FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Scaling-up organization of Document Sets to facilitate their analysis

Rui Portocarrero Sarmento
PRODEI-FEUP
FCT FOUNDATION

Programa Doutoral em Engenharia Informática

Supervisor: Prof. Dr. João Gama

Co-Supervisor: Prof. Dr. Pavel Brazdil

September 5, 2023

Contents

Glossary **9**

Acknowledgments **1**

Abstract **2**

Resumo **3**

1 Introduction **5**
1.1 Motivation 5
1.2 Main Objectives 6
1.3 Main Contributions 7
1.4 Outline 8

2 Base Concepts and Related Work **11**
2.1 Text documents 12
2.2 Clustering Evaluation 13
2.3 External Clustering Validation Measures 14
2.4 Sampling Large Static Networks 18
2.5 Visualization 19
2.6 Centrality Measures 20
2.7 Feature/Keyword Selection 24
2.8 Text Sparsity and Sparse Matrices 26

3 Affinity Miner Prototype, Concept and Characteristics **29**
3.1 Affinity Miner Case Study 32
3.2 Summary 35
3.3 Problems to Address 36
3.4 Related Work 37
3.4.1 Base Concepts 37
3.4.2 Used Methodology 40
3.4.3 Extraction of Topics 41
3.4.4 LDA 42

3.4.5 TF-IDF . . . 42
3.4.6 TF-IDF with Clustering . . . 43
3.4.7 Stemming . . . 43
3.4.8 Evaluation of Topics/Keywords Generated . . . 43
3.4.9 Results . . . 44
3.5 Summary and Contributions . . . 47

4 Organizing Large Datasets of Text Documents 49
4.1 Methodology for Sampling Networks . . . 50
4.1.1 Top-K Sampling with Top-K itemsets . . . 50
4.1.2 Open Issues . . . 54
4.1.3 Summary and Contributions . . . 55
4.2 Community Detection . . . 56
4.2.1 Introduction . . . 56
4.2.2 Related Work . . . 56
4.2.3 Community Detection and Average Density . . . 59
Implemented Algorithms . . . 61
4.2.4 Case Study and Evaluation . . . 62
Results on Real Networks . . . 63
4.2.5 Summary and Contributions . . . 68
4.3 Centrality Analysis . . . 70
4.3.1 Introduction . . . 70
4.3.2 Related Work . . . 70
Incremental Centrality Measures . . . 70
Incremental Betweenness Centrality . . . 70
Incremental Closeness Centrality . . . 71
4.3.3 Incremental Laplace Centrality Algorithm . . . 71
Locality of the Laplacian Centrality . . . 71
4.3.4 Text Node Centrality - Laplace Centrality Solution . . . 73
4.3.5 Summary and Contributions . . . 75
4.4 Incremental Graph-based Keyword Representation . . . 77
4.4.1 Introduction . . . 77
4.4.2 Related Work . . . 77
Space Saving Top-K . . . 77
TextRank for Text Streams . . . 78
Keywords Extraction . . . 78
Local or Global "Online Topic Modeling" . . . 78
Incremental PageRank . . . 80
Window-based Streaming TextRank . . . 81
4.4.3 Incremental TextRank . . . 81

Algorithmic Analysis . . . 82
4.4.4 Case Study and Evaluation . . . 83
Description of the Data . . . 83
Methodology . . . 84
Results . . . 86
4.4.5 Summary and Contributions . . . 88
4.5 Network Visualization . . . 89
4.5.1 Introduction . . . 89
4.5.2 Related Work . . . 90
4.5.3 Window-Based Visualization . . . 91
Introduction . . . 91
Related Work . . . 91
4.5.4 Ego Networks - Visualization of Text Nodes . . . 97
Case Study . . . 97
4.5.5 Summary and Contributions . . . 100

5 Incremental Similarity In Text-Streams 101
5.1 Introduction . . . 101
5.2 Incremental Similarity Measures . . . 103
5.2.1 IS-TFIDF for Text Streams . . . 104
5.2.2 IS-TFIDF and ICS Method . . . 104
One Document Streaming (ODS) . . . 105
Several Documents Streaming (SDS) . . . 105
Algorithmic Analysis . . . 106
5.2.3 Case Study and Evaluation . . . 107
Description of the Data . . . 107
Experiments . . . 108
Results . . . 108
5.2.4 Summary and Contributions . . . 109
5.3 Keyword Extraction . . . 111
5.3.1 Contextualization and Clustering . . . 111
Similarity and Hierarchical Clustering . . . 111
Regarding Clustering with Contextualization - Word2Vec . . . 111
Regarding Embedding with FastText . . . 112
5.3.2 Case Study and Evaluation . . . 112
Description of the Data . . . 112
Methodology . . . 113
Results . . . 115
5.3.3 Discussion . . . 118
5.3.4 Summary and Contributions . . . 119

6 Discussion, Future Work and Conclusions **121**
6.1 Future Work 122
6.2 Conclusions 123

Author's Publications **125**

References **127**

A Appendix **139**
Patent Pending 139
Temporal Line of Developments 139

Index **141**

List of Figures

2.1 Original TextRank workflow 25

3.1 Affinity Miner prototype 30
3.2 Keyword descriptors for a selected researcher in the left-hand side panel and for his/her affinity group in the right hand-side 32
3.3 Network of an R&D unit/center 33
3.4 Searching network authors by keyword 33
3.5 Affinity Miner prototype - Folded Networks 34
3.6 Frequencies Counting for Titles Words 40
3.7 Frequencies Counting for Abstracts Words 40

4.1 Network induced by the top-100 subscribers with the highest number of phone calls and corresponding direct connections. This network was generated without running the layout algorithm. 53
4.2 Network induced by the top-100 subscribers with the highest number of phone calls and corresponding direct connections. This network was generated after running the layout algorithm. 53
4.3 Cordeiro Incremental Community Detection (reproduced from Cordeiro et al. (2016)) 61
4.4 INFOMAP algorithm (top) and INFOMAP results with DENSOPT (bottom) . . . 67
4.5 Label Propagation algorithm (top) and label propagation results with DENSOPT (bottom) 68
4.6 Calculated node centralities with edge {(4, 6)} added. Dark grey nodes affected by addition of edges. Light grey nodes centralities need to be calculated due to their neighbourhood with affected nodes. 72
4.7 Results for the Cit-HepTh network (incremental only) 75
4.8 Incremental PageRank workflow - Addition of a Node 81
4.9 Incremental TextRank workflow 82
4.10 Efficiency results with the Reuters News Dataset (for all algorithms) 86
4.11 Visualization of an Ego-Network 92
4.12 Evolving inclusion or exclusion of a given node based on the forgetting factor function. P_n is the number of update periods considering that node n did not receive any connection 93

4.13 Visualization of the call graph using a sliding window approach 95
4.14 Visualization of the call graph using a sliding window approach (2nd version) . . 96
4.15 Ego-Network with forgetting factor 97
4.16 Ego-Network nodes counter 98
4.17 Ego-Network edges counter 99

5.1 Compacting document-term matrix 102
5.2 Bipartite Graph Method 105
5.3 Efficiency results with the Reuters News Dataset (for all algorithms) 109
5.4 Efficiency results with the INESC TEC research Dataset (for all algorithms) . . . 110

List of Tables

3.1 Average Similarity Table with LDA and TF-IDF without clustering 44
3.2 Average Similarity Table with LDA and TF-IDF clustering Exemplars Generalization 45
3.3 Average Similarity Table with stemming and for LDA or TF-IDF, with and without clustering 45

4.1 Examples with real networks from "networkdata" R Package 64
4.2 Number of Communities detected either with INFOMAP or Label Propagation (LP) algorithms, with and without DENSOPT algorithm processing maximization of ADC measure 65
4.3 Modularity of Community division, detected either with INFOMAP or Label Propagation (LP) algorithms, with and without DENSOPT algorithm processing maximization of ADC measure 65
4.4 ADC of Community division, detected either with INFOMAP or Label Propagation (LP) algorithms, with and without DENSOPT algorithm processing maximization of ADC measure 66
4.5 Example of centrality calculation for the network presented in Figure 4.6. 72
4.6 Qualitative measures of similarity with accumulated annotated keywords 88
4.7 Qualitative measures of similarity with current snapshot annotated keywords . . . 88

5.1 Number of groundtruth classes for each dataset 114
5.2 Quality Measurements Comparison - TED DATASET 115
5.3 IS-TFIDF and IS-TFIDF with Word2Vec - TED DATASET 116
5.4 Quality Measurements Comparison - Reuters Dataset 116
5.5 Quality Measurements Comparison - Covid-19 Dataset 117

List of Algorithms

1 Batch Laplace centrality Algorithm 22
2 Algorithm Pseudo-Code for Topic Extraction 41
3 Algorithm Pseudo-Code for Topic Extraction with clustering 41
4 Top-K algorithm for call graphs 52
5 Algorithm Pseudo-Code for Optimization of Community density 63
6 Incremental Laplace centrality Algorithm 74
7 *Top-K* Pseudo-Code for Keyword Extraction 83
8 ITR document Processing Pseudocode (ODS) 84
9 ITR document Processing Pseudocode (SDS) 85
10 Algorithm based on a landmark window model 94
11 Algorithm based on a sliding window model 95
12 IS-TFIDF document Processing Algorithm 106

Glossary

TF-IDF Term Frequency - Inverse Document Frequency.

IS-TFIDF Incremental Similarity TFIDF - a method based on TF-IDF with the use of complex networks theory to incrementally retrieve similarity between elements in the network.

ICS Incremental Cosine Similarity - a method for flux analysis to incrementally update cosine similarity measure.

ITR Incremental TextRank - a method for text flux analysis to rank the incoming text elements in a streaming context.

