana and spinach smoothie for breakfast.
- Chinchillas and kittens are cute.
- My sister adopted a kitten yesterday.
- Look at this cute hamster munching on a piece of broccoli.

In this example, the number of latent topics will be set to 2; the LDA algorithm discovers the following topics:

- **Topic A**: 30% broccoli, 15% bananas, 10% breakfast, 10% munching

- **Topic B**: 20% chinchillas, 20% kittens, 20% cute, 15% hamster

LDA describes the topics by terms probabilities, from which it is possible to infer the arguments they deal about; at this point, documents can be modeled and described by latent topic descriptions:

- **Documents 1 and 2**: 100% Topic A
- **Documents 3 and 4**: 100% Topic B
- **Document 5**: 60% Topic A, 40% Topic B

4.2 System overview

The ToPIC framework started with the aim to enhance the PASTA engine, applying a new method to cluster the documents in the corpus. The considered enhancement is to apply the LDA algorithm to the datasets under analysis.
Using this method to model the topics of the text documents, the reduction phase of the PASTA engine is no more needed, since LDA reduces itself the documents and, instead, could be even used as a reduction technique.
Before applying the LDA algorithm to the dataset, the steps that were done in PASTA before the reduction are executed as well in this new the framework, since preprocessing and weighting are essential to properly model the documents. After these steps, several LDA models are iteratively built on the dataset, varying the number of clusters the documents have to be assigned to.
In order to stop the computation and find a trade-off between costs and benefits, a new quality metric is used as stopping criterion. This new metric is ToPIC-Similarity, a newly proposed method to identify well separated topics and so propose the most suitable numbers of clusters to describe a corpus. In so doing, ToPIC reduces the complexity of the analysis, stores and proposes to the analyst only the more relevant results, helping her in the analysis task.
Figure 4.2 shows the ToPIC architecture and its building blocks, described in detail in the following Sections.

4.3 Preprocessing and weighting

In text based information retrieval, before analysing the documents in the corpus, the texts have to be prepared, by means of parsification and preprocessing. The parsing preprocessing phases include all the steps and techniques used in PASTA and described in Section 2.1.1. Once the documents have been split and tokenized, all the stopwords have to removed and all the terms have to be stemmed and normalized. A further step, newly included in the preprocessing phase of the framework,

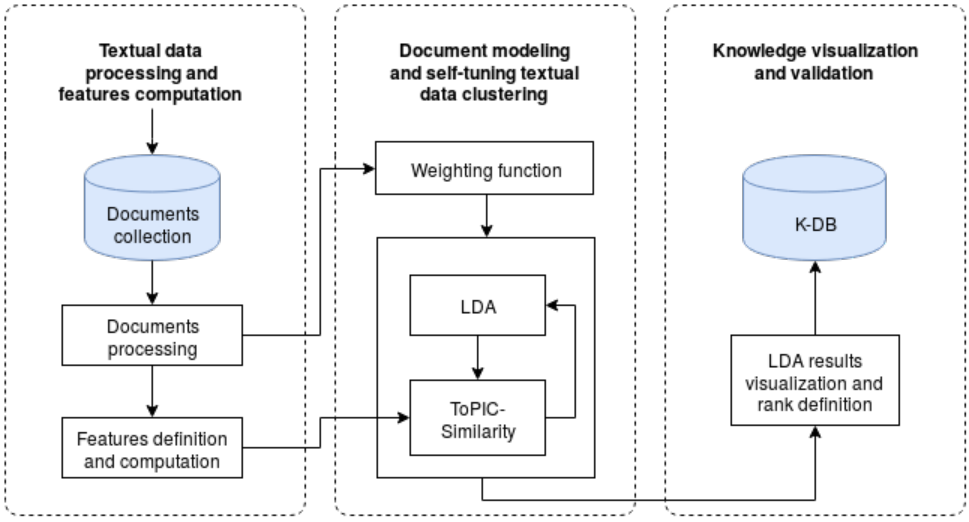


Figure 4.2: ToPIC architecture.

is the hapax removal. Since LDA draws words from the vocabulary taking into account the terms probabilities for each given topic of the documents mixture, excluding the words that appear only once in the corpus not only does not affect the resulting model, but instead will reduce the noise in the vocabulary and will improve the overall probabilistic modeling. In probabilistic topic modeling hapax removal is very effective, indeed being performed also in [3].
After the preprocessing of the dataset, the features described in Section 2.1.1 characterizing the dataset are computed. Then, the weighting phase can start. Weighting the terms in documents emphasizes the more relevant words, accordingly to the used weighting schema. This leads to different LDA models, reflecting the importance given to the words. Five combinations of different weights have been used in this study, namely:

- TF-IDF
- TF-Entropy
- LogTF-IDF
- LogTF-Entropy
- Boolean-TF_{glob}

Some of these weighting schemas are the ones that have also been used in the PASTA engine. As pointed out in [4], different weighting schemas are expected to lead to different results. Based on the document statistics and the desired granularity of the outcomes, one of the weighting schema is expected to outperform with respect to the others.

Weigth	Definition
Local	$\text{TF} = tf_{ij}$ $\text{LogTF} = \log_2(\text{tf}_{ij} + 1)$ $\text{Boolean} = \begin{cases} 1, & if\ tf_{ij} > 0 \\ 0, & if\ tf_{ij} = 0 \end{cases}$
Global	$\text{IDF} = \log\left(\frac{\lvert D\rvert}{df_i}\right)$ $Entropy = 1 + \frac{\sum_j p(i,j) log\ p(i,j)}{log(M)}$ $\text{TF}_{glob} = tf_i$

Table 4.1: ToPIC weighting functions.

Table 4.1 shows the definitions of the local and global weights used in ToPIC, where i is the ith term, j is the jth document. Observable patterns related to the different weighting schemas will be discussed in Chapter 5.

4.4 Applying LDA

Given the document-term matrix of weights X the LDA model is computed for a specific number of topics K, in order to perform the inference of the posterior distribution of the latent variables for the given corpus.
The obtained probabilistic model will be evaluated by means of the ToPIC-Similarity strategy and then, based on the results of the current and previous models' evaluations, the analytic flow can be fed back into the LDA algorithm to produce a new model given a new algorithm configuration. This process is repeated until a good trade-off between the quality of the obtained results and the execution time is achieved.
The used implementation of the Latent Dirichlet Allocation algorithm is the one available in the `org.apache.spark.ml.clustering` library [1] [16]. The considered LDA model implementation has been built specifically for text documents, and it needs several parameters in order to be created. The parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are the ones have to be set in order to maximize the log-likelihood of the data; the number of topics (and so the number of cluster) has to be defined by the parameter K. A detailed discussion of these parameters is given below and in the following Sections of the Chapter.

[1] Online documentation available at: `https://spark.apache.org/docs/2.2.0/api/scala/index.html#org.apache.spark.ml.clustering.LDA`

4.4.1 Setting LDA parameters

Parameter $\boldsymbol{\alpha}$

The $\boldsymbol{\alpha}$ parameter represents the concentration for the prior placed on documents' distributions over topics ($\boldsymbol{\theta}$). This means that low $\boldsymbol{\alpha}$ values will create documents that likely contain a mixture of only few topics, while high values will place more weight on having documents composed of many dominant topics. In the *Spark ML* library implementation of LDA, if this parameter is not specified the default value is set automatically; if a single value instead of a vector of values is set, then the single value is replicated to build an array of length K. The $\boldsymbol{\alpha}$ value used in this study is reported in Section 5.2.

Parameter $\boldsymbol{\beta}$

The $\boldsymbol{\beta}$ parameter describes the concentration for the prior placed on topics' distributions over terms. This means that low $\boldsymbol{\beta}$ values will likely produce topics that are well described just by few words, while high values will create topics composed by a mixture of most of the words (and so not any word specifically). Also, in this case, the *Spark ML* library implementation of LDA sets automatically the value of $\boldsymbol{\beta}$ if not set from the user. The $\boldsymbol{\beta}$ value used in this study is reported in Section 5.2.

Parameter $\boldsymbol{K}$

The number of topic K, besides being a parameter for the LDA, is one of the main targets of this research. Indeed, the more accurate the number of topics given to the model, the better are the clustering results given by the LDA. In literature, different solutions in order to find the most suitable K have been explored and proposed, as described in Section 2.3.
In this study we will propose a new approach to identify the most suitable K values, taking into account the level of semantic similarity of the topics inferred by the LDA algorithm.
As reported in [23], the research has not yielded to easy way to choose proper values for K beyond a major iterative approach. The followed approach is still iterative, as all the approaches known so far in literature: this means that in the framework, several LDA models with different values for the K parameter will be created.
Though, the goal of the research is to find the optimal K values evaluating not only probabilistic quality metrics, base the evaluation of the models on the topics content. A trade-off between the computational costs and the goodness of the results will be considered, to make the index efficient and effectively usable, even when applied to large data volumes. The newly proposed approach, called ToPIC-Similarity is described in detail in Section 4.5.

4.5 Finding optimal number of clusters

In order to find the most suitable number of topics to model a given corpus, in ToPIC we used a novel proposed approach, called ToPIC-Similarity. This strategy assesses how topics are semantically diverse, and then chooses proper configurations for the LDA modeling process.
Unlike the previously presented approaches to determine the most proper number of topics (and clusters) K, the approach we propose in this Section is not based on the internal LDA perplexity parameter or on probabilistic quality metrics, but evaluates the topics based on their terms representation. Since $\boldsymbol{\phi}$ models the topic-terms distributions, a description of the topics based on their content is available. Having the topics content and representation, it is possible to analyse how similar they are, and so choose K in order to maximize the difference among them.
Given a lower and an upper bound number of topics set by the analyst (i.e, $[K_{min}$, $K_{max}]$) a new LDA model is generated for each K. For each of these partitioning processes, ToPIC-Similarity requires three steps to be gone through:

1. *topic characterization*, to describe each t topic ($0 \leq t < K$) with the most n representative words;

2. *similarity computation*, to assess how the topics in the same partitioning are similar;

3. *K identification*, to find the optimal clustering configurations to be proposed to the analyst;

Steps 1) and 2) are repeated for all the t topics in every K clustering model.
The enumerated steps will be discussed in detail in the following parts of the Section, while a pseudocode of the implemented algorithm is reported in Algorithm 1.

4.5.1 Topic characterization

Among the results obtained by the LDA, the inferred topic-terms distribution $\boldsymbol{\phi}$ is used by ToPIC to determine topic similarities. Sorting $\boldsymbol{\phi}$ by decreasing probabilities, a subset of the most representative words of the topics can be retrieved. Topics are described by the probabilities the words of the corpus dictionary have to be drawn given the topic itself.
However, considering all the terms to compute the similarity is excessive because of the high computational cost that would be required, especially taking in consideration that many terms will have a very low and insignificant probability of being extracted. To reduce the costs and the computation, each topic is described by a subset of the terms in the dictionary, namely by its n most representative words. This number of words is automatically set depending on the corpus vocabulary $|V|$,

the average frequency of the terms corpus (*AvgFreq*), the Type-Token Ratio (TTR, which values belong to [0, 1]), and the currently considered number of topics K.
For each topic t, the number of terms n is obtained taking the corpus dictionary sorted by the distribution $\boldsymbol{\phi}$, considering only richest part of it (by means of the TTR index, that represents the lexical variation of the corpus) and then sampling the remaining words by the average frequency of the terms. The resulting number (named Q) represents the total number of terms considered, and it will have to be split over the K topics.
Defining $Q := \dfrac{|V| \cdot TTR}{AvgFreq}$, the number of terms considered to describe a topic t is mathematically described by:

$$n = \begin{cases} \dfrac{|V| \cdot TTR}{AvgFreq \cdot K}\,, & if\ Q \geq K \cdot AvgFreq \\ AvgFreq\,, & if\ Q < K \cdot AvgFreq \end{cases} \tag{4.4}$$

This is done if the total resulting number of words if greater than the average frequency of terms times the considered number of topics, to make the sampling reasonable (it does not make sense to sample with a period greater than the number of items themselves) and to have every topic represented at least by a number of words equal to the terms average frequency. If the condition is not satisfied, then a minimum number of terms is considered. We set this lower bound quantity to be equal to the average frequency of the terms in the corpus. At the end, the total number of words considered for the ToPIC-Similarity index are approximately $K \cdot n$.
Repetitions among the considered terms characterizing each topic are removed and the resulting words are considered together to make the topic representations comparable. Then, for each term in every topic, if the word is present in the topic description, the correspondent value is set to the probability that the term has to be picked up in the topic, or to 0 if it is not.

4.5.2 Similarity computation

To compute the ToPIC-Similarity index of the whole clustering model, all the possible pairs of topics are considered. The similarity among the topics is computed by means of the cosine similarity. The cosine similarity is often used in information retrieval and data mining, especially in text analysis and topic modeling; it is chosen among the other similarity or divergence indices because of its efficiency [17] and its ability to reflect the human perception [19]. Since we use this similarity index in the text analysis context and the probabilities of the terms are always positive, the cosine similarity values obtained will always be in the interval from 0 to 1. It is derived from the Euclidean dot product and, given two topics t' and t" of the

Algorithm 1: ToPIC characterization algorithm

```
Data: X, K_min, K_max
Result: kSol
// variable inizialization
topicS = [ ],   NTerms = [ ] ;
for K ← K_min to K_max do
    // build the LDA model;
    LDAModel ← lda.fit(X);
    Q ← (|V| · TTR)/AvgFreq;
    // set the number of terms per topic;
    if Q ≥ K · AvgFreq then
        n ← Q/K);
    else
        n ← AvgFreq;
    end
    // collect together the terms of each topic;
    for t ← 0 to (K-1) do
        NTerms.append(LDAModel.describeTopics()[t].sort().take(n));
    end
    N ← NTerms.size() ;
    topicsDescr = zeros(K, N) ;
    simMatrix = zeros(K, K) ;
    for t ← 0 to (K-1) do
        for word ← 0 to N do
            // take the probability that the term has to be drawn
            // from the topic, given the LDAModel
            topicsDescr[t][word] ← LDAModel.describeTopics()[t,
              NTerms[word]];
        end
    end
    for t ← 0 to (K-1) do
        for s ← 0 to (K-1) do
            simMatrix[t][s] ← cosine(topicsDescr[t], topicsDescr[s]);
        end
    end
    topicS.append(norm(simMatrix)*100/K);
    if topicS[K] ≥ topicS[K-1] && secondDerivative(topicS[K-1]) > 0 then
        kSol.append(topicS[K-1]);
        if kSol.size() > 3 then
            return kSol.take(3);
        end
    end
end
```

same partitioning K, it is computed as follows: $similarity(t', t'') = \frac{\mathbf{N}_{t'} \cdot \mathbf{N}_{t''}}{\|\mathbf{N}_{t'}\|_2 \cdot \|\mathbf{N}_{t''}\|_2}$, where $\mathbf{N}_{t'}$ and $\mathbf{N}_{t''}$ are the set of representative words of topic t' and t", respectively. The result is a $K \times K$ symmetric matrix where each cell (i,j) is the similarity between topic in row i and topic in column j.
The ToPIC-Similarity of the clustering is obtained averaging the similarity matrix over K to keep in consideration the different number of clusters. For this issue, the norm of the whole similarity matrix (using the Frobenius norm) is computed and then the values is divided by K. To get the final ToPIC-Similarity this result is then expressed in percentage by multiplying it by 100.

4.5.3 *K* identification

Computing the ToPIC-Similarity values for the different LDA models generated for different K values, a topic similarity function is obtained.
To find optimal K values a trade-off approach has been chosen. It has been empirically seen that the obtained ToPIC-Similarity function is, in most of the cases, decreasing but not monotonic. This property allows us to choose as K values the local minima of the curve, namely the K for which ToPIC-Similarity(K_i) $<$ ToPIC-Similarity(K_{i+1}). A further condition is added in order to choose the relevant points: to consider only K values that belong to a decreasing segment of the curve, the second derivative is computed and only the points that have a positive second derivative are considered. In our study, we considered a trade-off between optimal results and computational cost: the selected values are the firsts three points that satisfy both the condition stated before. The topic modeling and the research for optimal K values can stop when the first three values are found, or when the algorithm reaches the K upper bound value set by the analyst (and in this case a lower number of optimal values will be proposed to the analyst).
This method is newly proposed, since at our knowledge considering the cosine similarity of the semantic description of the topics has been never considered in the literature to automatically infer the most suitable number of clusters K for the LDA topic modeling process.

4.6 Visualize and validate the LDA results

Evaluating data models using unlabelled data is hard. Many quantitative theoretic indices can be used to assess the quality of a clustering process and so identify the best partitioning. However, it is good practice to check if the obtained models actually make sense. To evaluate the goodness of the LDA modeling, and confirm the ranking obtained with the quality metrics, visualization methods can be used. This is possible because the LDA results allow to directly visualize the clustering inferred

in the learning process, both in terms of documents grouping and in terms of topics characterization.
Evaluating high dimensional datasets and models is hard, so several strategies to visualize different aspects of the data are needed, such as indices aggregating the many dimensions of the data items. Two quantitative validation approaches (perplexity, described in 4.6.1, and clustering metrics, described in 4.6.2) and two visualization techniques (t-SNE, described in 4.6.3, and topics-terms representations, described in 4.6.4) have been used to evaluate the obtained results.