Top-K a method or any algorithm for flux analysis to rank the incoming elements in a streaming context, sampling the flux by extracting k elements.

ODS One Document Stream - a method for text flux analysis that considers one document only at each time.

SDS Several Documents Stream - a method for text flux analysis that considers several documents at each point of processing.

DENSOPT DENSity OPTimization - a post-processing algorithm to achieve new communities division on complex networks, by increasing ADC measure.

ADC Average Density per Community, refers to the calculation of average density in a complex network context.

LDA Latent Dirichlet Allocation is a generative probabilistic model of a corpus. The fundamental concept is that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words.

SNA Social Networks Analysis refers to the area of studies of complex networks. It complies with a series of metrics and algorithms for the studies of connections between items of one or several types, represented by complex networks and their connections..

LP Label Propagation. A community detection algorithm that is simple and presents one of the lower processing complexity..

NLP Natural Language Processing. It is a branch of artificial intelligence (AI) that enables computers to comprehend, generate, and manipulate human language.

Acknowledgments

The author of this thesis would like to thank all the persons, directly and indirectly, involved in this work. This includes a great number of anonymous peer reviewers that, along the years of research, contributed in a constructive way to improve all the work presented in this thesis. A special thanks go to the editors of several conferences the author participated in and also the Journals the author published in, as they were in some way responsible for the accurate and well-organized peer reviewers' work.

A Special acknowledgment goes to my supervisors. They proved insuperable in their masterful and skillful way of driving me to the material that proved to be very important for developing these works and not disturbing my ideas, even when I had more doubts about them than themselves.

The author also wants to thank his close family, parents, and close friends for their great understanding that the author had a little time to provide them at certain phases of the developments and research.

Last but not least, a special acknowledgment to all the cited authors and their cited authors for all these years of work previous to this work. We are constantly learning from past achievements and milestones. We are here, and we have our tools. We all want to achieve something, sometimes alone, but with thousands of years and individuals as support to achieve something good for humanity and the future.

Abstract

The summarization and organization of document production of an organization in an intuitive and scalable way for massive amounts of data is of great importance in supporting decision-making. The objective of this thesis project is to develop a theoretical and practical study to solve these challenges. The topics of this thesis were conceived after developing a static software prototype enabling to provision of decision support for text documents organized in a network of authors and the corresponding scientific documents. Several advantages were proved by the use of this mentioned prototype. Nonetheless, there were some concerns regarding the prototype's ability to cope with higher dimensional networks and also a massive number of documents. Our case study considers the affinity between authors on a large scale and constantly evolving. The first challenge is to scale up the representation methods of documents of the authors. The second challenge is to capture the temporal development of the organization. Considering this context, we developed and implemented streaming techniques for the characterization of each author and other sub-units of the organization. This gave rise to improved algorithms for community detection and centrality analysis. The communities, sometimes referred to also as affinity groups, are identified by keywords and relevance measures that characterize them. We have tested several developed algorithms that overcome the disadvantages of the original prototype and represent solutions for problems related to text streaming techniques. We solved several associated issues with efficient text stream analysis, using several techniques from pure stream analysis techniques, like, for example, several applications of the Top-K algorithm and developed variants, to evolving complex network techniques. These techniques that served as a base for innovation and contribution with various new algorithms proved to improve the prototype and solve the issues that initially drove us to improve and contribute to several related areas of text analysis and streams. This led to new and improved algorithms in the area of incremental keyword extraction or incremental versions of the calculation of TF-IDF.

keywords: Streaming; Text Mining; Social network Analysis; Social network Visualization

Resumo

A sumarização e organização da produção de documentos de uma organização, de forma intuitiva e escalonável, para grandes quantidades de dados, é de grande importância no apoio à tomada de decisão. Esta tese pretende desenvolver um estudo teórico e prático, para resolver estes desafios. O conteúdo desta dissertação nasceu após a construção de um protótipo de software com algoritmos estaticos, para analisar e fornecer suporte à decisão, a partir de documentos de texto e de uma rede de autores de documentação científica. Várias vantagens foram comprovadas com o uso deste protótipo mencionado. No entanto, havia algumas preocupações em relação à capacidade do protótipo de lidar com redes de dimensões superiores e também com uma grande quantidade de documentos. O estudo de caso de desenvolvimento considera a afinidade entre autores em grande escala e em constante evolução. O primeiro desafio é dimensionar os métodos de representação dos documentos dos autores. O segundo desafio é capturar o desenvolvimento temporal da organização. Considerando este contexto, desenvolvemos e implementamos técnicas de streaming para a caracterização de cada documento e outras subunidades da organização. Para caracterizar, houve interesse por integração em grupos de afinidade identificados por palavras-chave e medidas de relevância. Concluímos este trabalho testando vários algoritmos desenvolvidos, para diminuir a desvantagem do protótipo original e reunindo uma panóplia de soluções para problemas relacionados às técnicas de streaming de texto, considerando uma abordagem em larga escala para a análise correspondente. Foram utilizadas técnicas de recuperação de informação, sendo necessária a análise de redes sociais e streaming de dados. Resolvemos vários problemas associados com análise eficiente de fluxo de texto, usando várias técnicas de análise de fluxo puro, como, por exemplo, vários aplicações do algoritmo Top-K e variantes desenvolvidas. Para a necessidade de utilização de técnicas de evolução de redes complexas, desenvolvemos adaptações incrementais de algoritmos conhecidos. Essas técnicas que serviram de uma base para inovação e contribuição com mais de dez novos algoritmos provou melhorar o prototipo em seus diferentes módulos e resolver os problemas que inicialmente se nos deparavam nas áreas de análise de texto e fluxos. Melhorámos algoritmos na área de Deteção de Comunidades, Análise de Centralidade e conceitos de NLP, por exemplo, extração de palavra-chave ou TF-IDF.

Palavras-Chave: Streaming; Fluxo e Dados Evolutivos; Análise de Texto; Análise de Redes Sociais e Complexas; Visualização de Redes Sociais e Complexas

Chapter 1

Introduction

1.1 Motivation

In recent years, the explosion of social networks like Twitter and Facebook has led to massive amounts of text. Additionally, enterprises generate huge amounts of documentation that need to be analyzed so as to support decision-making. Thus, the existing documents represent a significant resource for better decision support. However, in order to be helpful, the process of analyzing the existing documents needs to be efficient. This can be achieved through :

- better organization of large sets of documents,
- more efficient methods that exploit this organization,
- ability to cope with changing conditions.

Previous work in this area supported by the Affinity Miner prototype has some shortcomings, particularly when considering larger document sets that can change over time.

Our aim is to improve the design to minimize these shortcomings. To do this, our objective was to consider different aspects of the current prototype and examine how these could be improved. These are detailed in the next section.

1.2 Main Objectives

Our work exploits the basic organization of documents in the form of a graph, as in the previous work. We have also adopted the notion of "communities" used in social network analysis (SNA) used to represent groups/clusters of similar documents, as this seems to facilitate the search for the target document. So one of our goals was how to exploit this organization better when we have massive numbers of documents. This led us to define the following goal:

- Can we exploit sampling techniques when processing document graphs?

Communities can be characterized by diverse centrality measures. These are important, as they can be exploited in the search for the relevant document(s). The concept of importance is multi-dimensional. Each relevant dimension, such as reachability, embeddedness, influence, etc. is captured by certain specific measures. Examples of classical measures are degree, betweenness, closeness, and eigenvector centrality. Our aim was to analyze how to compute these measures in a dynamic setting, that is when we have a stream of documents. So, this led us to define the following goals:

- Which measure is useful for the process of determining communities?
- Can we design a method for some centrality measures that is incremental, and suitable for streaming settings?

When dealing with documents, keywords play also an important role, as they can be used to characterize documents, calculate the similarity between documents and facilitate the search for some documents of interest. So, given this, we define the following goal:

- Can we design a method of keyword extraction that is incremental?

Visualization of networks is important too, as it helps the user to navigate the network and identify regions of interest. However, care must be taken, when visualizing very large networks, as these can be cluttered and so do not provide much help. So, this leads us to define the following goal:

- Given a document stream, can we design networks that include the focal nodes, referred to as Ego-Networks?

1.3 Main Contributions

The work described in this thesis led to various contributions that are briefly described here. First, we have improved the scalability of streaming methods with recourse to dimension reduction methods and data streaming methods in Chapters 4 and 5 (Sarmento et al., 2021, 2015a). This way, the retrieval of data items from the data repository is facilitated, as the system explores the information about affinity among the individual documents. Data that involves different documents has a dynamic and evolving nature, so the challenge of dealing with this aspect is significant. We have made various proposals that provide answers to this challenge. Thus, all the algorithms and the improvements in efficiency reported in this document are in part or entirely related to the topic of complex networks and their state-of-the-art algorithms. Our goal was to improve the grouping/community detection methods in Chapter 4 (Sarmento and Lemos, 2020). We made this by considering an organization that is evolving and redesigning the methods of grouping discovery/-community detection. With these improved algorithms, the user is able to inspect evolving topics and groups/communities from given sets of documents. There are other methods that were improved and upgraded to stream setting, such as keyword extraction in Chapter 5, (Sarmento et al., 2018a), network visualization (Sarmento et al., 2015b, 2014a) in Chapter 4, centrality measures (Sarmento et al., 2017) in Chapter 4, and TF-IDF measure in Chapter 5.