4.6.1 Perplexity

Perplexity is a measure of the quality of probabilistic models, that describes how well a model predicts a sample (i.e. how much it is perplexed by a sample from the observed data). It is widely used in text and language modeling, and the authors of the LDA model used it to evaluate and compare the results of the LDA topics inference.
Perplexity decreases monotonically in the likelihood of the data, and it is equivalent to the inverse of the per-word likelihood. In [3], perplexity is defined as:

$$perplexity(\mathcal{D}) = exp\left\{\frac{-\sum_m logp(\mathbf{w}_m)}{\sum_m |\mathbf{w}_m|}\right\} \tag{4.5}$$

Given a computed model, the lower is its overall perplexity, the better are the performances of the model and the estimation of the corpus probabilities.

4.6.2 Clustering metrics

Silhouette

Silhouette [14] is one of the most used and well-known metrics to measure the consistency and the cohesion of a clustering. It can assume values in the interval [-1, 1], and since it describes how well the clustering performs for the given dataset, it is often used as a validation index.
Silhouette measures how an object is similar to its own cluster (cohesion), and with respect to the others (separation). High Silhouette values indicate that the clustering process well performed dividing the data collection, whether low values suggests that the results are not well describing the structure of the dataset. The Silhouette index can be computed with any distance metrics, such as the Euclidean or the Manhattan.
Given $a(i)$ the mean intra-cluster distance (namely the average distance of the data point i from all the other points assigned to the same cluster) and $b(i)$ the mean

nearest-cluster distance (the average distance with all the points in the nearest cluster, of which i is not a member), Silhouette is defined as: $s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}$.
The average $s(i)$ computed over all the data points of a cluster represents how thigh the considered cluster is, and how well its points are separated from the other ones. In our context, the Silhouette index is computed using the most probable topic of a document as its clustering label, and all document-topics probabilities as coordinates of the points themselves.

Entropy

In information theory, *entropy* [13] is defined as the amount of information contained in an event. Generally, the more the uncertainty in an event, the more information it will contain. This means that information decreases in uncertainty, or entropy. By this definition, entropy can be seen as an average ambiguity of a probabilistic event: the larger the entropy, the more uncertainty and ambiguous the event is.
Applied to the modeling context, entropy measures how uncertain the model is: the lower the entropy of the model, the more certain the model is describing the data collection under analysis.
Specifically, concerning our context and given an LDA model, for each document d_m in the corpus we computed the entropy it has to belong to one of the K topics is computed as follows:

$$H(d_m) = -\sum_{t=1}^{K} p(d_m = t) log(d_m = t) \tag{4.6}$$

being $p(d_m = t)$ the probability the considered document has to be assigned to topic t. To compute the entropy of the whole clustering model, we averaged the entropy of each document over the entire corpus: $H(model) = \frac{\left(\sum_{m=1}^{M} H(d_m)\right)}{M}$.

4.6.3 t-SNE

Visualizing the document clustering is not a trivial task for high-dimensional data such as text documents. Indeed, just grouping the same-labeled documents is not sufficient to describe the clustering results; for this reason, in this study we use the *t-Distributed Stochastic Neighbor Embedding* (t-SNE), a technique able to visualize high-dimensional data over a two or three-dimensional space [12].
Normally, high-dimensional datasets are represented graphically reducing the dimensionality of the dataset, nevertheless trying to preserve the most significant structure of the data. This approach is usually achieved with reduction techniques such as PCA. However, especially when the high-dimensional dataset is made of similar data points, it is often difficult to use a linear mapping (as the one performed by

the Principal Component Analysis) to properly visualize the differences among the data items.
t-SNE is a non-linear algorithm for dimensionality reduction, able to capture much of the local structure of the high-dimensional data very well, while at the same time also revealing global structure such as the presence of clusters at several scales [12]. This feature allows similar data points to be represented nearby and, in the meanwhile, different data points to be represented far in the new low-dimensional space. The algorithm converts the Euclidean distances among the data points into conditional probabilities representing similarities.
$P_{i|j}$ are the probabilities that a certain point x_i would pick x_j as its neighbour, if neighbours were distributed with a Gaussian probability centered in x_i: this probability is high for nearby data points, while it is almost zero for very different data points.
The same probabilities are computed for the new low-dimensional space: $P_{i|j}$ are the probabilities that the low-dimensional datapoint y_i would choose y_j as its neighbour. If the two representations have the same dimensionality, then the two conditional probabilities have to be equal. For a different number of dimensionalities, t-SNE minimizes their difference through the *Kullback-Leibler divergence* [11].

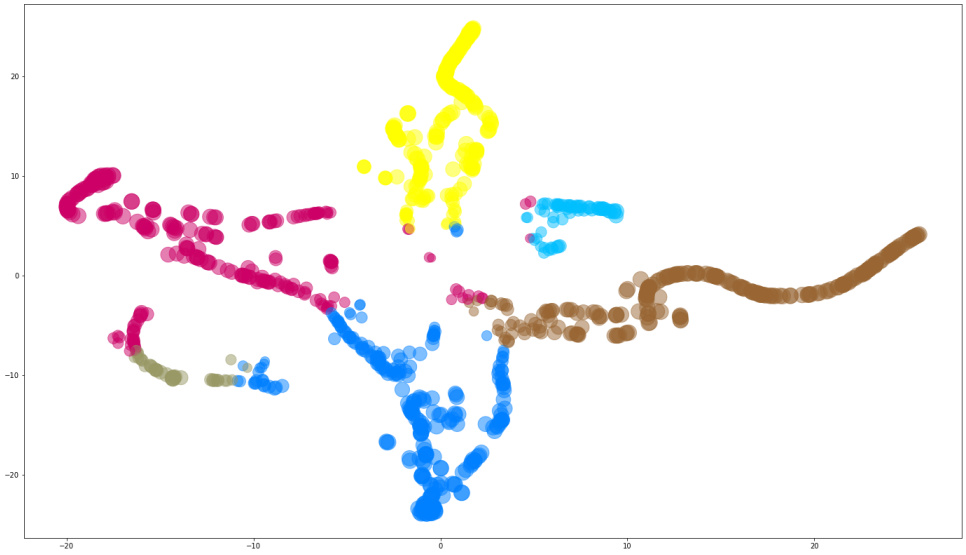


Figure 4.3: Example of the t-SNE representation, dataset D1.

The data reduced in dimensionality can then be easily printed, for example by means of a scatter plot, still being able to visualize the original structure and relations among the data points.
Figure 4.3 shows an example of a t-SNE dataset visualization. It is possible to see how data points are grouped together based on their characteristics and how the structure of the dataset is maintained by the algorithm. The coloring of the points is

based on the assignment to a specific topic, reflecting the results of the LDA model.

4.6.4 Topics-terms representations

Another way to evaluate the results of the LDA algorithm, is to take into account the representation of the topics, thus the content of the topics themselves.
This allows us to compare the terms across and within the topics found by LDA. Topics representation is obtained from the LDA model, by means of the method `LDAModel.describeTopics()`, that returns the topic description ϕ.

Termite

This representation has been proposed in literature by Chuang et al., with the aim to support effective evaluation of the terms distributions associated with LDA topics [5]. This representation enables the analyst to assess the quality of the clustering results, evaluating the goodness of each topic and of all the topics as a whole.
For each topic, a certain number of the most representative terms is taken: their belongingness to the topic is represented in the diagram as a point, which size depends on the probability that the terms has to be drawn from the topic when creating the document.
The topic-terms diagram helps the analyst to detect if a term is salient for a specific topic, or if it is equally probable that it would be drawn in the document creation process. This representation is also useful to spot stopwords left over by the preprocessing steps, or even to spot useless topics, that are not represented by any salient term.
In figure 4.4 an example of topic-terms representation is shown. The chart has been built for the dataset D1, considering few terms for the image visualization sake.

Word clouds

In order to visualize the content of the topics we used the word cloud technique. The word clouds represent the terms that, according to the obtained LDA model, most probably describe the topics.
The clouds represent the topic-terms matrices: the comparison of the cloud sets obtained by different models is left to the human judgment.
The word cloud visualization stresses the terms with the highest probabilities with a bigger font size. In this way is possible to straightforwardly observe if the classification results are good or the topic modeling has not yielded to acceptable outcomes. Because of its clearness and simplicity, the word cloud representation has been already used in literature to visualize the results of the LDA topic modeling [23].

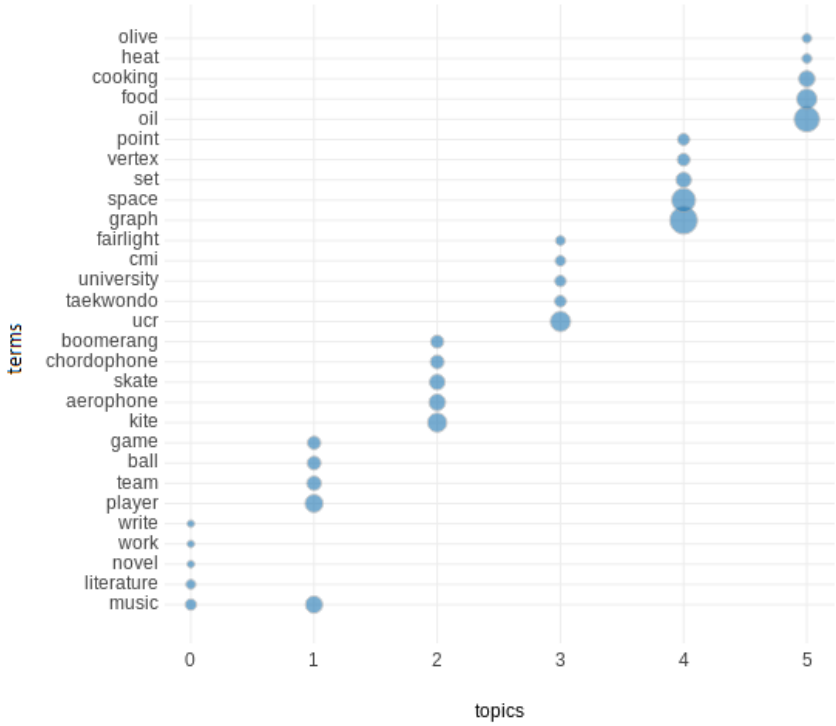


Figure 4.4: Example of the Termite topic-terms representation, dataset D1.

Word tables

Another way to represent the topics and their content is to use the word tables. This representation, the simplest among the topic-terms representations, basically lists the terms that describe the topics by decreasing order of probabilities.
As suggested in prior works [5], the quality of a topic is often determined by the coherence of its constituent words. The goal of this kind of visualization is, again, to assess how the clustering process performed, by considering the topics cohesiveness and coherence through their content.

Chapter 5

Experimental results

In this chapter the experimental results generated to test the effectiveness of ToPIC will be presented. The datasets used in this study and their descriptions are reported in Section 5.1. For every dataset, the set of its statistical indices is presented, along with a brief overview of the corpus. Section 5.2 briefly describes the settings and the running environment used for the study. Section 5.3 will show the results obtained for every dataset, together with the relative obtained quantitative evaluation indices and the comparison with the state-of-the-art outcomes. Section 5.4 consider a representative running example and shows in detail the ToPIC results, their evaluation process (by means of the methods reported in Chapters 4), and a specific comparison with the RPC and En-LDA approaches. The Section ends with an overview of the considerations done for the analysis of the representative dataset. Section 5.5 briefly reports the results obtaineed for the remaining datasets. The impact of the different weighting schemas proposed in ToPIC will be discussed in Section 5.6, in order to highlight and explain the differences obtained in the gathered results. Section 5.7 gives an overview of the comparison with the state-of-the-art techniques considering all the datasets used to test the framework. The Chapter ends with Section 5.8, giving a final sum-up and final considerations about the results and the effectiveness of the whole system.

5.1 Experiment datasets

The presented ToPIC framework has been tested over different datasets, belonging to different kind of sources and typologies. The corpora have been chosen in order to have different characteristics, from the number of documents to the length of the individual texts, from the lexical richness to the average frequency of the terms.
Documents in the same corpus should be characterized by homogeneous lengths and heterogeneous topics, beyond being produced by different authors. These characteristics allows the results to be comparable and generic, avoiding the over-fitting of

the datasets.

	Wikipedia		Reuters
	1000_5cat	**2500_10cat**	**21578**
dataset ID	D1	D2	D3

Table 5.1: Datasets IDs.

The datasets can be grouped based on their source and typology. Table 5.1 reports a summary of the used dataset, with the IDs they are assigned to for this study.
In the following part of the Section, the used datasets are described in detail. For every data collection, the indices characterizing the data distribution are reported, as computed before the hapax removal (**WH**) and after the hapax removal (**WoH**). Eliminating words that appear only once within the corpus (i.e., hapax) allows the LDA to construct a more precise probabilistic model. Since each term of the corpus is drawn from the vocabulary taking into account the terms probabilities for each given topic of the documents mixture, excluding the hapax terms will allow the reduction of noise in the vocabulary.
Moreover, the features computed with the hapax terms and without them and reported in the tables do not show significant differences. Indeed, the proportions among the number of words, the Guiraud index and the TTR do not change. This happens even removing nearly half the dictionary since, in some datasets, the hapax percentage reaches values higher than 50%.
Dataset D1 will be used as representative dataset: detailed considerations and conclusions, explanatory and crucial figures will be reported for this running example. Detailed results for all the other datasets used in the study will be given in Section 5.5.

5.1.1 Wikipedia

The first group of datasets used in the study is composed of two different collections of Wikipedia articles[1]. Wikipedia's contents are released under *Creative Commons* license, and so their usage is free and public. Wikipedia articles are written by different users, so unbiased by a certain writing style, their content is usually specific, and in average they have the same length. Categories were chosen to be sufficiently separated and so detectable by the clustering algorithms. From these categories two different datasets have been created, diverging for the number of documents taken for each topic.
In order to build the first dataset, 200 articles have been taken from the following five

[1] `https://en.wikipedia.org`

categories: cooking, literature, mathematics, music and sports. The following ten categories have been chosen instead to build the second article collection: astronomy, cooking, geography, history, literature, mathematics, music, politics, religion, sports. The second dataset collects the first 250 articles for each of the topics shown before. These datasets have been already used to test the PASTA engine.
Table 5.2 reports the statistical indices of the two Wikipedia datasets used in the study.

	Wikipedia			
	WH	**WoH**	**WH**	**WoH**
Dataset ID	D1		D2	
# documents	989		2,463	
Max frequency	6,343		9,405	
Min frequency	1	2	1	2
Avg frequency	16	32	21	44
# terms	1,109,408	1,075,210	2,995,762	2,882,781
Dictionary $\lvert V \rvert$	67,613	33,415	138,329	65,336
TTR	0.06	0.03	0.05	0.02
Hapax %	50.58	0	52.77	0
Guiraud Index	64.19	32.23	80.46	38.48

Table 5.2: Statistical indices for the Wikipedia datasets.

5.1.2 Reuters

The Reuters dataset, publicly available and known as *Reuters-21578*[2], is widely used test collection originally created in 1987 by the Carnegie Group, Inc. and Reuters, Ltd for text categorization research purposes, and then made available for research purposes only. This dataset is often used for information retrieval, machine learning and corpus-based researches.
The original dataset is made of 21578 articles but for this study, as did in order to test the PASTA engine, only a subset of documents has been taken into account. This subset has been created from the *Apte' Split 90 categories*[3], a formatted version of Reuters-21578, that divides the dataset in different categories and then again divides them in the testing and training partitions. The subset used for this study is the whole *Apte' Split 90 categories*, created merging together the test and the training

[2] `http://www.daviddlewis.com/resources/testcollections/reuters21578`

[3] Dataset on-line available at `http://disi.unitn.it/moschitti/corpora.htm`

part, totaling to 15.437 documents.
Table 5.3 reports the statistical indices of the Reuters datasets used in this study.

	Reuters	
	WH	**WoH**
Dataset ID	D4	
# documents	15,437	
Max frequency	42,886	
Min frequency	1	2
Avg frequency	55	76
# terms	1,337,225	1,316,988
Dictionary $\|V\|$	24,239	17,153
TTR	0.02	0.01
Hapax %	29.2	0
Guiraud Index	20.96	14,95

Table 5.3: Statistical indices for the Reuters dataset.

5.2 Experimental settings

For this study, the following settings for the LDA models have been used: the maximum number of iterations within the model has to converge has been set to be equal to 100, the Optimizer (or inference algorithm used to estimate the LDA model) has been set to be Online Variational Bayes (default).
The possible range of K values, $[K_{min}, K_{max}]$, has been set to [2,20]. Indeed, 2 is the lowest possible number of clusters to divide the corpus in, and 20 has been considered an acceptable upper bound, since having more clusters will lead to results difficult to be interpreted and validated.
Since we used `Online` optimizer, the $\boldsymbol{\alpha}$ value and the $\boldsymbol{\beta}$ value should be greater than or equal to 0. For this study, the value set for this parameter is $\boldsymbol{\alpha} = 50/K$, as proposed in the literature by several articles [9, 15, 20], and the value set for $\boldsymbol{\beta}$ is $\boldsymbol{\beta} = 0.1$, as proposed in the literature by Griffiths et al. [9].
The ToPIC framework has been developed to be distributed and so implemented in Spark. Since then, all experiments have been performed on the BigData@PoliTO cluster[4] running Apache Spark 2.0.0. Three virtual nodes, two executors and a driver, having a 7GB main memory and a quadcore processor have been deployed for this research.

[4]https://bigdata.polito.it/content/bigdata-cluster

5.3 ToPIC results and effectiveness

Table 5.4 shows the results obtained using ToPIC; for each dataset and weighting schema, it includes a row for each K obtained using our proposed clustering methodology and the three proposed metrics used to analyse the goodness of the statistical model generated.
To be able to present the differences of the results between our approach and the state-of-the-art ones, Table 5.5 reports the results obtained with the RPC and the En-LDA techniques, together with their evaluation, using the same metrics of table 5.4.
A general discussion about the ToPIC performance and the comparison with the other techniques will be given in Section 5.8 and Section 5.7.

5.4 A running example: results and evaluation

This Section shows in detail the results and the evaluation process and techniques for the representative dataset, i.e. D1. This dataset, namely the *Wikipedia 1000-5cat*, contains about 1000 documents related to cooking, literature, mathematics, music and sports themes.
The dataset will be evaluated for all the weighting schemas considered in ToPIC, assessing the goodness of the found results and then comparing the models obtained with the proposed framework and with the state-of-the-art techniques. The final part of the Section will sum up the considerations about the effectiveness of ToPIC, the different impacts of the weighting schemas, the differences and similarities with the state-of-the-art results.

5.4.1 Dataset D1, TF-IDF results

The results obtained by ToPIC with the TF-IDF weighting schema are reported in figure 5.1b. From the chart, the selected K values are 3, 6 and 10.
From table 5.4 we can state that the Silhouette and the entropy indices agree on the evaluation of the selected models. On the contrary, the perplexity index assigns to the higher values of K a lower uncertainty of the models.
As discussed before, quantitative metrics are not enough to evaluate the overall quality of the clustering process. So, in order to correctly examine the results, the t-SNE and the topic-terms representations of the considered models are reported from figure 5.2 to figure 5.5.
From the t-SNE representation it is possible to see that the clusters are all almost balanced in terms of document cardinalities, and the number of the main clusters is actually reflecting the actual structure of the dataset, originally composed of five categories.

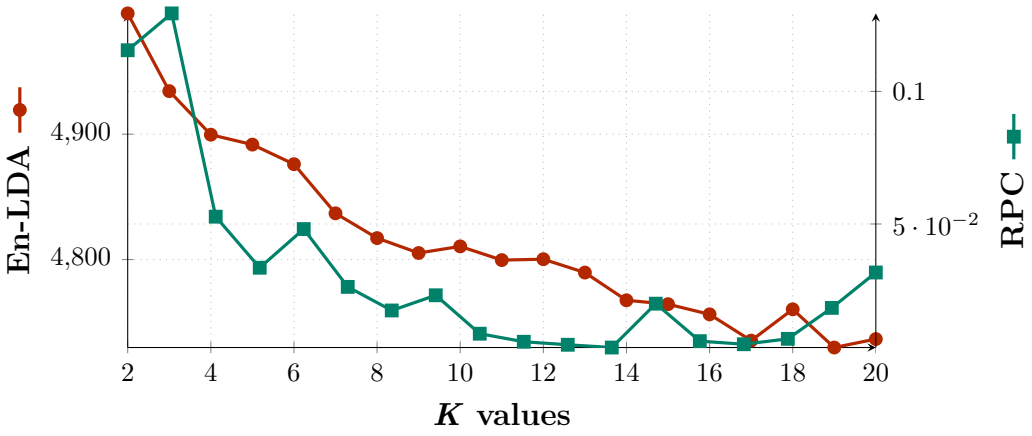
	Weight	K	Perpl	Silh	Entr	Execution Time
D1	**TF-IDF**	3	8.8127	0.7721	0.2561	1h 50min
		6	8.5970	0.6935	0.3634	
		10	**8.4822**	**0.6827**	**0.3956**	
	TF-Entr	**5**	**9.0724**	**0.7623**	**0.2825**	1h 30min
		8	9.2482	0.6324	0.3388	
		9	9.2679	0.6319	0.3395	
	LogTF-IDF	**8**	**9.1873**	**0.6754**	**0.3205**	1h 40min
		17	9.1262	0.6370	0.3626	
	LogTF-Entr	5	9.9126	0.8915	0.1004	1h 30min
		7	**9.8841**	**0.8460**	**0.1748**	
		11	9.9794	0.9515	0.1089	
	Boolean-TF	4	6.4926	0.6979	0.4214	2h 5min
		5	**6.4640**	**0.6618**	**0.4832**	
		17	6.4208	0.3813	1.0901	
D2	**TF-IDF**	3	9.2008	0.7715	0.2460	3h 20min
		8	8.9628	0.5878	0.5314	
		10	**8.9436**	**0.5530**	**0.6118**	
	TF-Entr	3	9.5568	0.8075	0.2161	3h 25min
		7	**9.4555**	**0.7008**	**0.3556**	
		8	9.4631	0.6985	0.3693	
	LogTF-IDF	**11**	**9.4108**	**0.6016**	**0.4895**	3h 20min
		14	9.4529	0.5652	0.4958	
	LogTF-Entr	**7**	**10.2031**	**0.8751**	**0.1258**	3h 10min
		9	10.2194	0.8922	0.1219	
		11	10.2327	0.9012	0.1253	
	Boolean-TF	6	6.6223	0.4398	0.7979	5h 20min
		13	6.5833	0.3380	1.1922	
		18	**6.5699**	**0.3205**	**1.3262**	
D3	**TF-IDF**	3	7.7154	0.7347	0.2763	1h 55min
		4	7.6455	0.6913	0.3485	
		9	**7.4389**	**0.5966**	**0.5586**	
	TF-Entr	4	8.5396	0.0564	1.3806	1h 50min
		6	8.6242	-0.247	1.7805	
		9	**8.7109**	**-0.0811**	**2.169**	
	LogTF-IDF	5	7.7503	0.7005	0.3565	1h 50min
		7	7.6686	0.6676	0.4440	
		13	**7.5614**	**0.5989**	**0.6396**	
	LogTF-Entr	**5**	**8.7880**	**0.0774**	**1.6090**	1h 50min
		9	9.0011	-0.0437	2.1955	
		13	9.1759	-0.0690	2.5600	
	Boolean-TF	4	3.9669	0.6373	0.4659	2h 20min
		7	3.8947	0.4730	0.7352	
		16	**3.7309**	**0.3016**	**1.3112**	

Table 5.4: ToPIC results table.

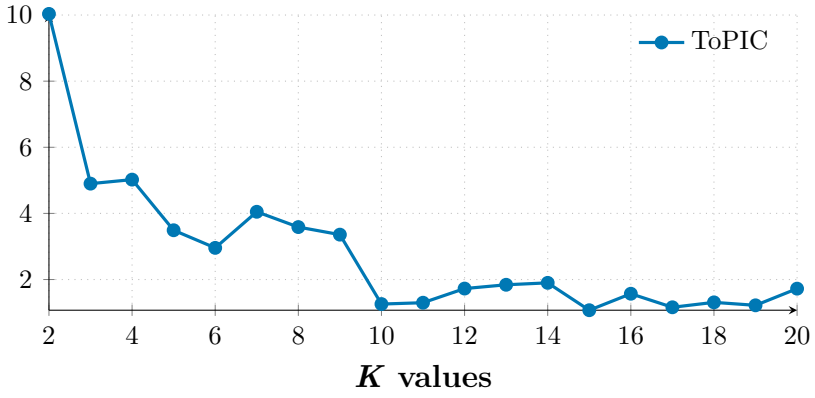
	Weights	Method	K	Perpl	Silh	Entropy
D1	**TF-IDF**	RPC	3	8.8127	0.7721	0.2561
		En-LDA	19	8.4273	0.6211	0.5345
	TF-Entropy	RPC	5	9.0724	0.7623	0.2825
		En-LDA	5	9.0724	0.7623	0.2825
	LogTF-IDF	RPC	7	9.1837	0.6935	0.3192
		En-LDA	16	9.1890	0.5530	0.4434
	LogTF-Entropy	RPC	3	9.7774	0.8524	0.1441
		En-LDA	3	9.7774	0.8524	0.1441
	Boolean-TF$_{glob}$	RPC	4	6.4926	0.6979	0.4214
		En-LDA	20	6.4127	0.6618	1.2558
D2	**TF-IDF**	RPC	6	9.0370	0.6214	0.4638
		En-LDA	20	8.8447	0.5274	0.6617
	TF-Entropy	RPC	5	9.4955	0.7198	0.3293
		En-LDA	20	9.5718	0.6060	0.4374
	LogTF-IDF	RPC	7	9.4280	0.6213	0.4601
		En-LDA	19	9.4052	0.5095	0.5345
	LogTF-Entropy	RPC	3	10.1156	0.7787	0.2142
		En-LDA	5	10.1365	0.8172	0.1762
	Boolean-TF$_{glob}$	RPC	7	6.6159	0.3956	0.8756
		En-LDA	20	6.5632	0.3153	1.3923
D3	**TF-IDF**	RPC	5	7.5937	0.6728	0.3924
		En-LDA	20	7.3124	0.5366	0.7986
	TF-Entropy	RPC	7	8.6670	-0.586	1.9340
		En-LDA	17	8.7483	-0.0428	2.6406
	LogTF-IDF	RPC	4	7.8130	0.7350	0.3088
		En-LDA	19	7.5663	0.5688	0.7740
	LogTF-Entropy	RPC	13	9.1759	-0.0690	2.5600
		En-LDA	17	9.2942	-1847	2.8132
	Boolean-TF$_{glob}$	RPC	6	3.9200	0.5007	0.6562
		En-LDA	20	3.6822	0.2954	1.4030

Table 5.5: State-of-the-art results table.

The topic-terms technique used in this case, for interpretability and visualization reasons, is the word cloud. From this representation it is possible to assess the cohesion of the clusters and assign a general label to each topic.
Looking at the case where K has been selected to be equal to 3 (Figure 5.3), it is possible to identify the following topics: *literature* (or more generally *art*), *sport* and *cooking/mathematics*. It is clear that some original topics are mixed, and a greater number of clusters may be needed to better represent the dataset.
With 6 clusters, LDA well divides the documents belonging to the following topics: *literature/art*, *sport*, *mathematics*, *cooking* (Figure 5.4). A further cluster is described by sports and music terms, specifically related to boards, air, water and wheels. Another one collects words about fighting sports and the geographical East area. The *music* topic appears to be split over different clusters, such as the art and



(a) Dataset D1, TF-IDF, En-LDA and RPC results.



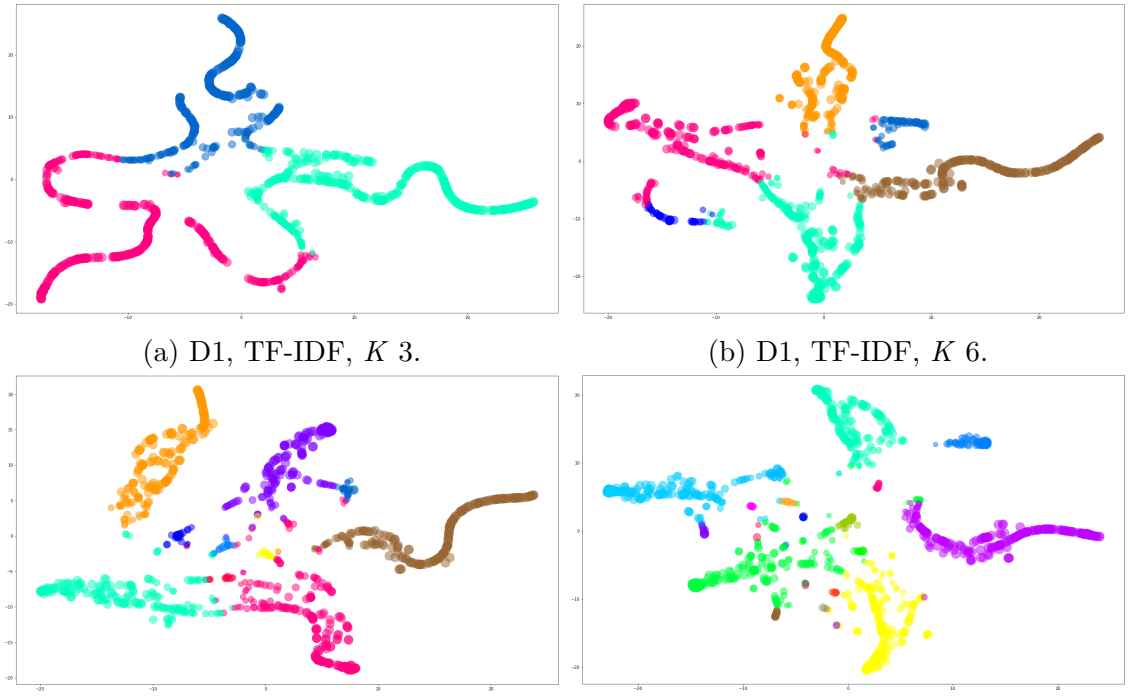
(b) Dataset D1, TF-IDF, ToPIC results.

Figure 5.1: En-LDA, RPC and ToPIC results diagrams for dataset D1, TF-IDF weighting schema.

the sport ones.

To visualize the results obtained for K equal to 10, only a subset of most representative topic descriptions has been reported. As it is possible to see from the relative t-SNE representation (Figure 5.2c) and the word clouds (Figure 5.5), the five bigger clusters well represent and describe the main categories of the dataset (i.e. *literature*, *music*, *mathematics*, *cooking* and *sports*). By the word descriptions of the remaining clusters, it is possible to see they represent sub-topics. The content is related to specific sports, rather than specific musical instruments.

The more detailed analysis of the models generated by the LDA modeling process, leads us to consider the solution obtained with K equal to 10 as the better one.



(a) D1, TF-IDF, K 3.

(b) D1, TF-IDF, K 6.

(c) D1, TF-IDF, K 10.

(d) D1, TF-IDF, K 19.

Figure 5.2: D1 t-SNE representation, TF-IDF weighting schema, K 3, 6, 10 and 19 respectively.

Figure 5.3: D1 WordCloud representation, TF-IDF weighting schema, K 3.

This is reported in Table 5.4 highlighting the solution in bold.

Comparing the ToPIC results with the state-of-the-art techniques, which produce as K values 3 and 19 (with RPC and En-LDA respectively), two different scenarios are depicted.
The RPC proposes 3 as optimal number of clusters. This is the same value proposed by the first solution of the ToPIC framework. As described above, the clustering result is not bad, but some of the original topics are mixed together (*music* and

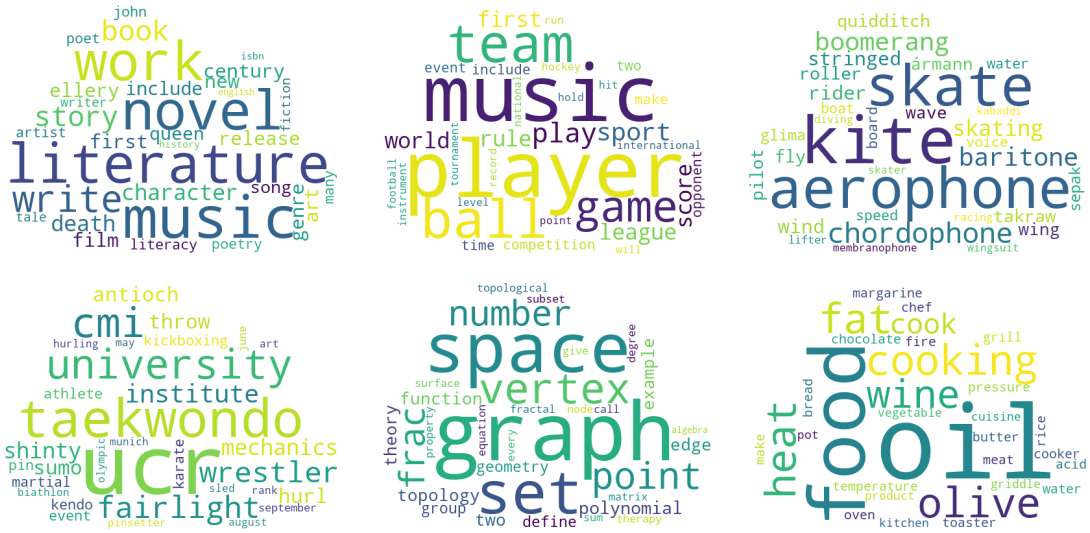


Figure 5.4: D1 WordCloud representation, TF-IDF weighting schema, K 6.