1.4 Outline

This section introduces the topics addressed in the research project throughout several chapters. Therefore, this document is organized in the following order of chapters:

- In chapter 2, we briefly introduce the basic concepts in the areas of research.
- In chapter 3, we present state-of-the-art and also a description of previous work and the Affinity Miner prototype.
- In chapter 4, we introduce some concepts of streaming and sampling of networks. We also introduce some of the routes we considered for the research at the time of these studies.
  - In section 4.1, we discuss the methodology for the sampling of networks.
  - In section 4.2, we deal with the community analysis problem, and we state our contributions regarding these areas of research.
  - In section 4.3, we deal with the centrality of nodes in evolving complex networks, and we state our contributions regarding these areas of research. This area is important to reach for influential or representative nodes in evolving networks of nodes, for example in text document graphs.
  - In section 4.4, we describe the methodology and contributions for the graph-based keyword representation and extraction for and from documents in the stream of data.
  - In section 4.5, we deal with the visualization/organization of nodes in evolving complex networks, and we state our contributions regarding these areas of research. This area is important to reach for a clearer organization in the user interface, and a better decision support system, for example in visually inspecting text document graphs.
- In chapter 5, we present the important milestones accomplished to reach several solutions, for the analysis and retrieval of similarity between documents in a evolving flux of text documents.
  - In section 5.1, we introduce the chapter where we explain the methods and contributions in the development of techniques to deal with incremental similarity of documents in the stream.
  - In section 5.2, we explain the use of graphs and bipartite graphs in the calculations of incremental similarity between changing documents in the stream.
  - In section 5.3, we discuss about keyword extraction and its parallelism with feature selection. In this research, we contributed several new ways of performing keyword extraction in streams of text documents.
- In chapter 6, we present a discussion about the work provided in the research, opening some discussion also about future work possible. We also state the conclusions we took from

this research. we finish the document with a summary of what we accomplished with this research.

- In section 6.1, we deal with the possibilities of directions for future work. We consider the work developed at the particular moment in the research. We state possible further research, by using our work and solutions.
- In section 6.2, we write the global conclusions of this document.

Chapter 2

Base Concepts and Related Work

In this chapter, we discuss previous work on the subjects of study. Thus, we briefly introduce basic concepts. The relevant aspects of some existing solutions that provide a starting point for the development of improvements in this research area will be presented, with more detail, also further in this document in chapter 3. The reader will be introduced to fundamental concepts of static algorithms in the stated areas in the following subsections. Further in the document, in other chapters, this document will proceed with introductions to evolving and flux solutions for these same areas.

This thesis builds on knowledge in three domains: Information Retrieval, analysis of evolving Social Networks, and Streaming Data Analysis. These significant research areas will serve as base concepts for the development of this document. There are several active researchers in several research groups in these areas. Additionally, there are several important events like conferences in the area and several high-ranked Journals.

A summary of these research groups, researchers, conferences, and journals is given in the Appendix.

The reader should be aware of key concepts related to clustering in the typical information retrieval fields of studies. In the social and complex evolving networks, the reader should be aware of visualization techniques, community detection, and centrality analysis. The reader will also be introduced to the keyword extraction area within the text analysis area.

2.1 Text documents

In a paper by Trigo and Brazdil (2014), the goal is to reveal affinities between researchers that are not identified through co-authorship. Many different techniques were used; Web mining, text data analysis, social network analysis, community detection, and characterization through automatically retrieved keywords. The methodology is applied to a group of researchers from INESC TEC Porto. Static data is analyzed through visualization and regarding the affinity between researchers. The data taken from the researchers' publications are analyzed statically, and a representative graph of the affinity network is generated. This allows to intuitively detect patterns and groups of people, even without a co-authorship, and conduct research in similar areas. In computer text processing, it is common to use the *bag of words* representation as in Feldman and Sanger (2006), accompanied by the usual pre-processing, including removal of numbers, irrelevant words ("stop-words"), punctuation, and other spurious elements. Then, the list of documents is usually transformed into a *document-term vector* representation with TF-IDF weighting. The vector representation is used to generate the cosine similarity matrix. This matrix can be visualized as a graph and is used for the subsequent processing (Iacobucci, 1994). The affinity network allows calculating certain important measures of researchers within their community and in the context of the different communities. *Betweenness centrality* of intermediation and *Eigenvectors* were used as published in Wasserman and Faust (1994b). After transforming the similarity matrix in a graphical format, a community detection algorithm, "Walktrap" (Pons and Latapy, 2005), is used to extract the affinity groups.

From our point of view, other methods may be applicable, such as using sliding windows to fit the dynamic nature of the data stream (Oliveira et al., 2014) or the development of a version of this previous work of Trigo et al., but this time with a dynamic architecture. For example, we could use an incremental version of the community detection algorithm such as the adapted Louvain method used in Sarmento et al. (2015a). Additionally, to add streaming features to these previous works, there is some interest in improving the clusterization or summarization of the authors' document production.

2.2 Clustering Evaluation

According to Tan et al. (2005) evaluation or validation of clusters or groups must be part of the whole cluster analysis. Almost all clustering algorithms identify some clusters in a data set, even though they do not present a natural grouping structure. However, understanding whether there is a non-random data structure is just one aspect of cluster validation. The authors present a list of essential tasks for this validation:

1. It is necessary to determine the trend of clustering in a set of data, that is, to verify whether if there is a non-random structure in the data.
2. It is necessary to determine the correct number of clusters.
3. We should assess whether the cluster analysis results fit the data to quantify the degree of fit without reference to external information.
4. Compare the results of a cluster analysis results to validation results.
5. Examine different sets of groups to determine which is the best.

According to Tan et al. (2005), several numerical measures allow the validation tasks mentioned above. However, it is necessary to note that some steps can be pretty limited in their applicability.

For the authors, the most used measures are classified into three types: unsupervised, supervised, and relative. For unsupervised validation measures - measuring the quality of a clustering structure is done without resorting to external information as in Tan et al. (2005). They can be divided into group cohesion measures that determine how closely the objects are in a group and groups separation measures that determine how they are separated from others.

The combined measure can include cohesion or separation measures or a combination of both. The value of w_i weights depends on the adopted validity measure. If one chooses cohesion, it is preferable to use higher values. On the contrary, for separation, lower values are preferred.

There have been several suggestions for a measure of similarity between two clusterings. Such a measure can compare how well different data clustering algorithms perform on a set of data. These measures are usually tied to the criterion considered in assessing the quality of a clustering method.

2.3 External Clustering Validation Measures

In an external evaluation, clustering results are evaluated based on data not used for clusterings, such as known class labels and external benchmarks. Such references consist of pre-classified items, which humans often create. Thus, the benchmark sets can be considered a gold standard for evaluation. These assessment methods measure how close the clustering is to the predetermined reference classes. However, it has recently been discussed whether this is adequate for real data, or only on synthetic data sets with factual ground truth, since classes can contain internal structure, the attributes present may not allow separation of clusters, or the classes may include anomalies as exposed in Färber et al. (2010). Additionally, from a knowledge discovery point of view, the reproduction of known knowledge may not necessarily be the intended result as explained in Färber et al. (2010). In the particular scenario of constrained clustering, where meta information (such as class labels) is already used in the clustering process, the hold-out of information for evaluation purposes is non-trivial.

Some measures are adapted from variants used to evaluate classification tasks. In place of counting the number of times a class was correctly assigned to a single data point (known as true positives), such pair counting metrics assess whether each pair of data points that is genuinely in the same cluster is predicted to be in the same cluster.

Rand measure

The Rand index (Rand, 1971) computes how similar the clusters (returned by the clustering algorithm) are to the benchmark classifications. One can also view the Rand index as a measure of the percentage of correct decisions made by the algorithm. It can be computed using the following formula:

$$RI = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.1}$$

where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives. One issue with the Rand index is that false positives and false negatives are equally weighted. This may be an undesirable characteristic for some clustering applications. The F-measure addresses this concern, as does the chance-corrected adjusted Rand index.

F-measure

The F-measure can be used to balance the contribution of false negatives by weighting recall through a parameter $\beta \geq 0$. Let precision and recall be defined as follows:

$$P = \frac{TP}{TP + FP} \tag{2.2}$$

$$R = \frac{TP}{TP+FN} \tag{2.3}$$

where P is the precision rate, and R is the recall rate. We can calculate the F-measure by using the following formula (Manning et al., 2008):

$$F_\beta = \frac{(\beta^2+1)\cdot P \cdot R}{\beta^2 \cdot P + R} \tag{2.4}$$

Notice that when $\beta = 0$, $F_0 = P$. In other words, recall has no impact on the F-measure when $\beta = 0$, and increasing β allocates an increasing amount of weight to recall in the final F-measure.

Jaccard index

The Jaccard index is used to quantify the similarity between two datasets. The Jaccard index takes on a value between 0 and 1. An index of 1 means that the two datasets are identical, and 0 indicates that the datasets have no common elements. The Jaccard index is defined by the following formula:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{TP}{TP+FP+FN} \tag{2.5}$$

This is simply the number of unique elements common to both sets divided by the total number of unique elements.

Fowlkes–Mallows index

The *Fowlkes-Mallows* index in Fowlkes (1983) computes the similarity between the clusters returned by the clustering algorithm and the benchmark classifications. The higher the value of the Fowlkes-Mallows index, the more similar the clusters and the benchmark classifications are. It can be computed using the following formula:

$$FM = \sqrt{\frac{TP}{TP+FP} \cdot \frac{TP}{TP+FN}} \tag{2.6}$$

where TP is the number of true positives, FP is the number of false positives, and FN is the number of false negatives. The FM index is the geometric mean of the precision and recall P and R, while the F-measure is their harmonic mean (Hubert and Arabie, 1985). Moreover, precision and recall are also known as Wallace's indices B^I and B^{II}.

Mutual information is an information-theoretic measure of how much information is shared between a clustering and a ground-truth classification that can detect a non-linear similarity between two clusterings. Adjusted mutual information is the corrected-for-chance variant, which has a reduced bias for varying cluster numbers. A confusion matrix can be used to quickly visualize

the results of a classification (or clustering) algorithm. It shows how different a cluster is from the gold standard cluster.