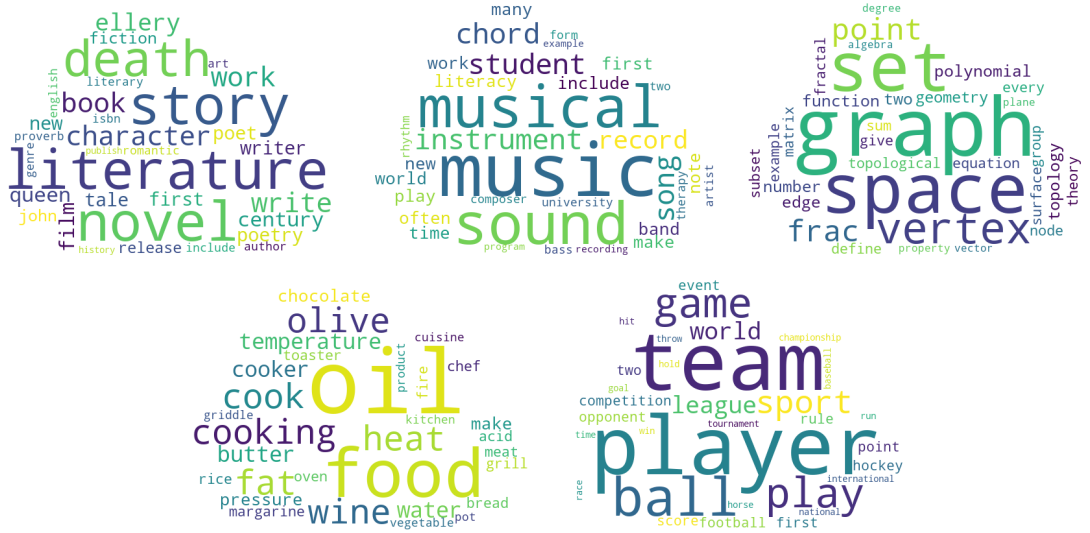


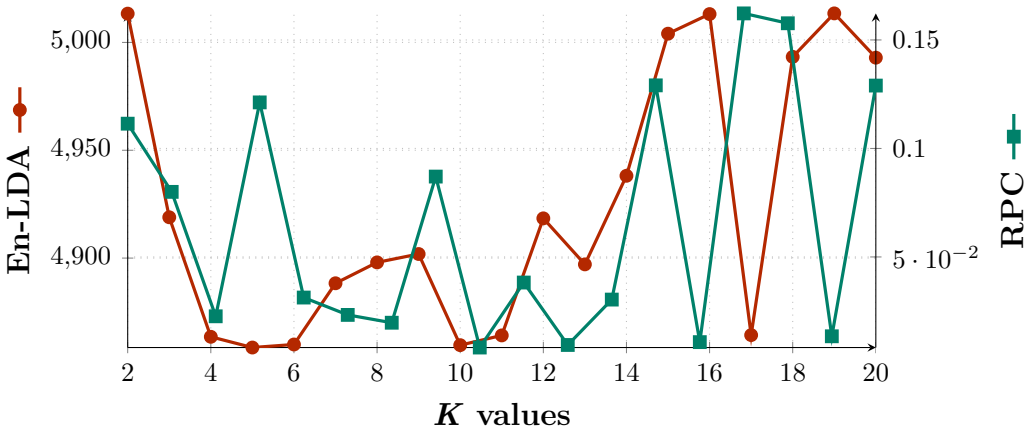
Figure 5.5: D1 WordCloud representation, TF-IDF weighting schema, K 10.

literature, *sports* and *mathematics*). In this sense, ToPIC outperforms RPC giving more options to the analyst, letting her the possibility to choose among different solutions with different granularity levels.
With the En-LDA approach, which proposes 19 as the optimal number of clusters, good partitions are identified (the t-SNE representation of the clustering result is reported in Figure 5.2d). Indeed, all the original categories of dataset can be recovered in topics. Furthermore, the model identifies very specific topics, that describe only

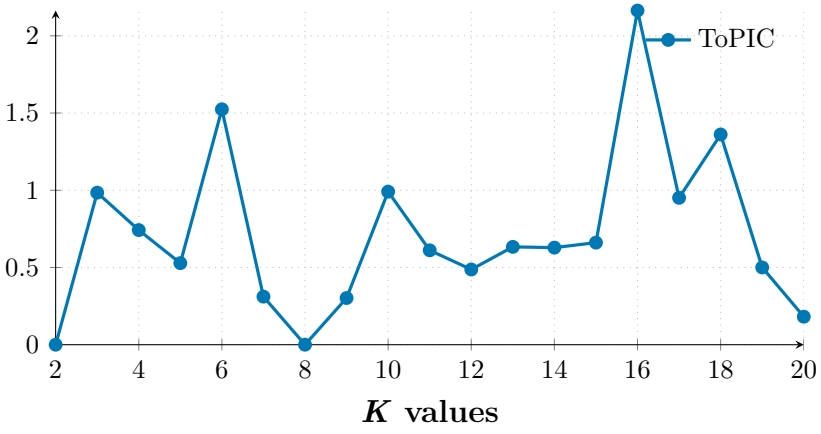
few documents, and often divide the main categories in subtopics, that deal about more specific arguments with respect to main ones. Examples are the *opera* and the *instruments* topics, that both belong to the *music* main category. The modeling is overall good, but having more topics that the ones actually required not necessarily means having a better result. Indeed, too many topics may not be useful for the analysis since then the analysts have a more complex result set to consider in their work.

5.4.2 Dataset D1, TF-Entropy

The results obtained by ToPIC with the TF-Entropy weighting schema are reported in figure 5.6b. From the chart, the selected K values are 5, 8 and 9.
Computing the perplexity, Silhouette and the entropy metrics to evaluate the proposed results, we obtain concordant values for Silhouette and entropy and an opposite trend for the perplexity values. Evaluating graphical results and going deeper in the topic analysis could lead to more precise evaluation of the clustering outcomes. The t-SNE and the topic-terms representations of the LDA results obtained with the TF-Entropy weighting schema are reported from Figure 5.7 to 5.8.
The t-SNE charts show very different results of the clustering processes. The topics modeled with K equal to 5 seems to be balanced in the number of documents assigned to each one. On the contrary, the results obtained with 8 and 9 clusters look similar, having moreover strongly unbalanced cardinalities for different topics. Looking at the topic-terms visualization will help the analysis process.
The Termite representation of the clustering obtained with K equal to 5 is reported in Figure 5.8. The represented topics are well described, and they identify the original categories of the dataset. Indeed, among the topics we can clearly assign the following labels: *sports* (mixed with *food*), *literature*, *music*, *mathematics*. To the last cluster, even if it does not identify a specific topic, are assigned only few documents (15), so it is not a relevant cluster for the analysis.
Being the results obtained with K equal to 8 and 9 very similar not only in terms of t-SNE representation, but also in terms of topic-terms representation, only the results with 8 clusters will be discussed below. Good topics descriptions can be identified in the results, such as the ones dealing about *food*, *music*/*sports*, *literature*. However, most of the topics descriptions contains words that are difficult to be linked to a unique semantic area. Indeed, they mix together words of mathematics, food or sports, but especially they include words that are not relevant to any theme. This happens because of used the weighting schema: the local weight LogTF flattens the relevance of the terms in the documents. Thus, finding the keywords in the texts becomes more complicate and the inference of the LDA models worsens.
Based on these considerations and the analysed results, the more optimal solution results to be the one obtained with 5 clusters. Indeed, the solutions with greater



(a) Dataset D1, TF-Entropy, En-LDA and RPC results.

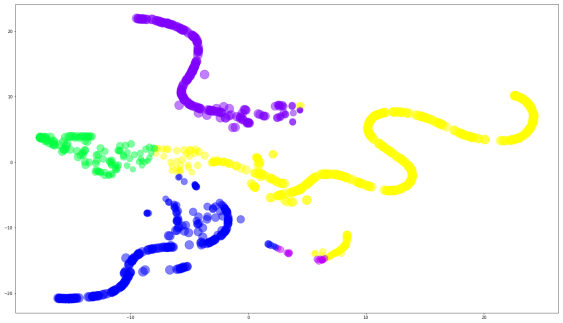


(b) Dataset D1, TF-Entropy, ToPIC results.

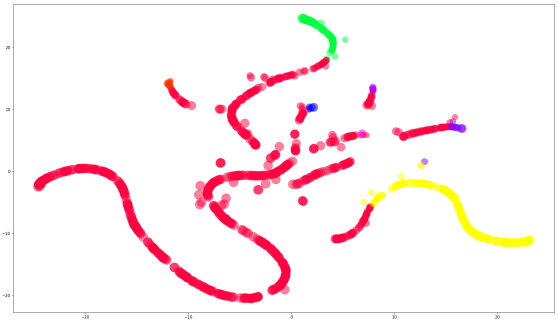
Figure 5.6: En-LDA, RPC and ToPIC results diagrams for dataset D1, TF-Entropy weighting schema.

K values are more unbalanced than the one obtained with $K = 5$. This solution is reported in bold in the ToPIC results table (Table 5.4).

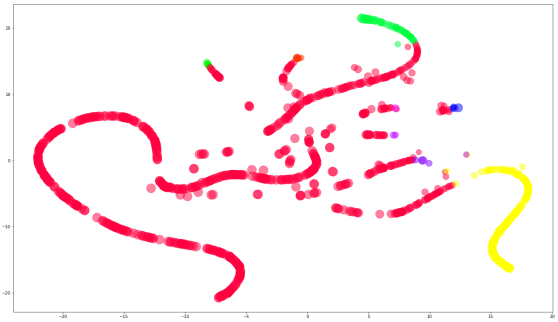
The choice of K equal to 5 as best results among the three proposed by the framework is confirmed by the other two approaches considered in the study. Indeed, for the TF-Entropy weighting schema, the state-of-the-art techniques propose as solutions the same values proposed by ToPIC. These results have already been shown in order to evaluate the performance of the ToPIC-Similarity technique.



(a) D1, TF-Entropy, K 5.



(b) D1, TF-Entropy, K 8.



(c) D1, TF-Entropy, K 9.

Figure 5.7: D1 t-SNE representation, TF-Entropy weighting schema, K 5, 8 and 9 respectively.

5.4.3 Dataset D1, LogTF-IDF results

The results obtained by ToPIC with the LogTF-IDF weighting schema are reported in figure 5.9b. From the chart, the selected K values are 8 and 17.

As in the TF-IDF weighting schema case, from table 5.4 it is possible to see that the Silhouette and the entropy indices have the same trend in evaluating the results. Again, perplexity on the contrary assigns to the model with more clusters a lower uncertainty. In order to correctly evaluate and choose the better modeling of the dataset, a deeper analysis of the proposed results is needed. To do that, the t-SNE and the topic-terms representations of the models are reported from figure 5.10 to figure 5.11.

The t-SNE visualization of the results shows us a slightly unbalanced clustering outcome. The effective number of categories and topics is indeed is not very well reflected from the structure inferred by the topic modeling and the t-SNE representation. Topic-terms visualization is needed to better understand the obtained results.

The results obtained with the LogTF-IDF weighting schemas (K equal to 8 and K equal to 17) show that LDA has more difficulty in well clustering the documents.

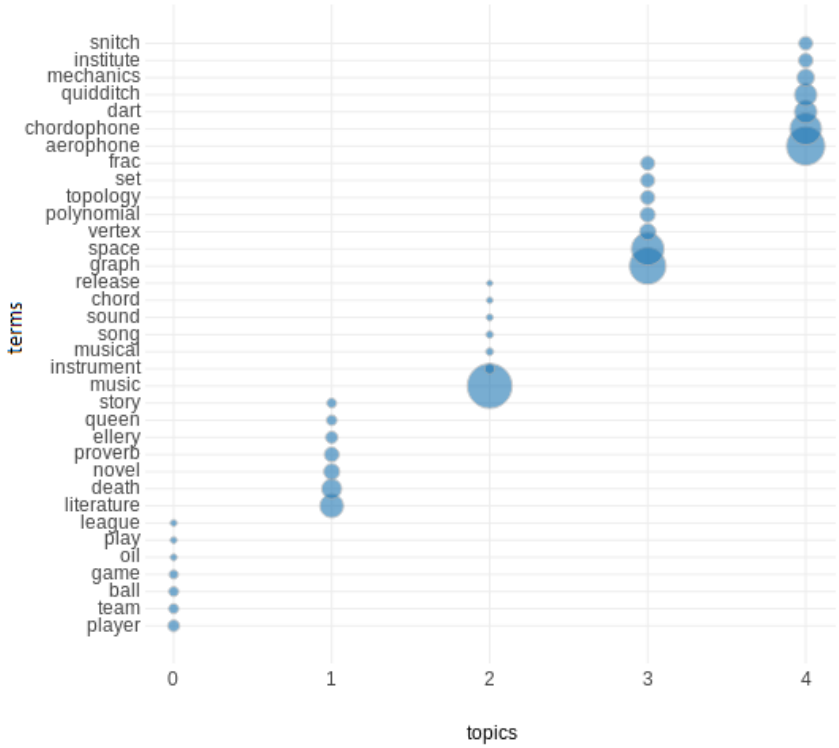
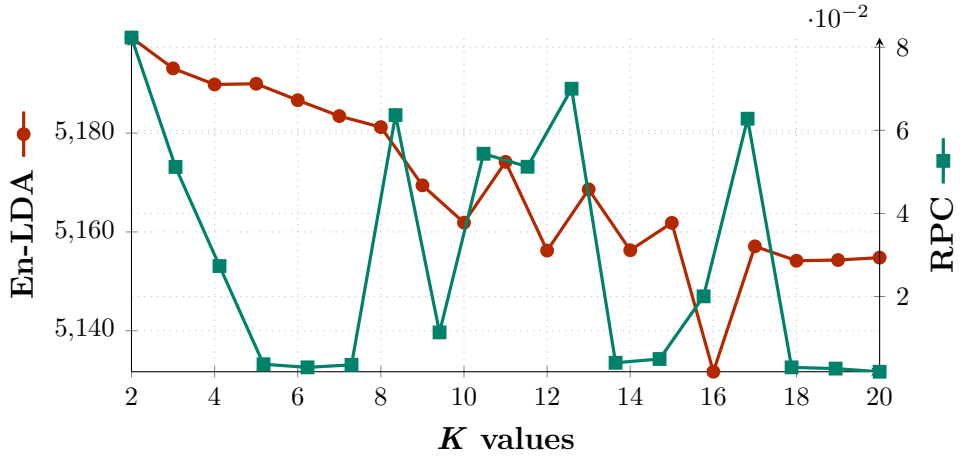
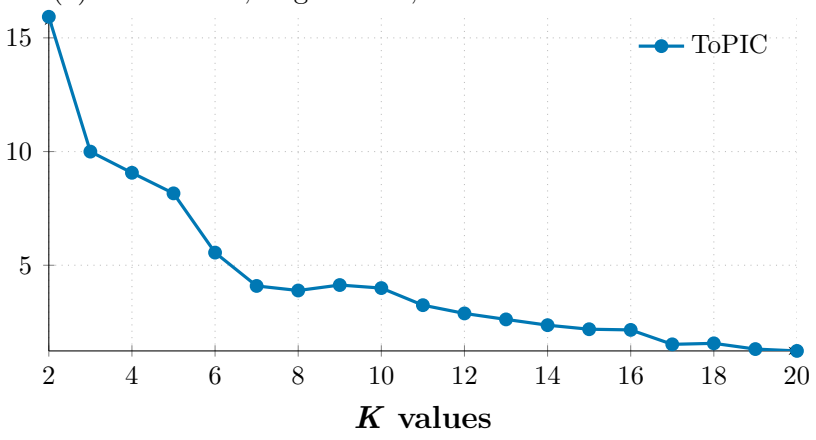


Figure 5.8: D1 Termite representation, TF-Entropy weighting schema, K 5.

However, looking at the t-SNE representation of results obtained with K equal to 8 (Figure 5.10a) it is possible to identify, among all the clusters, three main groups. They contain most of the documents of the corpus, and they are represented with the word clouds in Figure 5.11. The other topic descriptions are very specific and they describe very particular arguments (such as fighting sports or themes related to the East), so they are not relevant for the evaluation of the modeling.
Looking at results obtained with 17 different topics, the t-SNE representation shows a quite unbalanced clustering. Again, looking at the topic-terms representation and at the labels assigned to the documents, it is possible to better understand the LDA outcome. Many of the found topics seems to have a good level of cohesion, clearly dealing about sports, historical o literary characters, mathematics, literature, arts, while some others describe more specific topics. However, when looking at the number of documents assigned to each cluster, we can assess that the modeling did not well perform in dividing the texts. Indeed, the four topics including most of the documents (99%) result to be not the more generic, but the ones reported in Table 5.6.
The reported results are clearly not completely satisfying, since they do not well describe the texts collection. Because of these reason, the value selected to be the



(a) Dataset D1, LogTF-IDF, En-LDA and RPC results.

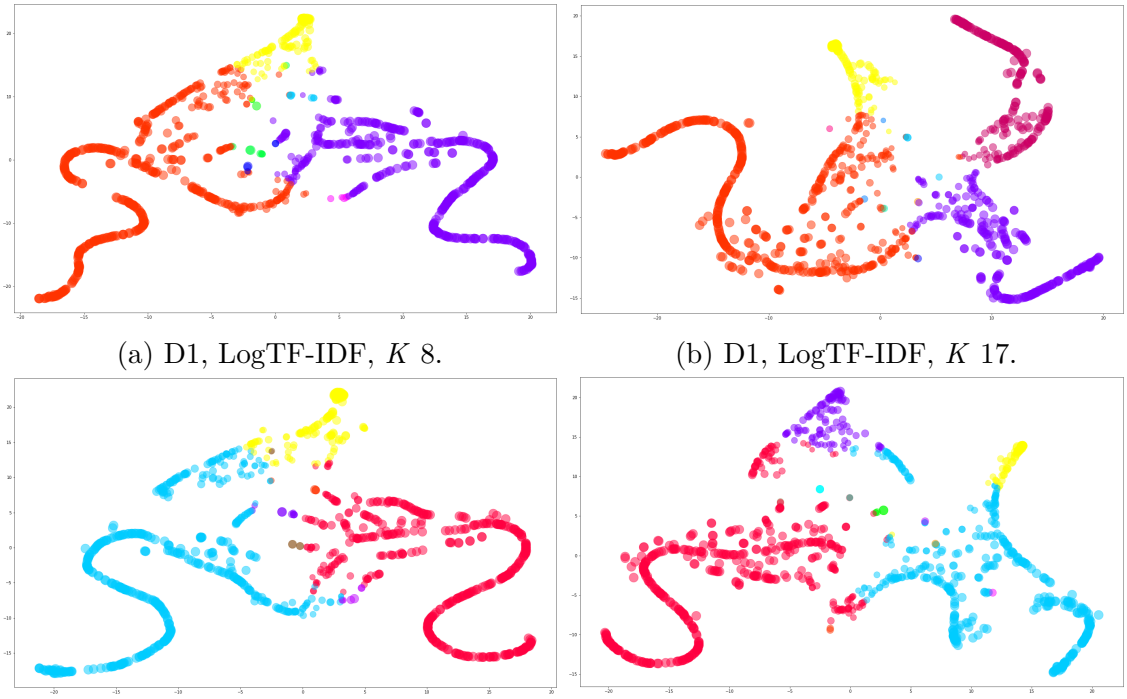


(b) Dataset D1, LogTF-IDF, ToPIC results.

Figure 5.9: En-LDA, RPC and ToPIC results diagrams for dataset D1, LogTF-IDF weighting schema.

optimal one is 8, as reported in bold in Table 5.4.
An explanation of the worsening of the results with respect to the previously presented weighting schema, resides in the local weight LogTF. The logarithm indeed flattens the relevant part of the weight associated to the terms. The logarithm, tends to make the weight distribution homogeneous, lowering especially the greater values. This makes the LDA modeling more difficult, thus giving the results showed before.

For the LogTF-IDF weighting schema, the state-of-the-art approaches propose 7 and



(a) D1, LogTF-IDF, K 8.

(b) D1, LogTF-IDF, K 17.

(c) D1, LogTF-IDF, K 7.

(d) D1, LogTF-IDF, K 16.

Figure 5.10: D1 t-SNE representation, LogTF-IDF weighting schema, K 8, 17, 7 and 16 respectively.

Figure 5.11: D1 WordCloud representation, LogTF-IDF weighting schema, K 8.

16 (RCP and En-LDA respectively) as optimal number of clusters. These results are similar to the one obtained with ToPIC. The t-SNE representations of these clustering outcomes are reported in Figure 5.10, while the same considerations done for the previously discussed results can be done as well for these clustering outcomes.

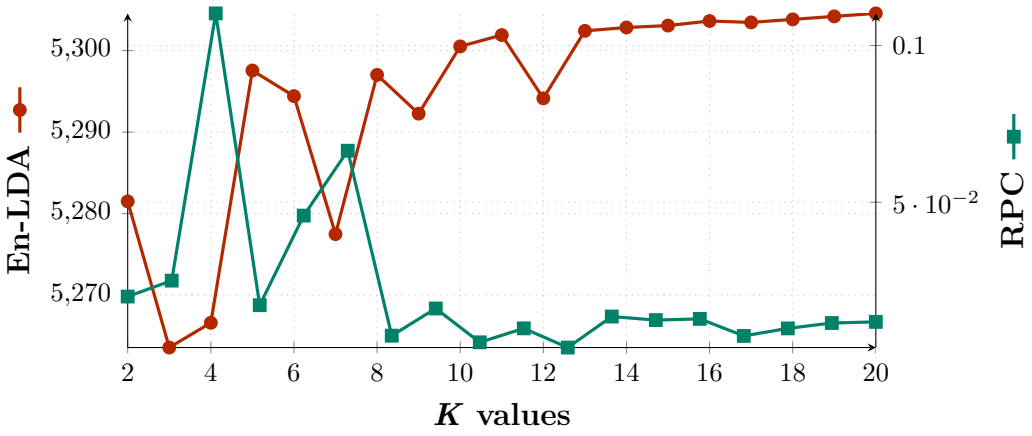
K	Topic description
1	ann, glima, umf, brejohn, thomas, robert, the, james, michael, king, richard, william, literature, henry, david, die, george, charles, sir, tale
4	make, two, sport, first, include, team, world, point, time, call, game, many, allow, competition, will, hold, form, often, player, take
5	music, write, work, example, include, often, first, term, see, time, form, new, musical, play, character, many, two, book, century, note
16	kickboxing, kickboxer, karate, muay, kickbox, bra, kyokushin, yamada, iska, tadashi, sawamura, thai, sae, sanshou, noguchi, khmer, sanda, kurosaki, osamu, kenji

Table 5.6: D1 topic-terms representation, LogTF-IDF weighting schema, K 17.

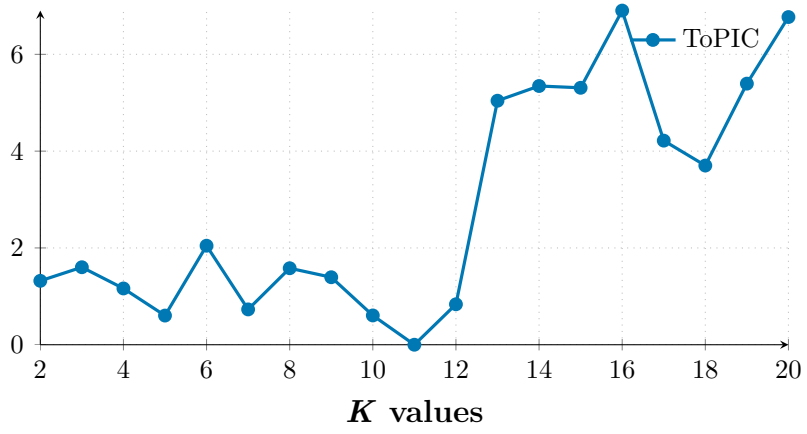
5.4.4 Dataset D1, LogTF-Entropy results

Figure 5.12 reports the results obtained by ToPIC with the LogTF-Entropy weighting schema. From the chart, the selected K values result to be 5, 7 and 11.
For the considered clustering results, perplexity, Silhouette and entropy all disagree in the evaluation of the models. As it is possible to see from the t-SNE representations, all the models badly perform in describing the texts documents. Indeed, besides having not cohesive topics that do not describe a specific argument, almost all the documents are assigned to a single cluster. Clearly, this clustering outcome does not represent the documents and their variety of contents.
Since even the topic-terms representation is not relevant for the evaluation of the clusters, the content of the topics is not described in the discussion of the results.
Nevertheless the model obtained with 7 clusters is considered to be the better one, given the perplexity index, the presence of two relevant topics (see in Figure 5.13b) that describe the *mathematics/music* and the *sport* topics (respectively red and yellow in the t-SNE chart). In the other clustering results, indeed, documents are almost all assigned to only one topic. This solution is reported in bold in the ToPIC results table.

The state-of-the-art approaches, both select 3 as optimal number of clusters. The t-SNE representation of this results is reported in Figure 5.13d: the produced clusters seem to be more balanced than the ones generated for other K values. Looking at the topic-terms representation, it is possible to identify the categories *music*, *sports* and *food* in the bigger cluster (yellow), *mathematics* in the blue one and *literature* in the smaller cluster.
Even if this result is better than the ones obtained with ToPIC, the LogTF-Entropy weighting schema clearly badly performs in giving relevance to the terms of corpus in order to help the clustering process.



(a) Dataset D1, LogTF-Entropy, En-LDA and RPC results.



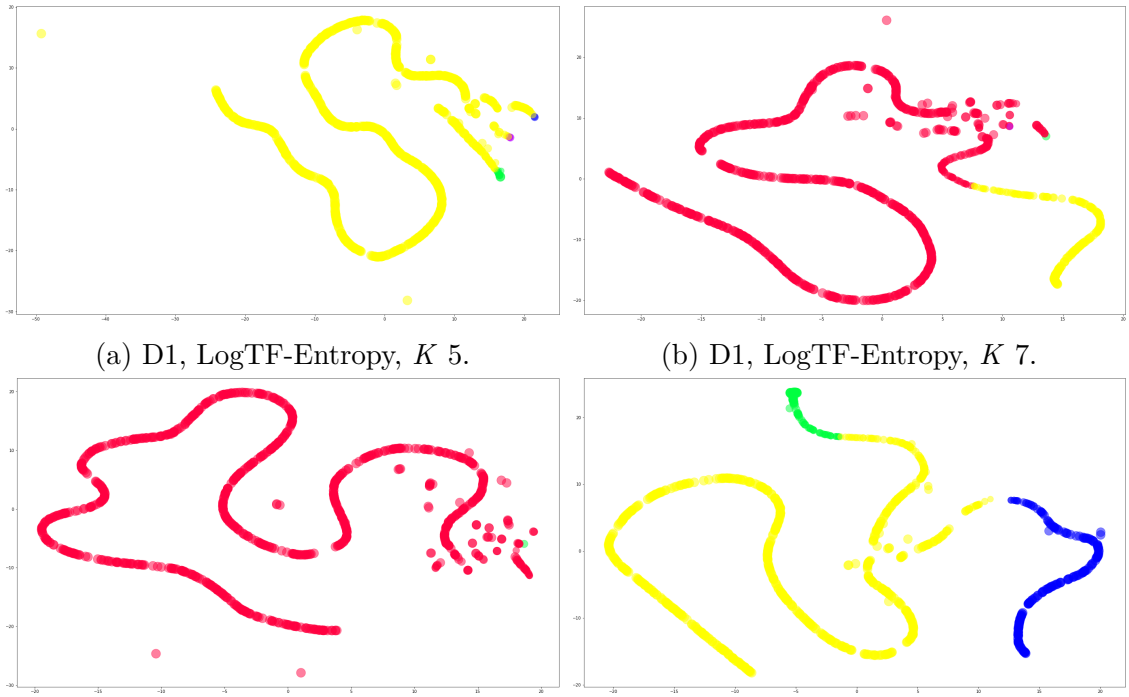
(b) Dataset D1, LogTF-Entropy, ToPIC results.

Figure 5.12: En-LDA, RPC and ToPIC results diagrams for dataset D1, LogTF-Entropy weighting schema.

The motivations of these affirmations and the bad performances of the modeling results obtained with this weighting function will be given in Section 5.6.

5.4.5 Dataset D1, Boolean-TF$_{glob}$ results

The results obtained by ToPIC with the Boolean-TF$_{glob}$ weighting schema are reported in Figure 5.14b. From the chart, the selected K values are 4, 5 and 17. As reported in table 5.4, the perplexity variation over the results is very low. However, among the solutions, it assesses the one with K equal to 17 to be the better one. On the contrary, Silhouette and entropy decisively classify it as the worst one.



(a) D1, LogTF-Entropy, K 5.

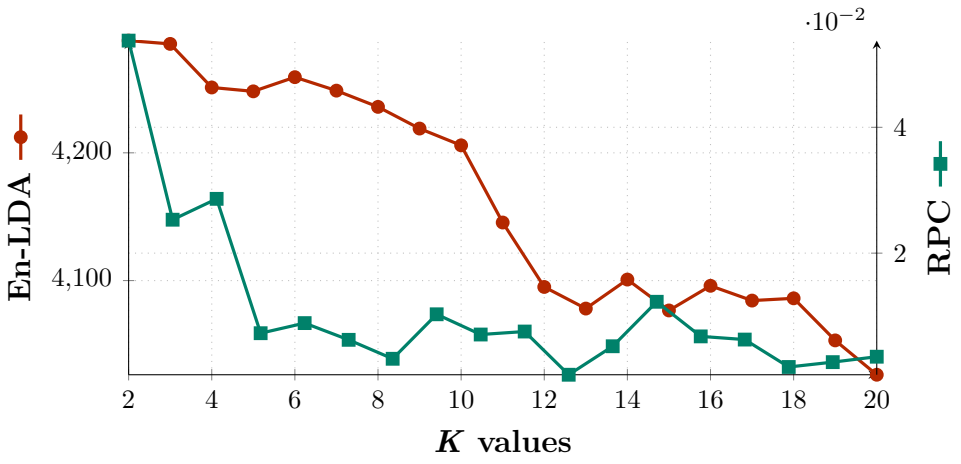
(b) D1, LogTF-Entropy, K 7.

(c) D1, LogTF-Entropy, K 11.

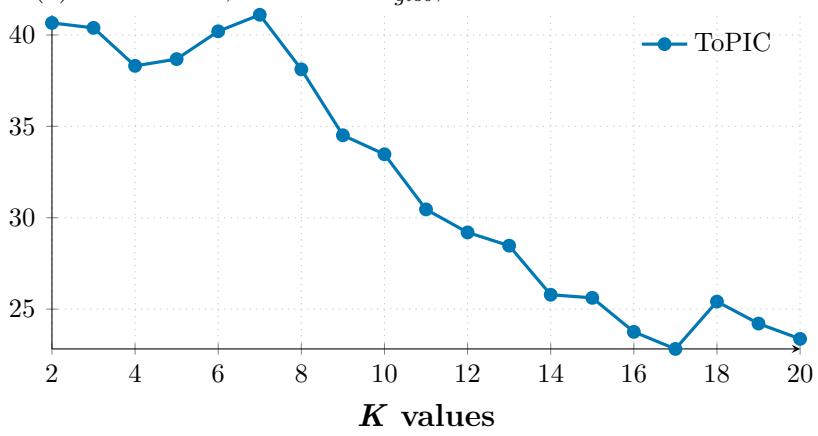
(d) D1, LogTF-Entropy, K 3.

Figure 5.13: D1 t-SNE representation, LogTF-Entropy weighting schema, K 5, 7, 11 and 3 respectively.

In order to have a clearer insight of the results, we have to look at the semantic representations of the models. They are reported from figure 5.15 to figure 5.16.
The t-SNE representation in Figure 5.15 shows that the obtained clustering results are balanced for all the found values. Looking at the topic-terms representation however, the obtained clusters seem to contain almost the same descriptions.
The representation in Figure 5.15b, even if at first sight seems to be worse than the others, it shows 5 well cohesive clusters with an outlier in right-upper corner: this result is not necessarily bad.
In the following discussion, has been seen that the LDA models obtained with the K values 4 and 5 are very similar. Because of this reason, to avoid similar visualizations, only the results related to $K = 5$ will be shown.
Considering the clustering obtained with K equal to 5 and its topic descriptions, when looking at the word clouds in Figure 5.16, many terms (such as *include* or *first*) appear to be in all the sets of the topic most representative words. This happens because the Boolean-TF$_{glob}$ weighting schema indeed gave more relevance to the words appearing most in the whole corpus. However, it often happens that these words do not belong to any specific topic, or they just do not bring any additional information useful for the modeling.