Normalized Mutual Information

High purity clustering is easy to achieve when the number of clusters is significant - in particular, purity is one if each document gets its own cluster. Thus, we cannot use purity to trade off the quality of the clustering against the number of clusters.

A measure that allows us to make this tradeoff is normalized mutual information or NMI :

$$\mathrm{NMI}(\Omega,\mathbb{C}) = \frac{I(\Omega;\mathbb{C})}{[H(\Omega)+H(\mathbb{C})]/2} \tag{2.7}$$

I is mutual information:

$$\mathrm{I}(\Omega,\mathbb{C}) = \sum_k \sum_j P(\omega_k \cap c_j) \log \frac{P(\omega_k \cap c_j)}{P(\omega_k)P(c_j)} \tag{2.8}$$

$$= \sum_k \sum_j \frac{|\omega_k \cap c_j|}{N} \log \frac{N|\omega_k \cap c_j|}{|\omega_k||c_j|} \tag{2.9}$$

where $P(\omega_k)$, $P(c_j)$, and $P(\omega_k \cap c_j)$ are the probabilities of a document being in cluster ω_k, class c_j, and in the intersection of ω_k and c_j, respectively. Equation 2.9 is equivalent to Equation 2.8 for maximum likelihood estimates of the probabilities (i.e., the estimate of each probability is the corresponding relative frequency).

H is entropy defined as :

$$\mathrm{H}(\Omega) = -\sum_k P(\omega_k) \log P(\omega_k) = -\sum_k \frac{|\omega_k|}{N} \log \frac{|\omega_k|}{N} \tag{2.10}$$

where, again, the second equation is based on maximum likelihood estimates of the probabilities.

$I(\Omega;\mathbb{C})$ in Equation 2.8 measures the amount of information by which our knowledge about the classes increases when we are told what the clusters are. The minimum of $I(\Omega;\mathbb{C})$ is 0 if the clustering is random on class membership. In that case, knowing that a document is in a particular cluster does not give us any new information about its class. Maximum mutual information is reached for a clustering Ω_{exact} that perfectly recreates the classes - but also if clusters in Ω_{exact} are further subdivided into smaller clusters. In particular, a clustering with $K = N$ one-document clusters has maximum MI. So MI has the same problem as purity: it does not penalize large cardinalities and thus does not formalize our bias that, other things being equal, fewer clusters are better.

The normalization by the denominator $[H(\Omega)+H(\mathbb{C})]/2$ in Equation 2.7 fixes this problem since entropy tends to increase with the number of clusters. For example, $H(\Omega)$ reaches its maximum $\log N$ for $K = N$, ensuring that NMI is low for $K = N$. Because NMI is normalized, we

can use it to compare clusterings with different numbers of clusters. The particular form of the denominator is chosen because $[H(\Omega) + H(\mathbb{C})]/2$ is a tight upper bound on $I(\Omega;\mathbb{C})$. Thus, NMI is always a number between 0 and 1.

2.4 Sampling Large Static Networks

Large-scale network sampling has recently become a hot topic in network analysis, and only a few methods have been proposed so far. The most common approaches for static network sampling are the *random sampling* and the *snowball sampling*.

In the *snowball sampling* (Goodman, 1961), a starting node is chosen. Then, the connections in the 1^{st}, 2^{nd}, to n, neighborhood-order of this starting node are extracted until the network reaches the desirable size. While easy to implement, this approach has known problems: it is biased towards the region of the network to where the starting node belongs and potentially misses important network properties. Nevertheless, it is one of the most common sampling approaches.

On the other hand, *random sampling* (Granovetter, 1976), randomly selects a user-defined percentage of nodes and keeps all the corresponding edges. Alternatively, the sampling can be performed on the edges by randomly selecting a user-defined percentage of edges and keeping the corresponding nodes. The main problem with this approach is that random edge sampling is biased towards high-degree nodes. In contrast, the random node sampling may be unable to generate a representative sample since the structural properties of the sample may not reflect the ones observed on the original network. Despite these drawbacks, random sampling is easy to understand and implement.

Therefore, the task must be to generate a sample so that the sampled network is representative of the original one in terms of structural properties. A primary question is related to the definition of *representative sample*. Existing work considers measures such as similarity in degree distributions and clustering coefficients (Hubler et al., 2008; Leskovec and Faloutsos, 2006).

Leskovec and Faloutsos (2006) introduce a great variety of graph sampling algorithms. They conclude that methods combining random node selection and some vicinity exploration generate the best network samples. They show that a 15% sample is usually enough to match the properties of the original graph and that no list of network properties serving as a basis for sampling evaluation will ever be perfect.

Two common strategies for sampling are *random sampling* and *snowball sampling*. In *snowball sampling* a starting node is selected. The network is built from that node, starting on its 1^{st} order connections, moving to the 2^{nd} order connections, 3^{rd} order connection, and so on, until the network reaches the right size for analysis. This approach is easy to implement but has some pitfalls. It is biased toward the part of the network sampled and may miss other features. Nevertheless, it is one of the most common sampling approaches. The *random sampling*, randomly selects a certain percentage of nodes and keeps all edges between them. In an alternative approach, it randomly selects a certain percentage of edges and keeps all mentioned nodes. The main problem with this method is that edge sampling is biased towards high degree nodes, while node sampling might lose some network structural features.

2.5 Visualization

The definition of large-scale networks regarding number of nodes or edges diverges. Publications may consider a large-scale network ranging from dozens of thousands of nodes to millions of nodes and billions of edges. The main goal of any graph visualization technique is to be visually understandable. It is also desirable that the information is represented clearly and objectively to convey knowledge to the viewer. To achieve this goal, two types of graph representation, node-link, and matrix graph representations like in Lee et al. (2006) may be used. visualization readability is related to the network size (number of nodes) and density (average number of edges per node). It is known that node-link representation has low performance with dense networks and requires aggregation methods reducing density to increase visual comprehensibility of the output. matrix representation is usually combined with hierarchical aggregation like in Abello and van Ham (2004). Hierarchical clustering implies grouping the nodes but not their ordering. The main goal of this representation type is to have a fast clustering algorithm and meaningful clusters. matrix representation methods may also rely on reordering rows and columns in the representation matrix instead of just clustering the nodes. This ordered matrix representation might enhance the structure visualization because the data is more than simply clustered. The main drawback of this solution is that it is unfeasible for networks of millions of nodes that need many computations for reordering the matrix. More recently, Elmqvist et al. (2008) introduced a fast reordering mechanism, data aggregations, and GPU-accelerated rendering to deliver higher scalability solutions. Other solutions rely on controlling the visual density of the graph view and restricting the clustering overlap probability to low levels as in Shi et al. (2009). Moreover, a new probability-based network metric was introduced by Ham et al. (2009) to identify potentially exciting or anomalous patterns in the networks.

2.6 Centrality Measures

Centrality measures in SNA have been an essential area of research as they help us identify the relevant nodes. Researchers have invested much effort to develop algorithms that could efficiently calculate the centrality measures of nodes in networks.

Betweenness Betweenness is a classical centrality measure that quantifies the importance of a network element (node or edge) based on the frequency of its occurrence in the shortest paths between all possible pairs of nodes in the network. The intuitive idea behind this measure is to identify graph elements that act as bridges, i.e., which connect dense regions of the network and without which the information would not pass from one of these regions to the other. An edge with high betweenness is likely to act as a bridge between dense graph regions and, thus, occurs in many shortest paths. Nodes with high betweenness are usually located at the ends of these edges. These nodes occupy a strategic position in the network, which allows them to control the information transfer between different network regions, either by blocking the information between them or by accessing it before other nodes belonging to their region.

Node (or edge) betweenness is computed as the fraction of shortest paths that go through a given node (or edge) among all shortest paths in the network. This is a global centrality measure since it requires complete information about the network to compute all-pairs shortest paths. Since it is based on the computation of shortest paths for the whole network, betweenness is computationally demanding. The best-known algorithm for computing this measure in static unweighted graphs was proposed by Brandes (2001b). This algorithm runs in $O(nm)$ time, being n the number of nodes/vertices and m the number of links/edges, and has a space complexity of $O(n^2 + nm)$, which is prohibitive for networks with millions of nodes and billions of links. Given this, it is necessary to develop new algorithms that avoid the full recomputation of betweenness every time a new edge or node is added to, or removed from, the network. Incremental algorithms offer a solution to this problem since they can handle large-scale data and can adapt to incremental changes in evolving networks. This is achieved by performing early pruning and updating only the network regions affected by the changes.

Recently, several solutions were proposed to compute node and/or edge betweenness centrality in streaming networks. Lee et al. (2012) introduced QUBE, an incremental algorithm that relies on the decomposition of the graph into biconnected components. The algorithm's performance is strongly associated with the size of the components, which is usually very large in real-world applications. Consequently, it suffers from scalability and efficiency problems, which are of utmost importance in streaming settings. In the same year, Green et al. (2012) proposed an exact algorithm, which extends Brandes's approach (Brandes, 2001b), to compute node betweenness in unweighted streaming graphs. The idea is to preserve information from prior computations of betweenness values and the needed data structures and update only the values and data structures directly affected by the changes in the network. However, the algorithm has some drawbacks: it only supports one type of change (insertion of new edges), and has the same space complexity as

Brandes's static algorithm, which is $O(n^2 + nm)$. Hence, the algorithm is not entirely suited to handle large-scale and dynamic networks. Kas et al. (2013c) propose a slightly better solution. They present an incremental algorithm for dynamic maintenance of node betweenness centrality values in rapidly growing networks, using as a building block the dynamic all-pairs shortest path algorithm introduced by Ramalingam and Reps (1996b). Similarly to the research documented in Green et al. (2012), the technique of Kas et al. is also based on keeping in-memory information from previous computations but using data structures that are faster to update (e.g., shortest distances and number of shortest paths). Although the computational complexity can be lower than the one obtained by Green et al. (2012), the space complexity is similar, turning it prohibitive for very large graphs. More recently, Kourtellis et al. (2014) proposed an incremental and scalable algorithm for online computing of both node and edge betweenness centralities in very large dynamic networks. Besides being adapted to fully dynamic networks, where nodes and links are added and removed over time, these authors propose an algorithmic framework that can be efficiently used for real-world deployment (e.g., for identifying strategic subscribers in call graphs) since the algorithm allows for out-of-core implementation and is tailored for modern parallel stream processing engines.