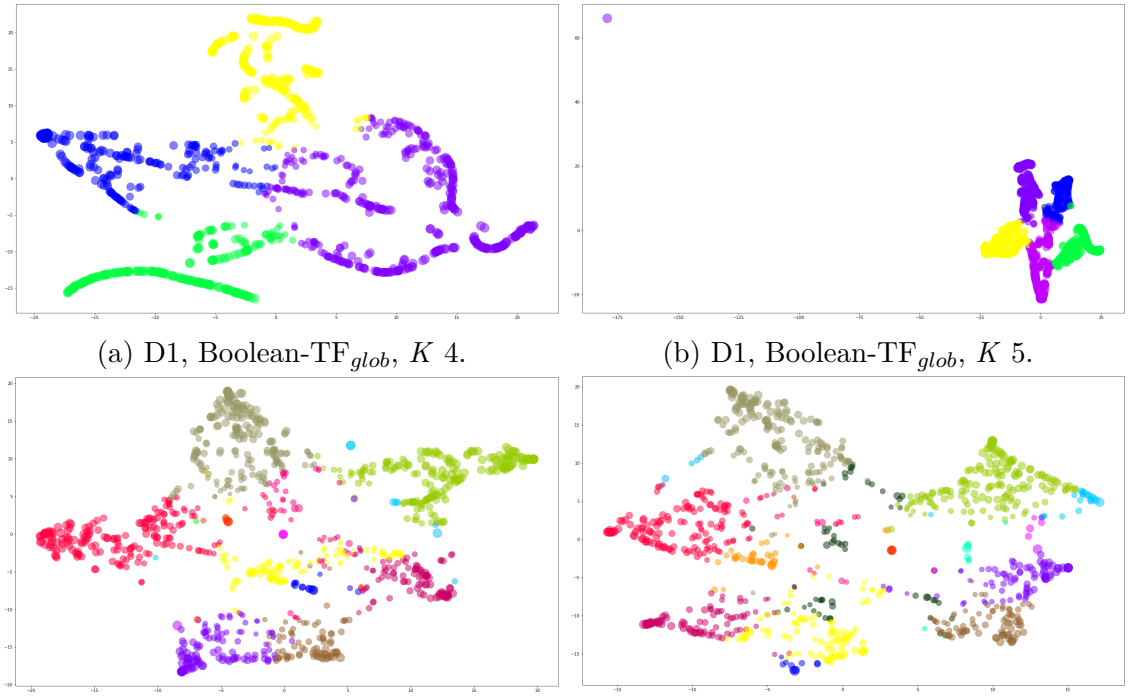
(a) Dataset D1, Boolean-TF$_{glob}$, En-LDA and RPC results.



(b) Dataset D1, Boolean-TF$_{glob}$, ToPIC results.

Figure 5.14: En-LDA, RPC and ToPIC results diagrams for dataset D1, Boolean-TF$_{glob}$ weighting schema.

In order to not consider these terms and bring up the words characterizing the topics identified by the LDA modeling process, we decided to apply a further post-processing step to evaluate the results. Once the models have been created and the K values selected, we took into consideration more words than the one used by ToPIC-Similarity to describe the topics, and then we removed from them all the words appearing at least in four topic representations. The results obtained by this postprocessing operation are reported in Table 5.7. In this way, the common words not bringing specific information has been excluded from the descriptions, and the terms relevant for the meaning of the categories are visible to the analysts.



(a) D1, Boolean-TF$_{glob}$, K 4.

(b) D1, Boolean-TF$_{glob}$, K 5.

(c) D1, Boolean-TF$_{glob}$, K 17.

(d) D1, Boolean-TF$_{glob}$, K 20.

Figure 5.15: D1 t-SNE representation, Boolean-TF$_{glob}$ weighting schema, K 4, 5, 17 and 20 respectively.

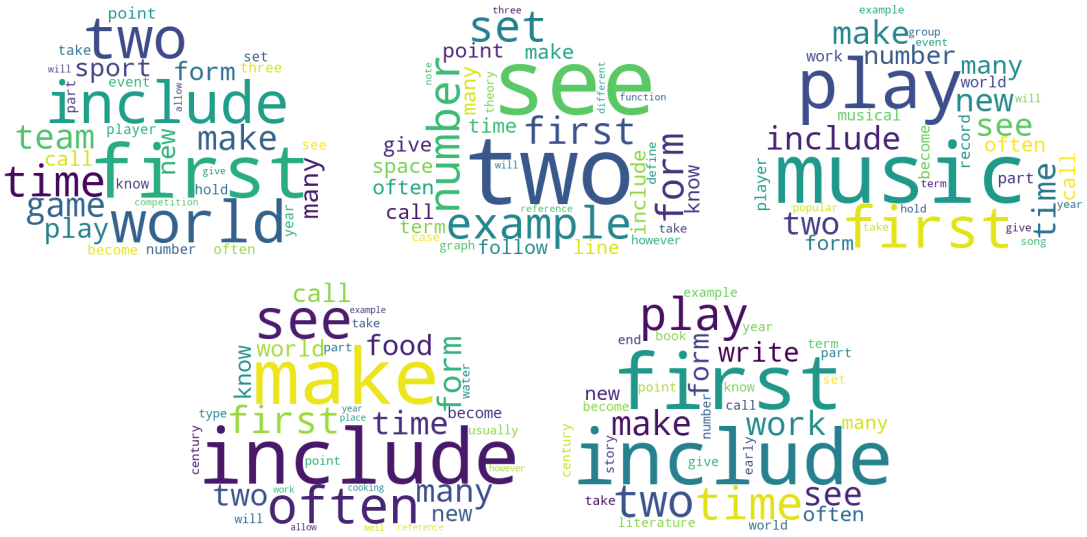


Figure 5.16: D1 WordCloud representation, Boolean-TF$_{glob}$ weighting schema, K 5.

Assigning labels to the clusters generated by the LDA model, the following main

topics cab be identified: *sport*, *mathematics*, *music*, *cooking*, *literature*.

K	**Topic description**
1	game, team, sport, player, event, competition, ball, rule, international, must, country, united, man, national, run
2	space, theory, case, graph, define, function, note, every, write, order, result, element, must, system, general
3	music, musical, player, record, song, event, write, release, instrument, note, sound, international, style, piece, back
4	food, water, cooking, united, sometimes, produce, result, high, oil, modern, large, require, must, list, process
5	write, book, literature, story, character, art, university, music, novel, modern, english, word, note, study, later

Table 5.7: D1 topic-terms representation, Boolean-TF$_{glob}$ weighting schema, K 5.

Analysing the clustering obtained with K equal to 17, the same issue is encountered. To assess the goodness of the results, the postprocessing is applied as well, not considering the terms appearing in more than nine clusters (namely, half of the cluster identified by K plus one, as it was done for the clustering with K equal to 5). For the visualization sake, in this discussion only the table showing the content is reported (Table 5.8), without the word cloud representation of the clusters.
From the analysis of the results, the most suitable selected number of topics turns out to be 5. This is reported in Table 5.4 by a bold line highlighting the result.

Concerning the state-of-the-art approaches, the RPC method selects 4 as optimal number of clusters. This value is equal to the lower one proposed by ToPIC and the t-SNE representation of this result is reported in Figure5.15a. The clustering is balanced and, applying the postprocessing applied also to the other results, satisfying partitions can be found (*sports*, *mathematics*, *music*, *literature*/*cooking*). On the other hand, En-LDA selects 20 as optimal value for K. This result, which t-SNE representation is reported in Figure 5.15d, is closer to the greater value identified by ToPIC. After applying the postprocessing good results can be found, similar to the ones obtained with K equal to 17.

5.4.6 Dataset D1: final observations and discussion

In the following part of the Section, a sum-up of the discussion about the results obtained for the dataset D1 with the different weighting schemas is reported. Firstly, an overall evaluation of the performance of ToPIC in detecting the topics of the documents in D1 will be given, then the impacts of the weighting schemas on the

K	Topic description
1	new, team, year, number, set, world, sport, united, take, event, part, example, will, follow, states
2	pattu, manipravalam, coral, champus, raja, ramanujan, rule, menon, player, give, set, write, work, example, charitam
3	food, cooking, new, know, reference, usually, term, become, water, popular, type, sometimes, part, oil, name
4	world, people, follow, base, end, show, link, since, product, know, external, reference, process, produce, however
5	play, world, work, new, write, become, example, year, set, story, take, literature, number, part, know
6	century, book, american, link, list, publish, type, external, reference, modern, french, later, name, english, follow
7	edge,design, however, surface, common, allow, usually, link, system, sound, require, especially, term, external, well
8	michael, richard, paul, james, peter, david, thomas, queen, jack, fire, award, ring, master, don, black
9	example, different, line, set, usually, will, take, end, give, reference, open, mean, however, contain, space
10	example, will, become, well, usually, place, set, take, number, method, know, however, part, different, allow
11	musical, water, series, human, drum, course, wind, air, rock, sea, white, ancient, production, pitch, piece
12	number, set, example, space, give, know, theory, term, graph, follow, case, write, note, three, new
13	game, player, play, sport, team, world, ball, line, know, three, take, end, year, league, competition
14	world, event, competition, hold, record, international, number, new, speed, body, three, man, start, modern, distance
15	ball, space, set, area, give, surface, reference, name, game, large, introduce, define, study, cover, list
16	music, play, new, work, write, example, term, become, number, part, musical, early, player, know, song
17	play, number, work, lead, change, must, base, term, take, new, type, major, study, level, however

Table 5.8: D1 topic-terms representation, Boolean-TF$_{glob}$ weighting schema, K 17.

clustering processes will be analysed. Lastly, general observations about the comparison of the results obtained with ToPIC and the state-of-the-art techniques are discussed.

For the analysis of dataset D1, ToPIC results to be an effective tool to help the analysts in the text mining activity. Indeed, the proposed framework finds for almost all the weighting schemas good clustering results, well describing the arguments and the topics in the corpus. The results mostly assign to the bigger clusters in the models the original main categories of the data collection. On the other hand, subtopics usually are assigned to few documents, that can eventually be considered together to main topics for a more general overview of the corpus arguments.
The different weighting schemas have different impacts on the clustering process. TF-IDF seems differentiate as much as possible the terms characterizing different topics. This really help the LDA topic modeling, that generally produces good results. The local weight LogTF and the global weight Entropy flatten the distribution of the words, thus producing a more difficult and challenging corpus to analyse. However, while the clustering generated with the LogTF-IDF weighting function still well performs in the describing the dataset, the TF-Entropy and especially the LogTF-Entropy provide the worse results. In these cases, it can be noticed that the clustering results worsen with greater K values: the Entropy global weights could be used in case the analysts desire few topics from the modeling. Instead, the Boolean-TF$_{glob}$ weighting schema, bringing relevance to the most frequent terms in the corpus, generates topics that are all described by similar words. However, this does not lead to bad results, since going deeper in the analysis of the topic representations it is possible to assess that the topics are actually different and actually describe the main categories of the datasets. Moreover, from the results we can state that clustering generated with the Boolean-TF$_{glob}$ weighting schema tends to find more cluster than the ones generated with the other weighting functions, still identifying the main topics of the corpus.
Comparing the results obtained for dataset D1 with ToPIC and the state-of-the-art techniques, we can assess that generally the smaller value among the ones produced by ToPIC is comparable to the results produced by RPC, while the greater values selected by the framework is in line with the results identified by En-LDA. Moreover, similar trends can be seen among the approaches when looking at the results obtained with different weighting schemas: for example the Entropy global weight generally provides less topics than IDF or the TF$_{glob}$. This confirms the ToPIC clustering outcomes being in line with the expected results. However, while RPC almost always finds very low number of clusters, ToPIC is able to propose results with different granularities. On the other hand, if En-LDA seems to be more sensitive to the weighting schemas, ToPIC outperforms it with respect to the efficiency of the algorithm and the computational costs.

5.5 Further results

In this Section, the results obtained for the datasets D2 and D3 are concisely reported (Section 5.5.1 and 5.5.2 respectively). For each dataset and for each considered weighting schema, the curves of the state-of-the-art approaches and the ToPIC index are illustrated. The t-SNE representations of the best result obtained by means of ToPIC and the results produced by RPC and En-LDA are shown next to the diagrams.

5.5.1 D2 results

Below, the results obtained for dataset D2 (the *Wikipedia_2500* articles collection) with TF-IDF, LogTF-IDF and Boolean-TF$_{glob}$ as weighting schemas are given.

TF-IDF weighting schema

Figure 5.17 shows the diagrams of the different approaches curves, obtained using the TF-IDF weighting schema. Figure 5.18 reports the t-SNE representations of the solutions found with ToPIC (best solutions among the ones proposed), RPC and En-LDA respectively.

LogTF-IDF weighting schema

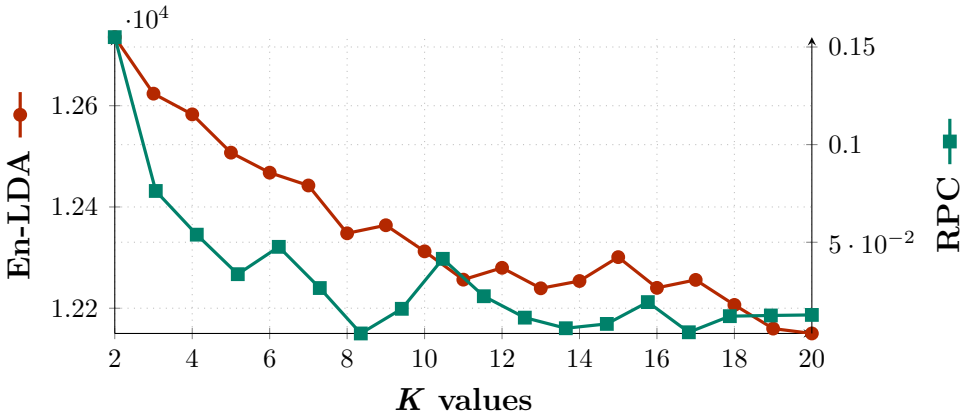
Figure 5.19 shows the diagrams of the different approaches curves, obtained using the TF-IDF weighting schema. Figure 5.20 reports the t-SNE representations of the solutions found with ToPIC (best solutions among the ones proposed), RPC and En-LDA respectively.

Boolean-TF$_{glob}$ weighting schema

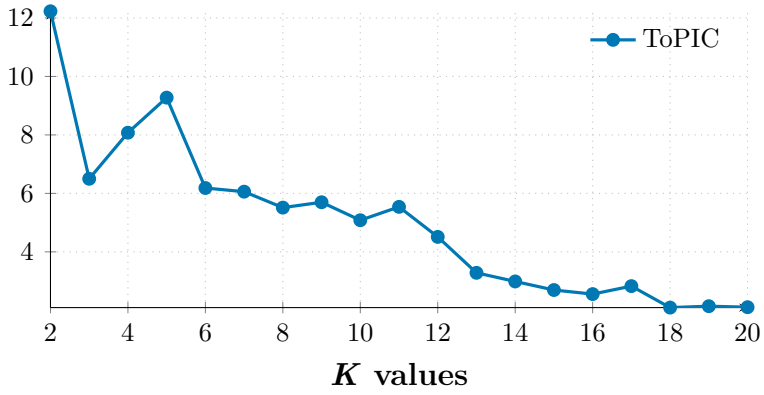
Figure 5.21 shows the diagrams of the different approaches curves, obtained using the TF-IDF weighting schema. Figure 5.22 reports the t-SNE representations of the solutions found with ToPIC (best solutions among the ones proposed), RPC and En-LDA respectively.

5.5.2 D3 results

Below, the results obtained for dataset D3 (the *Reuters-21578* collection) with TF-IDF, LogTF-IDF and Boolean-TF$_{glob}$ as weighting schemas are given.

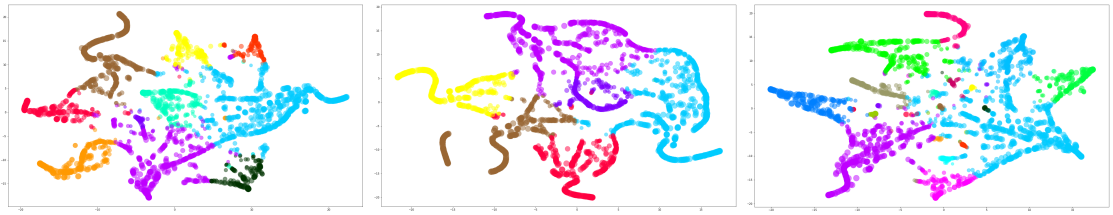


(a) Dataset D2, TF-IDF, En-LDA and RPC results.



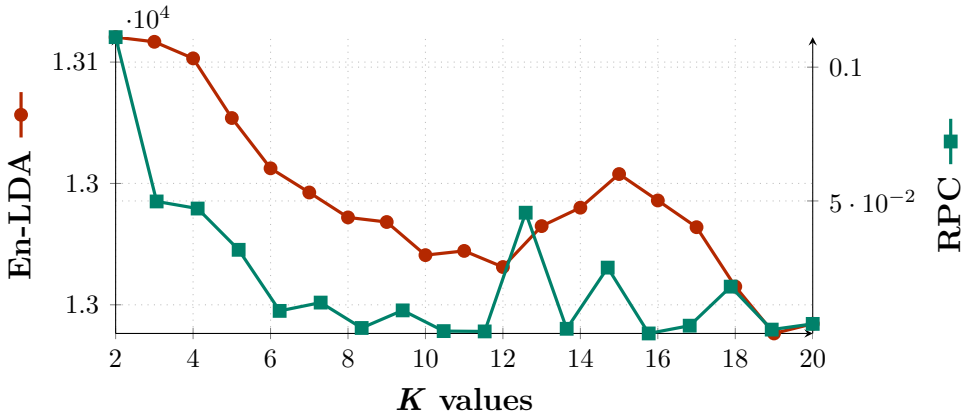
(b) Dataset D2, TF-IDF, ToPIC results.