A different approach from the above mentioned was introduced by Kim and Anderson (2012). They presented the time-ordered graph model, which converts a dynamic network into a static network with directed flows and proposes temporal centrality measures to extract information from the graph. These measures are simple extensions of the classical measures to this specific type of dynamic model. However, this method was devised for dynamic networks which evolve in non-streaming scenarios, thus not being fully suitable for the online analysis of call graphs.

Closeness Closeness is another classical SNA measure that quantifies the importance of a node based on its ability to reach other nodes in a network through shortest paths. The higher the closeness, the less the cost for a node to reach the rest of the network. For instance, in a call graph, the subscriber with the highest closeness can quickly spread information to other network nodes as long as the nodes belong to the same connected component.

Closeness is computed as the inverse of the sum of the shortest path distances between a node i and the remaining $n-1$ nodes in a network of size n. Similarly to betweenness, a few incremental algorithms were proposed to compute the closeness centrality in large dynamic graphs. Kas et al. (2013b) proposed an algorithm for the fast computation of closeness in large-scale networks. Their technique supports the efficient computation of all-pairs shortest paths and is suited to dynamic networks since it handles adding, removing, and modifying nodes and links. This algorithm is similar to the one proposed by the same authors in Kas et al. (2013c). A more recent work by Khopkar et al. (2014) presents a partially dynamic and incremental closeness algorithm that runs in $O(n^2)$.

Laplacian centrality In Qi et al. (2012), the authors introduced a novel centrality measure. The authors stress that their new measure based on the Laplacian energy of a node would outperform

Betweenness and Closeness centrality regarding complexity. Therefore, the authors achieved a more efficient way of retrieving centrality measurements in a network. In their paper, authors also compare their new measure with Betweenness centrality and Closeness centrality and prove the reliability of their measure by providing similar results per node. The definition of the Laplacian centrality of a node u_i is based on the Laplacian Energy of the graph G obtained after removing the node u_i from the graph, and by following the expression (Qi et al., 2012):

$$C_L(u_i, G) = \frac{(\Delta E)_i}{E_L(G)} = \frac{E_L(G) - E_L(G_i)}{E_L(G)} \tag{2.11}$$

From equation 2.11, we can expect a Laplacian centrality value between 0 and 1. These values increase correspondingly to the increased centrality of the node. Nonetheless, since the definition is in itself a normalization, it is a normal procedure to present the non-normalized values $(\Delta E)_i$ for each node. This way, we can achieve a faster-ranking computation without considering the total graph energy $E_L(G)$. The following algorithm (Algorithm 1) presents an implementation of the Laplacian centrality measure as described in Qi et al. (2012, 2013) for unweighted networks. Presented pseudo-code is based on the implementation of the Laplacian centrality as made available by Wheeler (2015) and Contributors (2021).

Algorithm 1 Batch Laplace centrality Algorithm

$V \leftarrow \{u_1, u_2, .., u_v\}$, $E \leftarrow \{(i_1, j_1), (i_2, j_2), .., (i_e, j_e)\}$
procedure LAPCENT($G \leftarrow (V, E)$)
    $\mathcal{C}_{entralities} \leftarrow \{\}$
    $\mathcal{D}_{egrees} \leftarrow G.degrees()$
    $\mathcal{V}_s \leftarrow G.nodes()$
    for each v **in** $\mathcal{V}_s$ **do**
        $\mathcal{N}_{eighbors} \leftarrow G.neighbors()$
        $loc \leftarrow \mathcal{D}_{egrees}[v]$
        $nei \leftarrow 2. \sum_{i=1}^{\mathcal{N}_{eighbors}} \mathcal{D}_{egrees}[i]$
        $\mathcal{C}_{entralities}[v] \leftarrow (loc^2 + loc + nei)$
    end for
    return $\mathcal{C}_{entralities}$, $\mid \mathcal{V}_s \mid$
end procedure
procedure MAIN
    $\mathcal{D}_{ataset} \leftarrow \{G_0, G_1, ..., G_n\}$
    for each *snapshot* **in** $\mathcal{D}_{ataset}$ **do** ▷ calculation in the full network
        $\mathcal{C}_{entralities}, \mathcal{N}_{um}\mathcal{C}_{entralities} \leftarrow$ LAPCENT($snapshot$)
    end for
end procedure

By observation of the `LapCent()` method, for each loop of the algorithm, it can be concluded that the centrality parameter is a function of the local degree plus the degree's of the neighbors (with different weights for each). Therefore, the metric is not a global measure. The local degree and the first order neighbors degree are needed to calculate the metric for unweighted networks. The `Main()` method shows the pseudo-code required for performing batch Laplacian centrality calculation on an evolving network. The algorithm will require performing a full calculation for

each one of the snapshots $\{G_0, G_1, ..., G_n\}$ included in the dataset $\mathcal{D}_{ataset}$. Notice that no Laplacian centrality data or network data is shared between snapshots. We will explore this inefficiency of the algorithm in Section 4.3.3 when proposing the incremental version of the algorithm.

2.7 Feature/Keyword Selection

This section presents related work regarding automatic keyword extraction systems, particularly those based on graph approaches. Automatic keyword extraction is a concept that relies on the process of selection of words or phrases that, without human intervention, provide a clear idea about the core area of a document or set of documents. Those implementations can be classified into four classes according to Siddiqi and Sharan (2015), Rule-Based Linguistic approaches, Statistical approaches, Machine Learning approaches, and Domain-specific approaches. These approaches have their advantages and disadvantages when compared to one another. Inside each class, the approach can also be divided into supervised and unsupervised methods according to Siddiqi and Sharan (2015). The use of human tasks or contributions in unsupervised approaches is minimal. Otherwise, by using supervised approaches, researchers consider previous annotations by humans, and their participation in finding quality keyword annotations from texts is a significant concern in these methods. According to Siddiqi and Sharan (2015) survey, there are several unsupervised approaches of interest.

In Bracewell et al. (2005), the authors extract noun phrases from a document and then cluster the terms which have the same noun term. Then, the author's proposal consists of ranking clusters, taking into account term and noun phrase frequencies. Finally, only the Top-ranked clusters are selected as keyphrases. In Liu et al. (2009), researchers proposed to extract keyphrases by utilizing clustering techniques. According to these authors, they ensure that these keyphrases semantically cover the document. In Rose et al. (2010), the authors described Rapid Automatic Keyword Extraction (RAKE), a method for extracting keywords from individual documents. RAKE is based on the observation that keywords frequently contain multiple words, but they rarely include punctuation or stop words; therefore, any words having minimal content or information. This new method is domain and language-independent. The document by Gazendam et al. (2010) describes the extraction and ranking of keywords with a restricted vocabulary with the help of a thesaurus. For ranking words, it uses a weighting scheme called tf-rr, which uses both the term frequency and the number of thesaurus relations realized between the thesaurus terms found in the specific document. This approach does not need any training from a reference corpus. Finally, we can also account for two graph-based methods. The authors Mihalcea and Tarau (2004a) proposed TextRank, a new graph-based ranking model for graphs extracted from texts. Thus, to rank keywords based on the co-occurrence links between words, Mihalcea et al. use the concept of PageRank between words to extract important keyphrases with linked words that have a higher weight in the word graph. In Litvak et al. (2011), researchers proposed DegExt, a graph-based, cross-lingual keyphrase extractor. The authors proposed a graph representation based on a simple graph-based syntactic representation of a document and enhanced the model by taking into account structural document features. In this work, we choose to improve TextRank, due to its considerable usefulness in this area and relatively good results for keyword extraction. Additionally, by being a graph-based approach, it is reasonable to think that an incremental approach can significantly improve running time or efficiency in a streaming context. By considering small graph updates

Figure 2.1. Original TextRank workflow

instead of all the graph in each update, we will try parallelism with other works in the graphs area, where researchers consider only affected nodes in each update. The improvements in graph processing efficiency are those authors' main contributions.

TextRank As previously stated, Mihalcea and Tarau (2004a) developed TextRank as a solution to obtain keywords automatically from the text. Figure 2.1 shows the workflow of the original TextRank algorithm. The text corpus processing starts with pre-processing the text and removing stop words, numbers, and punctuation. Then, the document goes through the annotation process where the remaining single words are categorized, for example, nouns, verbs, or adjectives, among others. This method is called Part-of-Speech tagging (POS tagging). According to the authors, only a few of these annotated words are essential. The authors studied which group of words delivered the best results, and they concluded that the best automatic keyphrases were obtained with nouns and adjectives. Then, with these filtered words, a graph-based approach is used. Each word is considered a graph node, and the order determines the connections of words in this directed graph they appear in the text. The weight of these links is obtained by counting the number of times these pairs of words occur in the text corpus. The next phase of the algorithm regards selecting the words of high importance. This is done with the use of the PageRank algorithm by Page et al. (1998). The words with high PageRank values are selected as potential keywords. Finally, the keyphrases are obtained with a post-processing stage. This stage involves using a sliding window evolving through the initial text to assess the order of words contained in the keyphrases or keywords. This step considers punctuation and other structural features of the document to retrieve reasonable keyphrases. TextRank has been widely praised as a consistent method of automatically retrieving keywords from the text. Inclusively, it has been used in prototypes of decision support systems as narrated by Brazdil et al. (2015).

2.8 Text Sparsity and Sparse Matrices

Sparse matrix In numerical analysis, a sparse matrix is a matrix in which most of the elements are zero. By contrast, if most elements are non-zero, the matrix is considered dense. The number of zero-valued elements divided by the total number of elements (e.g., $m \times n$ for an $m \times n$ matrix) is called the sparsity of the matrix (which is equal to 1 minus the density of the matrix).

Conceptually, sparsity corresponds to systems that are loosely coupled. The concept of sparsity is useful in combinatorics and application areas such as network theory, which has a low density of significant data or connections.

Large sparse matrices often appear in scientific or engineering applications when solving partial differential equations. In our case study, usually, TF-IDF matrices appear with sparsity higher than 90%.

When storing and manipulating sparse matrices on a computer, it is beneficial and often necessary to use specialized algorithms and data structures that take advantage of the sparse structure of the matrix. Operations using standard dense-matrix structures and algorithms are slow and inefficient when applied to large sparse matrices as processing and memory are wasted on the zeroes. Sparse data is by nature more easily compressed and thus requires significantly less storage. Some very large sparse matrices are infeasible to manipulate using standard dense-matrix algorithms.

Storing a sparse matrix A matrix is typically stored as a two-dimensional array. Each entry in the array represents an element $a_{i,j}$ of the matrix and is accessed by the two indices i and j. Conventionally, i is the row index, numbered from top to bottom, and j is the column index, numbered from left to right. For an $m \times n$ matrix, the amount of memory required to store the matrix in this format is proportional to $m \times n$ (disregarding the fact that the dimensions of the matrix also need to be stored).

In the case of a sparse matrix, substantial memory requirement reductions can be realized by storing only the non-zero entries. Depending on the number and distribution of the non-zero entries, different data structures can be used and yield huge savings in memory compared to the basic approach. The trade-off is that accessing the individual elements becomes more complex, and additional structures are needed to recover the original matrix unambiguously.

Formats can be divided into two groups:

Those that support efficient modification, such as DOK (Dictionary of keys), LIL (List of lists), or COO (Coordinate list). These are typically used to construct the matrices. Those that support efficient access and matrix operations, such as CSR (Compressed Sparse Row) or CSC (Compressed Sparse Column).

List of lists (LIL)

LIL stores one list per row, with each entry containing the column index and the value. Typically, these entries are kept sorted by column index for faster lookup. This is another format good for incremental matrix construction.

Chapter 3

Affinity Miner Prototype, Concept and Characteristics

In Trigo. et al. (2015), a tool to analyze the document production and visualize affinity groups of researchers is proposed, the "Affinity Miner" prototype.

In Figure 3.1, the affinity between authors is translated in a social network. Additionally, this social network is divided into communities of researchers represented by nodes in different colors.

Each document includes the list of publication titles of a particular author/researcher in this practical application. This chapter presents the main steps to uncover the unknown information regarding affinities and their validation. The method involves the following main steps:

- Identify institutions and obtain researchers' names;
- Use web/text mining to processing researchers' publications;
- Discovering potential communities linked by affinities;
- Identification of important nodes (researchers) in the graph;
- Characterization of nodes using keywords

Additionally, we can resume the Affinity Miner workflow like the following complex mixture of different algorithms in this pseudo-algorithm:

Figure 3.1. Affinity Miner prototype

Affinity Miner Pseudo-algorithm

1. document production is aggregated by author (e.g., titles, abstracts are extracted)
2. Pre-processes text (removes stopwords, stemming, etc.)
3. Builds a document-term matrix where rows are authors and columns are tokens/-words
4. Performs cosine similarity of the TF-IDF matrix among documents (=authors)
5. Builds a graph from the obtained similarities (weighted edges) between authors (nodes).
6. Proceeds with affinity group detection in the author's network with random walk algorithm
7. Calculates centrality measures in the author's network

In preliminary work (Brazdil et al., 2015), the steps to retrieve and process the information used in the prototype are described. The first step is selecting the institutions and obtaining the researchers' names on their web pages. This information can be extracted by an expression in XPath query language to obtain their names from the website. Each researcher's name can be used to search through the chosen bibliographic database, such as DBLP, which enables direct access to each researcher's list of publications. The matching between author's names in the institution's website and bibliographic databases bring name ambiguity problem that was addressed before by Bugla (2009). The Authenticus team kindly provided the web mining results of publications related to each author.

Publications titles are extracted from plain text files/documents representing a particular author. The text files are retrieved and pre-processed in the usual manner. We use bag-of-words (BoW), and vector representation (Feldman and Sanger, 2007), but perform usual pre-processing, including removal of numbers, stop-words, punctuation, and other spurious elements. After this task, the list of documents is transformed into a document-term vector representation with TF-IDF weighting. The vector representation is used to generate the cosine similarity matrix. This matrix can be visualized in a graph and is used as the basis for further processing.

The affinity network enables the calculation of some measures of the importance of the researchers within their affinity group and in the context of different affinity groups. Two centrality measures (Iacobucci, 1994) were extracted from the graph. The *betweenness centrality* indicates the number of times a vertex joins two other vertices on the shortest path. The *eigenvector centrality* shows the importance of vertices that connect to a given vertex and capture the strength of the vertex within the network.

For the affinity group extraction task, the *Walktrap* algorithm (Pons and Latapy, 2005) was selected. This technique finds densely connected sub-graphs, also referred to as communities, through random walks. It assumes that short random walks tend to stay in the same community.

Furthermore, TextRank algorithm (Mihalcea and Tarau, 2004b) to extract keywords from the existing text (publication titles) was used.

The validation process compares the rankings obtained from dissimilarity matrices and rankings obtained via questionnaires for a research institute having slightly more than 30 researchers and collaborators, including Ph.D. students.

3.1 Affinity Miner Case Study

With this prototype, similarities of about 120 researchers of seven units of a Portuguese R&D institution (INESC-TEC, 2015) and their 4153 publications were analyzed.

In the application's main screen, the prototype presents the affinity network of all the R&D units/centers on the canvas. Further exploration and browsing features are presented below.

The prototype shows the similarities between researchers and their affinity groups. There are several ways to infer the importance of a particular author/node. The most immediate visual clues are the node size, proportional to the author's publications and the number of connections.

The other graphic elements for this relevance assessment are *betweenness* and *eigenvector centralities* boxplots that show the relative position of the author concerning the others.

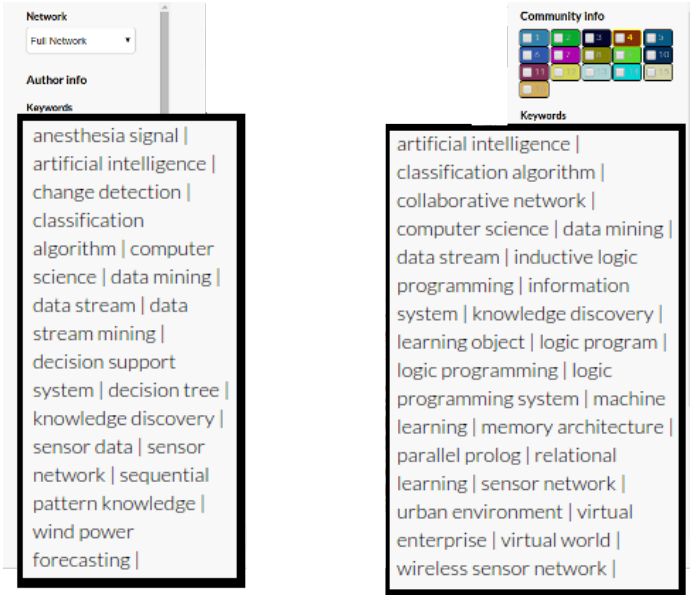


Figure 3.2. Keyword descriptors for a selected researcher in the left-hand side panel and for his/her affinity group in the right hand-side

The user has several options to select the node(s) he is looking for. It can be visually made by clicking on a node or by textual search – researcher's name in the main tab or keywords in a secondary tab. keywords provide the characterization of the activity of each researcher. Keyword descriptors for the selected author are presented in the left-hand side panel. Keyword descriptors of the researcher's affinity group are displayed on the right-hand side (Figure 3.2).

The quality of the characterizing keywords generated by this prototype is quite reasonable. For instance, for the selected author, the keywords collected from his webpage (Data Mining and Decision Support; Knowledge Discovery from Data Streams; Artificial intelligence) have a significant overlap with the ones that were automatically extracted.

Beyond the visual clues in the network canvas, the interface presents two distribution pie charts that give an insight into the distribution of the members that belong to the same affinity group and R&D unit of the selected researcher.

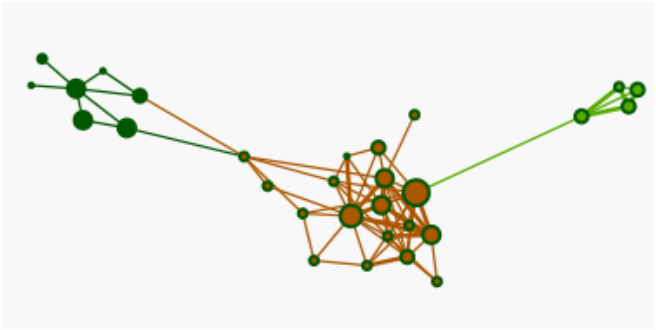


Figure 3.3. Network of an R&D unit/center

Concerning visual clues in the network representation, the strength of the similarity between researchers is represented by the thickness of the edge. The node border color identifies the affiliation unit, and the core color identifies the affinity group. If the user selects one of the research units in the left-hand dropbox, the canvas shows the corresponding network. Figure 3.3 shows the R&D network and their three affinity groups discriminated by the core color of the nodes.