Figure 5.17: En-LDA, RPC and ToPIC results diagrams for dataset D2, TF-IDF weighting schema.

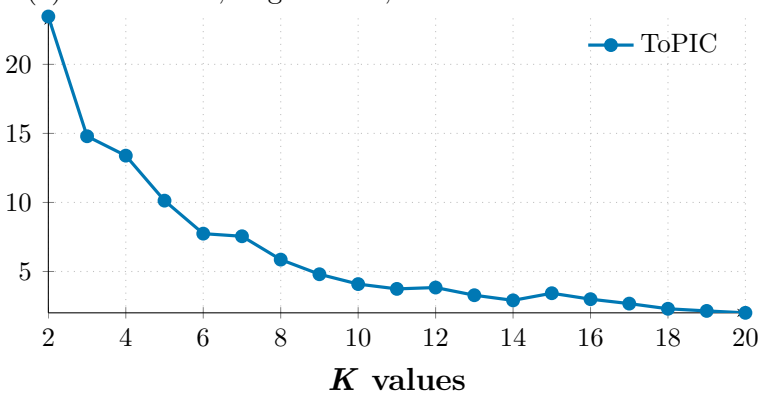


(a) D2, TF-IDF, K 10. (b) D2, TF-IDF, K 6. (c) D2, TF-IDF, K 20.

Figure 5.18: D2 t-SNE representation, TF-IDF weighting schema, K 10, 6 and 20 respectively.

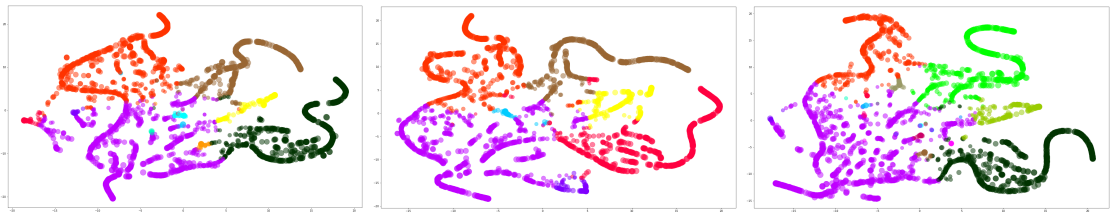


(a) Dataset D2, LogTF-IDF, En-LDA and RPC results.



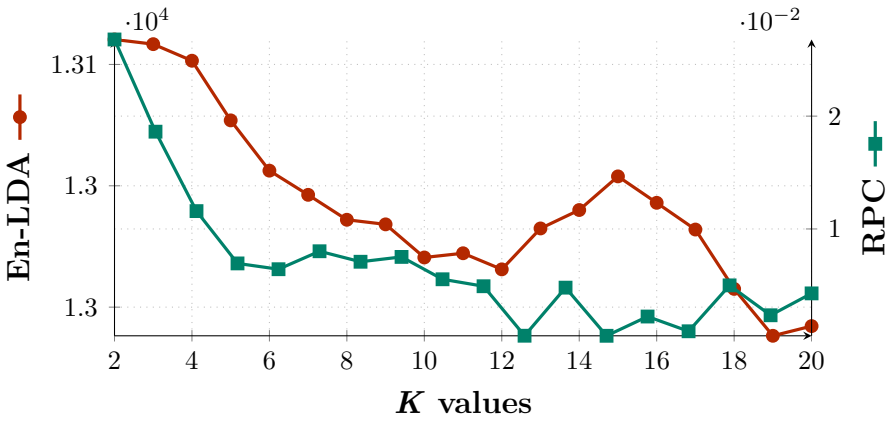
(b) Dataset D2, LogTF-IDF, ToPIC results.

Figure 5.19: En-LDA, RPC and ToPIC results diagrams for dataset D2, LogTF-IDF weighting schema.

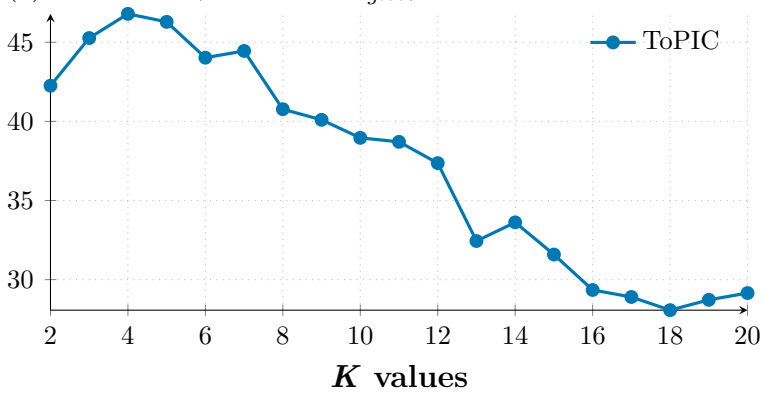


(a) D2, LogTF-IDF, K 11. (b) D2, LogTF-IDF, K 7. (c) D2, LogTF-IDF, K 19.

Figure 5.20: D2 t-SNE representation, LogTF-IDF weighting schema, K 11, 7 and 19 respectively.

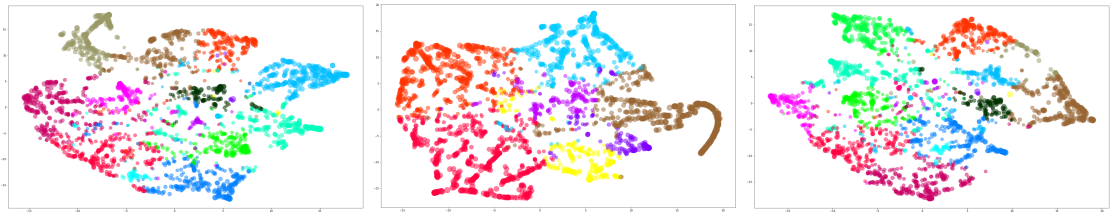


(a) Dataset D2, Boolean-TF$_{glob}$, En-LDA and RPC results.



(b) Dataset D2, Boolean-TF$_{glob}$, ToPIC results.

Figure 5.21: En-LDA, RPC and ToPIC results diagrams for dataset D2, Boolean-TF$_{glob}$ weighting schema.

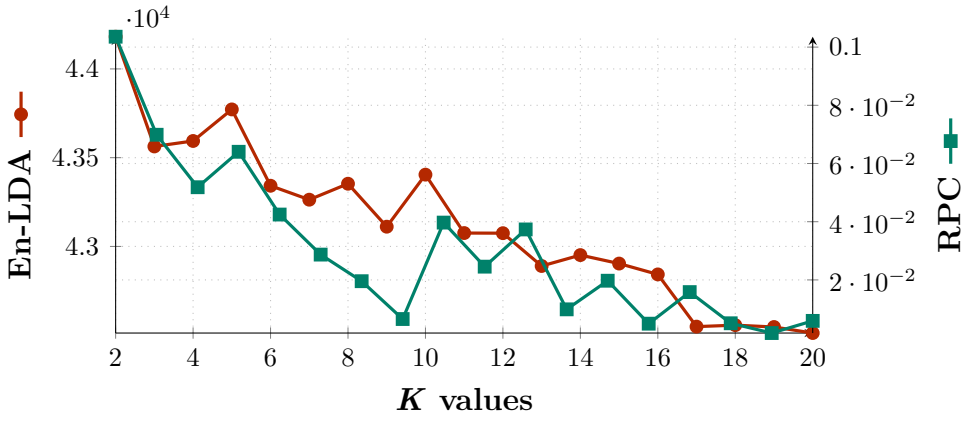


(a) D2, Bool-TF$_{glob}$, K 18. (b) D2, Bool-TF$_{glob}$, K 7. (c) D2, Bool-TF$_{glob}$, K 20.

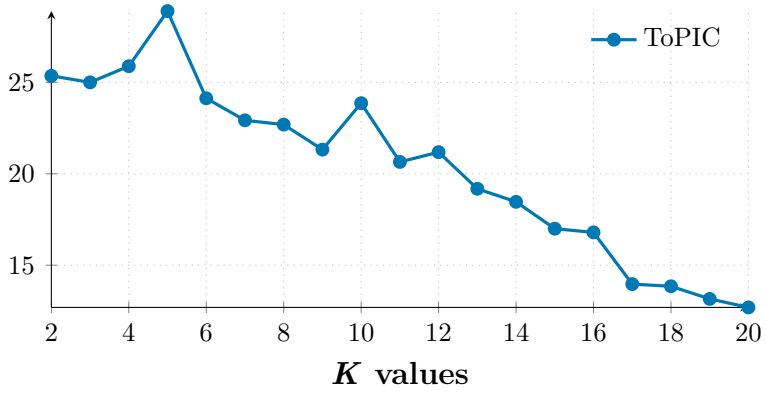
Figure 5.22: D2 t-SNE representation, Boolean-TF$_{glob}$ weighting schema, K 18, 7 and 20 respectively.

TF-IDF weighting schema

Figure 5.23 shows the diagrams of the different approaches curves, obtained using the TF-IDF weighting schema. Figure 5.18 reports the t-SNE representations of the solutions found with ToPIC (best solutions among the ones proposed), RPC and En-LDA respectively.



(a) Dataset D3, TF-IDF, En-LDA and RPC results.

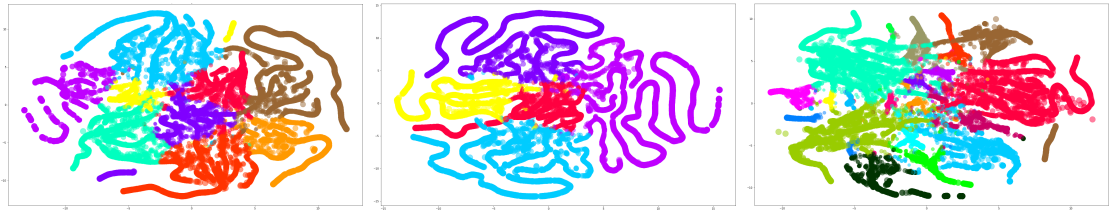


(b) Dataset D3, TF-IDF, ToPIC results.

Figure 5.23: En-LDA, RPC and ToPIC results diagrams for dataset D3, TF-IDF weighting schema.

LogTF-IDF weighting schema

Figure 5.25 shows the diagrams of the different approaches curves, obtained using the TF-IDF weighting schema. Figure 5.20 reports the t-SNE representations of the



(a) D3, TF-IDF, K 9. (b) D3, TF-IDF, K 5. (c) D3, TF-IDF, K 20.

Figure 5.24: D3 t-SNE representation, TF-IDF weighting schema, K 9, 5 and 20 respectively.

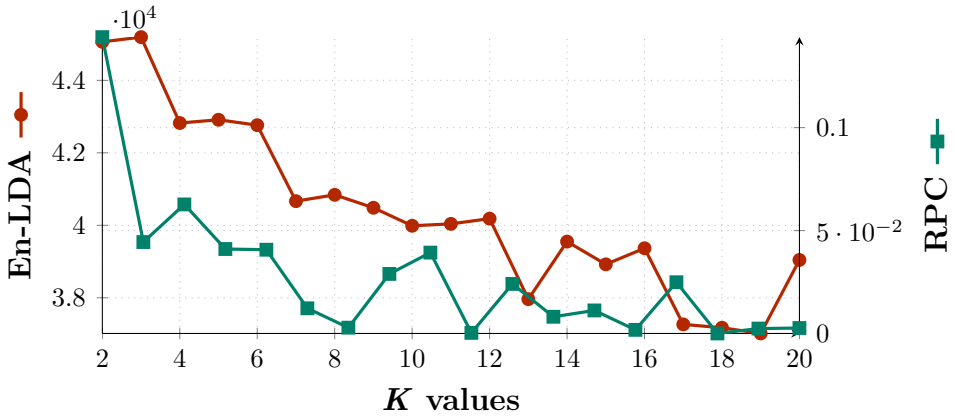
solutions found with ToPIC (best solutions among the ones proposed), RPC and En-LDA respectively.

Boolean-TF$_{glob}$ weighting schema

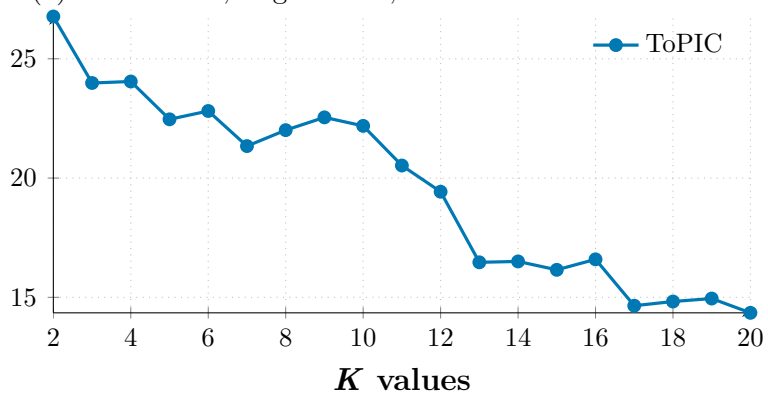
Figure 5.27 shows the diagrams of the different approaches curves, obtained using the TF-IDF weighting schema. Figure 5.22 reports the t-SNE representations of the solutions found with ToPIC (best solutions among the ones proposed), RPC and En-LDA respectively.

5.6 Weighting impacts

Since weights highlight the importance of words within documents, analysing how different weighting schemas affect the model is important. For the representative dataset under analysis, we computed the histogram of the TF-IDF and LogTF-Entropy weights. The values of LogTF-Entropy present almost a uniform distribution in the range [0,1] (Kurtosis index > 0 and standard deviation $= 0.5$) and the distribution has maximum value 8. With the IDF, instead, an asymmetrical bell distribution is obtained with average values between [2, 5] (Kurtosis index > 0 and standard deviation $= 12.7$) and maximum value equal to 1161. From the histograms, and also from the results analysed in the previous Sections, we can assess that the IDF weight schema better differentiates the weights within the corpus, thus producing a probabilistic model with better performances.
As for the running example, also for all the other datasets has been observed that the Entropy global weight badly performs in bringing relevance to the words. This can be explained by the visualization charts (t-SNE and Termite), even if the quantitative evaluation metrics (perplexity, Silhouette and entropy) can not spot the bad results of the clustering produced with these weighting schemas. Indeed, the probabilistic quantitative metrics evaluate the confidence the model has in assigning the documents the topic labels. In the results obtained with the Entropy global

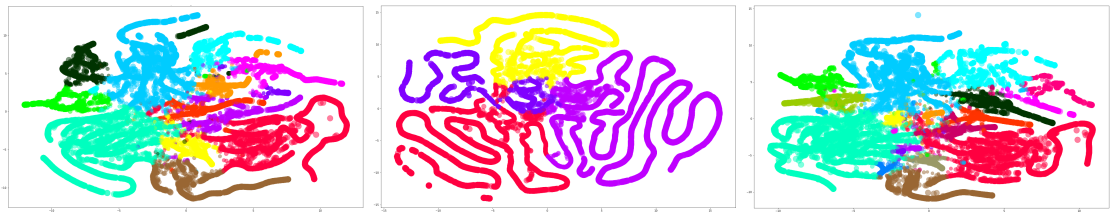


(a) Dataset D3, LogTF-IDF, En-LDA and RPC results.



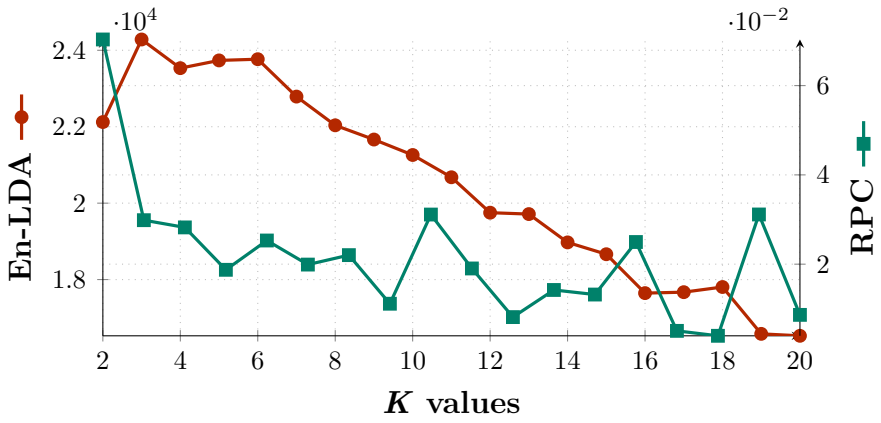
(b) Dataset D3, LogTF-IDF, ToPIC results.