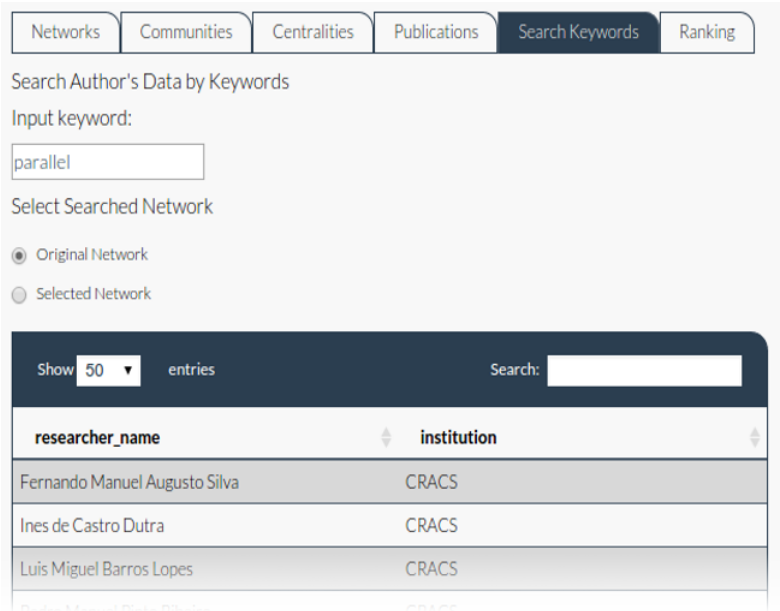


Figure 3.4. Searching network authors by keyword

If the user selects the other tabs on the top of the canvas, more detailed information is presented with some search options. One of the tabs enables the search for the researchers characterized by some particular keyword. Thus, finding authors associated with specific keywords or research areas is straightforward and intuitive. Figure 3.4 shows a specific search by the keyword *parallel*.

Figure 3.5 shows the folding of the full network in existing affinity groups. The network is shown in a higher hierarchical snapshot of the full network on the right side. The size of nodes is, in this context, proportional to the number of elements the affinity group has, and the width of

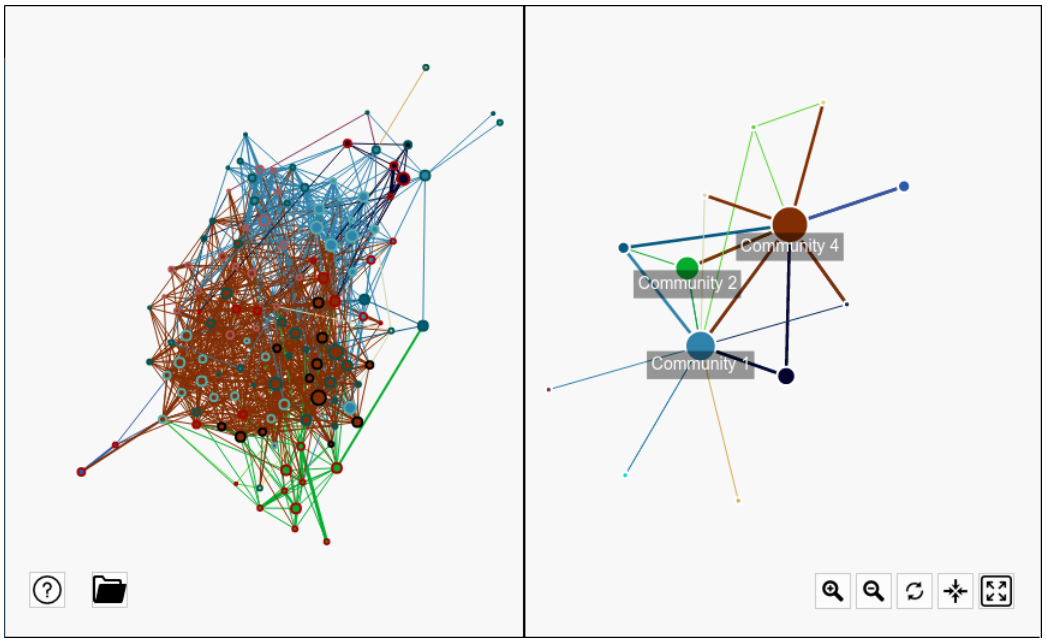


Figure 3.5. Affinity Miner prototype - Folded Networks

the connection/edges is given by the mean of the similarity weights for each inter-group element's connections.

Although this solution of visualizing higher hierarchical snapshots is comprehensible with the static data we tested, it might not be the best option with larger amounts of data.

3.2 Summary

Resuming, the Affinity Miner prototype has some interesting features:

- Allows to search documents through keyword search;
- Allows to inspect the structure of an institution;
- Allows for good visualization of the documents sets;
- Intuitive analysis through network visualization;
- Can be used to improve team/department work and workflow.

This prototype has been successfully used in mining affinity groups between researchers of a research center in Porto, Portugal.

Additionally, this same prototype concept has been used in mining the academic program of a Medical School in the Czech Republic, providing analysis of the affinity groups between Professors and their administered courses. The prototype was applied to the publications of the Faculty of Economics and Business of the University of Porto. With these case studies, the goal was to analyze the similarity between courses, and with a better decision support foundation, to achieve a better distribution of courses program and human resources.

3.3 Problems to Address

The analysis of large streaming networks poses processing or memory issues when using conventional hardware or software. Even if the available computational resources can analyze a network comprised of millions of nodes, it is difficult for the analyst to gather valuable knowledge from the outcome. Thus, some improvements are needed when considering static algorithms or procedures. Regarding applications scalability, when applied to large datasets of written documents, we should aim to improve them constantly. Particularly, considering the prototype that serves as a basis for this research:

- It does not scale well to larger sets of documents;
- The high dimensional TF-IDF matrix is difficult to process efficiently;
- It is a static solution. We should use a streaming approach;
- It is based on batch algorithms for keyword extraction or community detection.

Our aim is to improve these issues/characteristics. We focus on the main problems and one or several solutions while exploiting ideas from the existing work. Thus, we expect to use some of these techniques or algorithms to begin the research, test, and improve them as we develop our work of the thesis.

Problem

The streaming approach might raise some issues regarding related features of the system.

The Streaming Approach - Issues What should we do? How to work with the streaming approach?

- Using Sliding Windows can raise several questions:
  - What is the ideal size of the data window?
  - Will the window be time-based, e.g., one year or one month of publications?
  - Will the window be better based on the number of documents instead of time? e.g., 100 documents or more?

Our work is related to the scalability issue of evolving affinity networks. The introduction of more research centers and consequently more documents/researchers becomes much more demanding regarding computational processing. Thus, the processing of larger data requires the careful use of processing methods, storage, sampling, and display large-scale networks and streaming techniques similar or adapted from publications such as in Sarmento (2013); Sarmento et al. (2014a, 2013).

3.4 Related Work

Some authors, for example, in Blei (2012), emphasize that generative statistical and distribution models for text have the potential to make essential contributions to the statistical analysis of extensive document collections. One example of these generative models is the topic model. These models enable advanced statistical methods to identify the structure that underlies a set of words combined in phrases. Topic models might use many of the critical assumptions behind Latent Semantic Analysis (LSA) but enable the identification of a set of interpretable probabilistic topics rather than a semantic space. Investigating these models allows expanding both the practical benefits and the theoretical understanding of statistical language learning.

Our contribution to this preliminary work is the study of clustering and its implications when applied to the author's data. We experimented with the TF-IDF matrix after applying cosine similarity to it. We aim to find representative elements in the cluster division of the authors' similarity data and their topics. We used these extracted topics from the cluster's exemplar (or central element) to generalize to every cluster's element. We provided a comparison with a well-known benchmark method and obtained results for our dataset. We also tested the application of stemming in topic extraction and provided the results of this experiment.

3.4.1 Base Concepts

Several studies have already addressed topic modeling. This section first introduces related work with a more generic approach, including an overview of research in this area. The selected publications imply milestones or novelties regarding this subject of research.

Lin and Hovy claim that only about 30% of topic keywords are not mentioned in the text directly as written in Lin and Hovy (1997). The authors conclude that only about 30% of the abstracts in this domain derive from some inference processes. They also conclude that only about the same amount has to be derived by processes yet to be determined with further research in a computational implementation. The authors wrote that the titles contain about 50% of the topic keywords; the title plus the two most rewarding sentences provide about 60%, and the next five add another 6%. The authors, therefore, conclude that a reasonably small number of sentences provides 2/3 of the keyword topics.

Lin and Hovy provide empirical validation for the Position Hypothesis. The authors also describe a method of deriving an Optimal Position Policy for a collection of texts within a genre, as long as a small set of topic keywords is defined with each text. The Precision and Recall scores indicate the selective power of the Position method on individual topics. Additionally, the Coverage scores indicate a kind of upper bound on topics and related material as contained in sentences from human-produced abstracts.

Lean and Hovy evaluations treat the abstract as ideal - the authors rest on the assumption that the text's central topic(s) are contained in the abstract made of it. In many cases, this is a reasonable assumption; it provides what one may call the author's perspective of the text. However, this

assumption does not support goal-oriented topic search, in which one wants to know whether a text pertains to some particular prespecified topics.

Topic models have also been extended to capture some properties of language, such as the hierarchical semantic relations between words (Blei et al., 2003), and the interaction between syntax and semantics like described in Griffiths and Steyvers (2004).