Figure 5.25: En-LDA, RPC and ToPIC results diagrams for dataset D32, LogTF-IDF weighting schema.

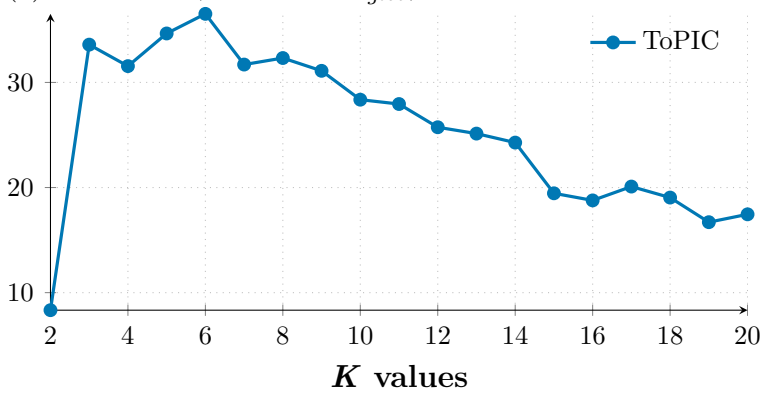


(a) D3, LogTF-IDF, K 13. (b) D3, LogTF-IDF, K 4. (c) D3, LogTF-IDF, K 19.

Figure 5.26: D3 t-SNE representation, LogTF-IDF weighting schema, K 13, 4 and 19 respectively.

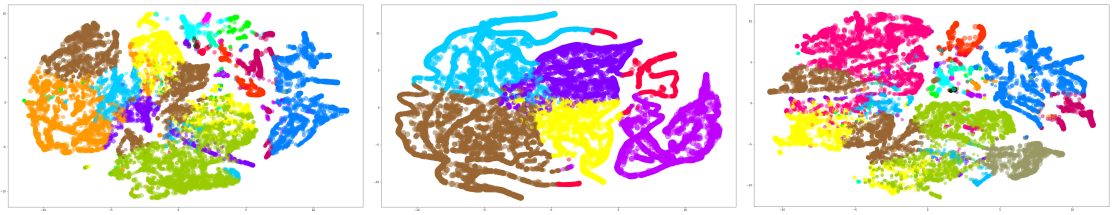


(a) Dataset D3, Boolean-TF$_{glob}$, En-LDA and RPC results.



(b) Dataset D3, Boolean-TF$_{glob}$, ToPIC results.

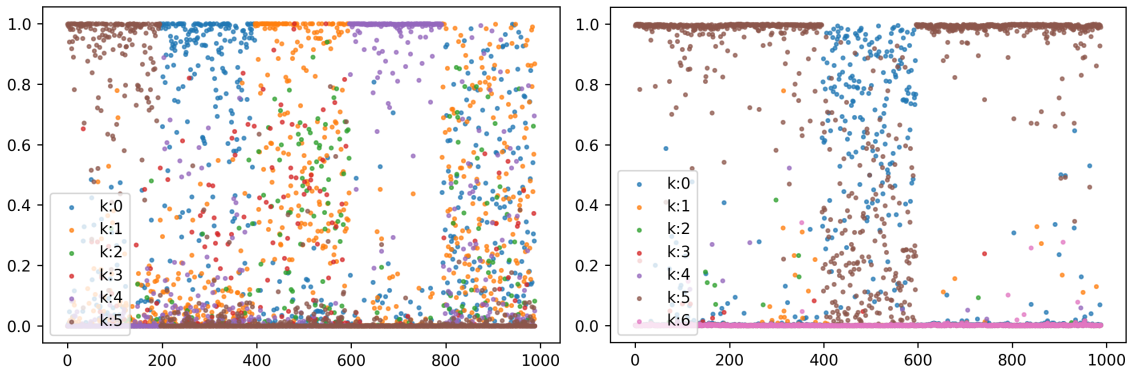
Figure 5.27: En-LDA, RPC and ToPIC results diagrams for dataset D3, Boolean-TF$_{glob}$ weighting schema.



(a) D3, Bool-TF$_{glob}$, K 16. (b) D3, Bool-TF$_{glob}$, K 6. (c) D3, Bool-TF$_{glob}$, K 20.

Figure 5.28: D3 t-SNE representation, Boolean-TF$_{glob}$ weighting schema, K 16, 6 and 20 respectively.

weight, even if erroneously, LDA assigns with a high confidence the document to a topic, thus leading the quality metrics to not spot badly performance of the model in dividing the corpus in different cohesive clusters.



(a) D1, TF-IDF, K 6.

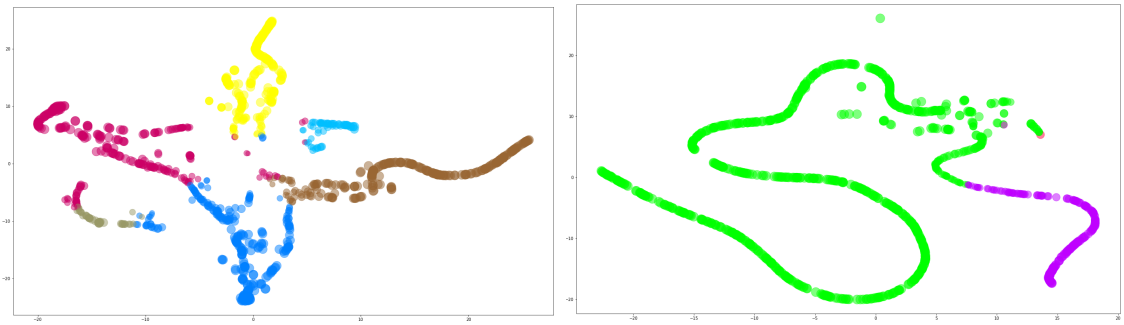
(b) D1, LogTF-Entropy, K 7.

Figure 5.29: D1 documents-topics probabilities, TF-IDF and LogTF-Entropy weighting schemas, K 6 and 7 respectively.

As a matter of fact, Figure 5.29a and Figure 5.29b show, for the LDA models obtained with the two weights, the probability distribution of each document of the corpus D1 to belong to the K topics selected by the algorithm (6 for TF-IDF and 7 for LogTF-Entropy). Figures 5.30a and 5.30b show instead the t-SNE representation of the clustering results. In detail, the documents present a more homogeneous distribution using IDF weight, with topics balanced by number of documents. Instead, with the Entropy weight, there is one cluster in which 90% of the documents have a probability greater than 0.90 of membership. It turns out that 90% of the documents belong to a single cluster (topic) and the result is due to the fact that the weight Entropy fails to isolate the most relevant terms within the collection of documents.

When unbalanced clusters are generated, the use of only goodness metrics is not able to guarantee good performance. Indeed, high values of Silhouette or low values of entropy do not involve a good clustering but represent a simplification of the problem. It is like classifying 90% of the documents in a single topic, thus generating many false negatives. Having the class label available, indices such as recall or precision could help identify these incorrect assignments. However, if the label were not available, the use of quantitative indicators would not be effective. Methods that take into account the semantics must be presented.

Because of these considerations on the weighting impact and the bad results obtained by clustering processes having Entropy as global weights, these will not be considered in the ToPIC results discussion.



(a) D1, TF-IDF, K 6.

(b) D1, LogTF-Entropy, K 7.

Figure 5.30: D1 t-SNE representation, TF-IDF and LogTF-Entropy weighting schemas, K 6 and 7 respectively.

5.7 Comparison with the state-of-the-art

In this Section, a comparison of the results obtained with ToPIC and state-of-the-art techniques is presented. Table 5.5 shows the results obtained with the state-of-the-art approaches and their evaluation using the same metrics used in Table 5.4. These values should be compared with the ToPIC solutions reported in bold in Table 5.5.
Several trends can be found in the comparison. Generally, RPC tends to almost always finds very low number of clusters, independently from the used weighting schema. On the contrary, ToPIC is able to propose results with different granularity levels. However, it is observable that the smaller number of topics found by ToPIC is comparable with the value found by the RPC.
En-LDA, instead, tends to find greater values for the K parameter and seems to be more sensitive to the used weighting schema. Indeed, for example when using the Entropy global weight En-LDA agrees with ToPIC in finding less topics in the corpus. En-LDA, which sometimes even takes the upper bound of the possible K values as the optimal solution, is comparable to ToPIC considering its greater solution, and especially using as weighting function the Boolean-TF$_{glob}$.
Moreover, when using different weighting schemas, the three different approaches tend to follow a similar trend: for example, TF$_{glob}$ normally generates more topics than the global weight IDF, and even more with respect to the Entropy. This confirms the ToPIC results being in line with the expected results.
Furthermore, with respect to the state-of-the-art techniques, ToPIC considers the semantic descriptions of the topics to assess the level of separation of the clusters. This is not considered in the state-of-the-art approaches, that only evaluate the goodness of the results by means of probabilistic metrics. In ToPIC, the quantitative indices of confidence, could be used instead to deeper analyse the proposed results.

Looking at the computational costs, ToPIC outperforms En-LDA and it is comparable with RPC. Calculating En-LDA indices is computationally very expensive, and the number of iterations explodes with the growth in documents vocabulary and the cardinality of the corpus. Furthermore En-LDA needs to be computed for all the topics in the given set, having to find the entropy minimum among all the possible K possibilities. Instead, RPC requires, in the wort case, a computational time comparable to the one required by ToPIC, not having to be computed for all the possible K values in the set $[K_{min}, K_{max}]$ defined by the user.

5.8 ToPIC final considerations

From the obtained experimental results, we can assess that ToPIC well performs in describing the corpora under analysis, clustering the documents based on their content. Indeed, the results show that the framework, by means of ToPIC-Similarity, is generally able to group the texts in well separated topics.
Considering the semantic similarity among the produced topics turned out to outperform the current used approach to find proper number of clusters. Indeed, ToPIC-Similarity is able to capture the effective cohesion level of the clusters, and then properly identify the optimal number of topics. The results obtained from all the datasets considered in the thesis confirm the clusters to be well separated, especially for certain weighting schemas such as TF-IDF.
ToPIC results to be an effective tool for topic modeling. Specifically about the selection of the number of topics, the framework outperforms the state-of-the-art approaches, performing in the worst case as the considered techniques considered in the study. Indeed, RPC tends to find too small values for the K parameter, regardless of the used weighting schema. On the contrary En-LDA is more sensitive to the weighting functions used in the analysis, but it is way too expensive in terms of computational costs.
ToPIC turns out to be really helpful for analyst in the analytic tasks. Indeed, the analyst can choose to assign to the words in the documents different relevance by means of different weights, then choose the granularity level required for her analysis (by means of the different K values proposed by ToPIC), and then evaluate the results by means of different evaluation techniques highlighting different aspects of the clustering. Thus, ToPIC can effectively lead the analysis process of textual data collections.

Chapter 6

Conclusion

This thesis presents a self-tuning data analytics system that effectively mines several textual data collections with different characteristics. The proposed framework, named ToPIC, includes ad-hoc auto-selection strategies to streamline the analytics process and off-load the parameter tuning from end-user. ToPIC features a distributed implementation in Apache Spark supporting parallel and scalable processing.

This study starts exploring the already existing approaches and solutions to automatically analyse and discover topics in textual datasets. Starting from PASTA, a distributed engine that automatically clusters collections of documents grouping them based on coherent and well separated contents, we propose a novel framework based on a different type of document representation and modeling. The proposed framework is ToPIC (T*uning* o*f* P*arameters for* I*nference of* C*oncepts*), and it offers topic modeling functionalities using a probabilistic model (i.e. Latent Dirichlet Allocation) instead of an algebraic one. Intuitively, in the LDA topic modeling, documents can be associated with a particular topic or can be seen as a mixture of topics in different proportions, and certain words can be expected to appear in a document more or less frequently. This topic modeling algorithm is then able to describe topics (and so documents) by means of similar word clusters. Since text mining and topic modeling requires the constant supervision of the analysts, ToPIC is built in order to free the analysis process from the human supervision and avoid the field expertise having to manually set the parameters of the involved algorithms. To do that, scientific literature have been explored to reach the state-of-the-art results by using the cutting-edge technologies and algorithms.
However, a novel approach to select proper values for the parameter modeling the number of topics to discover and divide the corpus in (main parameter in the whole framework and currently open research issue in literature) has been studied and proposed with the framework. This approach, named ToPIC-Similarity, does not only consider the quantitative probabilistic indices of the whole model, but it

consider the topics content and description, specifically evaluating their semantic similarity. ToPIC also includes different weighting functions, to explore how local and global weights of the terms in the documents collection influence the outcome of the modeling.
The proposed framework has been validated on different real data collections, characterized by different statistical indices and properties: Wikipedia articles, the Reuters-21578 collection, Twitter, PubMed and Journal collections.
The experimental results show the effectiveness and the efficiency of the proposed framework ToPIC. The results have been evaluated considering quantitative metrics such as perplexity, Silhouette and entropy, but also by means of visualizations techniques, to not disregard the effective structure and content of the results. These techniques include the t-SNE visualization and the topic-terms representation (Termite and word clouds).
At the end, ToPIC turned out to outperform the current state-of-the-art techniques aiming to automatically select optimal number of clusters, both in the quality of the mining activity results and in the computational costs of the algorithms, resulting to be an effective tool to describe and cluster textual datasets off-loading the parameter tuning from the end-user.

Possible extensions of the current work are (i) the inclusion of further probabilistic data transformation methods (e.g., p-LSI), (ii) the design of a self-learning strategy able to suggest good configurations which yield higher quality knowledge without performing the analytics task and (iii) the improvement of the characterization of the topics' semantic description to perform better modeling for a given data collection.

Bibliography

[1] D. Biber, S. Conrad, and G. N. Leech. *Termite: Longman student grammar of spoken and written English*. Harlow, Essex: Longman, 2002.

[2] D. M. Blei. Probabilistic topic models. *Commun. ACM*, 55(4):77–84, Apr. 2012.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[4] T. Cerquitelli, E. Di Corso, F. Ventura, and S. Chiusano. Data miners' little helper: Data transformation activity cues for cluster analysis on document collections. In *Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics*, WIMS '17, pages 27:1–27:6, New York, NY, USA, 2017. ACM.

[5] J. Chuang, C. D. Manning, and J. Heer. Termite: Visualization techniques for assessing textual topic models. In *Proceedings of the international working conference on advanced visual interfaces*, pages 74–77. ACM, 2012.

[6] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[7] E. Di Corso, T. Cerquitelli, and F. Ventura. Self-tuning techniques for large scale cluster analysis on textual data collections. In *Proceedings of the Symposium on Applied Computing*, pages 771–776. ACM, 2017.

[8] E. di Corso, F. Ventura, and T. Cerquitelli. All in a twitter: Self-tuning strategies for a deeper understanding of a crisis tweet collection. pages 3722–3726, 12 2017.

[9] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National academy of Sciences*, 101(suppl 1):5228–5235, 2004.

[10] H. Karau, A. Konwinski, P. Wendell, and M. Zaharia. *Learning Spark: Lightning-Fast Big Data Analytics*. O'Reilly Media, Inc., 1st edition, 2015.

[11] S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[12] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[13] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[14] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[15] I. Saleh and N. El-Tazi. Automatic organization of semantically related tags using topic modelling. In *Advances in Databases and Information Systems*, pages 235–245. Springer, 2017.

[16] A. Spark. The Apache Spark scalable machine learning library. Available: https://spark.apache.org/mllib/. Last access on October 2016. 2016.

[17] E. Spertus, M. Sahami, and O. Buyukkokten. Evaluating similarity measures: a large-scale study in the orkut social network. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 678–684. ACM, 2005.

[18] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[19] W. B. Towne, C. P. Rosé, and J. D. Herbsleb. Measuring similarity similarly: Lda and human perception. *ACM TIST*, 8(1):7–1, 2016.

[20] J. Wood, P. Tan, W. Wang, and C. Arnold. Source-lda: Enhancing probabilistic topic models using prior knowledge sources. In *Data Engineering (ICDE), 2017 IEEE 33rd International Conference on*, pages 411–422. IEEE, 2017.

[21] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 2–2. USENIX Association, 2012.

[22] W. Zhang, Y. Cui, and T. Yoshida. En-lda: An novel approach to automatic bug report assignment with entropy optimized latent dirichlet allocation. *Entropy*, 19(5):173, 2017.

[23] W. Zhao, J. J. Chen, R. Perkins, Z. Liu, W. Ge, Y. Ding, and W. Zou. A heuristic approach to determine an appropriate number of topics in topic modeling. *BMC bioinformatics*, 16(13):S8, 2015.