Titov and McDonald (2008) presented multi-grain topic models and claimed that they are superior to standard topic models when extracting ratable aspects from online reviews. According to the authors, these models are suited to this problem since they enable the identification of essential terms and cluster them into consistent groups, which is a handicap of previously proposed methods.

AlSumait et al. (2008), developed an online topic model for discrete data to model the temporal evolution of topics in data streams. The researchers used a non-Markov online LDA Gibbs sampler topic model (OLDA). The current model, along with the new data, guided the learning of a new generative process that reflects the dynamic changes in the data. They achieved this by using the generated model, at a given time, as a prior for LDA at the successive time slice, when a new data stream becomes available for processing. The weight of history in the generative process can be controlled by the weight matrix depending on the homogeneity of the domain. The authors claimed that the model results in an evolutionary matrix for each topic in which the evolution of the topic over time is captured. In addition, the authors proposed an algorithm to detect emerging topics based on the framework of OLDA. By processing small subsets of documents only, OLDA is claimed to enable learning meaningful topics, in some cases with higher quality than the LDA baseline. Additionally, the authors claim their method also outperforms LDA in detecting topics represented by a small set of documents at a certain point in time.

Sendhilkumar et al. (2013), claim that their hPAM method is better to topic model research articles as the authors experienced better performance in terms of accuracy, precision, and recall for retrieval of relevant documents. The authors include originality (inverse of similarity) as a parameter to define novelty in the documents. The described approach is not entirely quantitative as it considers the semantics of concepts in the research article. The authors add that they will be focused on further implementations and a qualitative approach for research articles involving sentence importance and sentence contribution to novelty.

Gansner et al. (2013), experiment streaming topic extraction with LDA and TF-IDF and argued that when extracting topics from short texts like Twitter posts, the authors have better results with TF-IDF. Nonetheless, the authors did not experiment with the online version of LDA, the OLDA. According to the previously cited publication, (AlSumait et al., 2008), when compared with LDA, OLDA presents better results than LDA for some use cases.

In this preliminary study, we used LDA and TF-IDF with and without clustering. The task to retrieve the key terms with both methods from the titles and abstracts data was evaluated by using the author's written keywords in the scientific publications. Topic models such as LDA (Blei et al., 2003) and hierarchical models (Li and McCallum, 2006) have been successfully applied to various publications such as The American Political Science Review and Science. The work of Hall et al.

(2008), a group of researchers introduces the study of the history of ideas developments by using LDA and topic entropy. In Paul and Girju (2009), the authors extend over the work of Hall et al. (2008) by adding two related fields (Linguistics and Education) and by employing various novel topic models for scientific research analysis.

For this previous case study, we selected R&D publications from Czech Republic researchers. The dataset is publicly available in RDICCR (2015). The dataset can be exported to a .html file or .xls. This dataset has a high amount of information, including research area for each publication, authors, titles, abstracts, keywords, type of publication (which might include conference papers, book chapters, conference proceedings, patents, software, algorithms, among many others), author's research institution and others. This is a complete source of information, and there is information for dozens of years from around 1985 until 2015. This structured data's high quality and high organization make it a good source for text mining or NLP tasks.

After exporting the data for 2014, we selected conference papers and book chapters only. Therefore, the number of publications was reduced from around 25000 to 5110. We selected the titles and abstracts for this amount of publications for our research tasks. The titles have 11 words on average, and the abstracts have 240 words. This amount of publications represents approximately 2800 different first authors.

Pre-processing the data is essential in an information retrieval context since we are interested in reducing noise in data. Thus, we achieve less entropy in our models by treating the text and achieve better results from automatic machine learning procedures. The pre-processing of data included the following sequence of procedures:

1. removal of whitespaces
2. removal of stopwords
3. removal of punctuation except for hyphenated compounds
4. removal of numbering
5. convert every word to its lowercase version

This pre-processing was done entirely with the *tm* package available for the R language.

After text data pre-processing, there are 38544 terms in titles and 822888 terms in abstracts. It is visible that the titles have high sparsity with few words repeated many times and a high amount of words existing only one time. Figure 3.6 representation of frequency distribution makes us conclude that a power-law distribution approaches it.

The distribution of words in abstracts is represented in Figure 3.7. It is visible that the abstracts have high sparsity with few words repeated many times and a high amount of words existing only one time. Again, as previously with the titles, this representation of frequency distribution makes us conclude that a power-law distribution approaches it. This is an expected characteristic when studying text data.

Both previous figures were smoothed by applying a logarithmic function to the values of frequency occurrences.

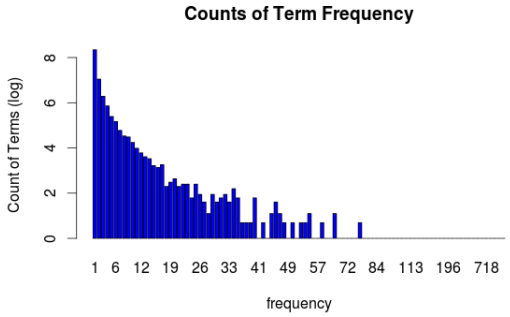


Figure 3.6. Frequencies Counting for Titles Words

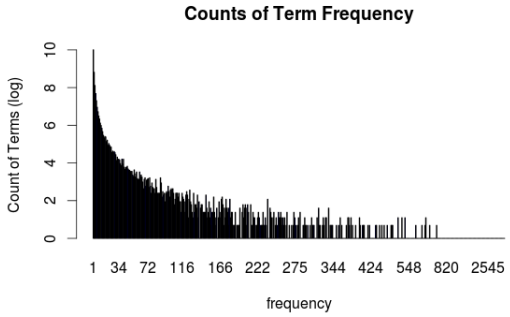


Figure 3.7. Frequencies Counting for Abstracts Words

3.4.2 Used Methodology

We were interested in finding topics in the documents available in the dataset with the described data. Our approach to the data had several possibilities. Succinctly, we could study:

- Titles of publications
- Abstracts of publications
- Titles and Abstracts concatenated

The data available enabled the use of the provided authors' keywords to validate all experimented methods and their results. Since we had the goal to find an affinity between authors and topics, we were interested in finding the topics each author approaches in their publications. Thus, we concatenated every author's publications in one document for each author. This was true either for titles, abstracts, or titles plus abstracts. To evaluate, we grouped all author's keywords in one document for each author.

The procedure to extract the topics relied on experimenting with two distinct methods, LDA and TF-IDF. For these methods, we found the similarity of the extracted topics/keywords with the keywords provided manually by the authors. In the end, we had the similarity/overlapping results for every author and regarding both methods. Additionally, we extended the TF-IDF method by

finding clusters of authors in the cosine similarity matrix obtained from the TF-IDF matrix. Then, we generalized the extracted topics of the central author (the exemplar or centroid) in each cluster for every author belonging to the individual cluster.

Algorithm 2 Algorithm Pseudo-Code for Topic Extraction

Input: *Authors_List*, *TextDataset* ▷ List of Authors in Dataset, Text Data (might be Titles of Publications, Abstracts or even Titles + Abstracts)
Output: *LDA_and_tfIDF_Results*

```
LDA_and_tfIDF_Results ← {}
for each Author ∈ Authors_List do
    LDATopics ← GETAUTHORTOPICSLDA(Author)
    tfIDFTopics ← GETAUTHORTOPICSTFIDF(Author)
    Authorkeywords ← GETAUTHORKEYWORDS(Author)
    simLDA ← SIMILARIYFUNCTION(Authorkeywords, LDATopics)
    simtfIDF ← SIMILARIYFUNCTION(Authorkeywords, tfIDFTopics)
    LDA_and_tfIDF_Results ← LDA_and_tfIDF_Results ∪ {simLDA, simtfIDF}
end for
```

Algorithm 3 Algorithm Pseudo-Code for Topic Extraction with clustering

Input: *Authors_List*, *TextDataset* ▷ List of Authors in Dataset, Text Data (might be Titles of Publications, Abstracts or even Titles + Abstracts)
Output: *LDA_and_tfIDF_Results*

```
LDA_and_tfIDF_Results ← {}
tfIDF_matrix ← GETTFIDFMATRIX(TextDataset)
Cosine_matrix ← GETCOSINEMATRIX(tfIDF_matrix)
Authors_Clusters ← GETCOSINEMATRIXCLUSTERS(Cosine_matrix)
while (Authors_Clusters) do
    for each Author ∈ Cluster do
        LDATopics ← GETAUTHORTOPICSLDA(Author)
        ExemplarTopics ← GETCLUSTEREXEMPLARTFIDFTOPICS(Cluster)
        Authorkeywords ← GETAUTHORKEYWORDS(Author)
        simLDA ← SIMILARIYFUNCTION(Authorkeywords, LDATopics)
        simtfIDF ← SIMILARIYFUNCTION(Authorkeywords, ExemplarTopics)
        LDA_and_tfIDF_Results ← LDA_and_tfIDF_Results ∪ {simLDA, simtfIDF}
    end for
end while
```

3.4.3 Extraction of Topics

We used two different methods to extract the topics from the data. Following some related work, we opted for LDA (Blei et al., 2003) and TF-IDF. For this task, we used the R language implementation of LDA from the package *topicmodels* (Grün and Hornik, 2011). The TF-IDF method was applied to the data by using the package *tm* also for R language (Feinerer and Hornik, 2018; Feinerer et al., 2008).

3.4.4 LDA

The intuition behind LDA is that documents have multiple topics. Furthermore, knowing that the document blends those topics would help situate it in a, for example, collection of scientific articles. LDA is a statistical model of document collections that tries to capture this intuition. It can be described by the generative process associated with this method. This process involves randomly selecting a model assumed to be one the documents arose from.

Both publications, Blei (2012); Blei et al. (2003) formally define a topic to be a distribution over a fixed vocabulary. For example, the *streaming* topic has words about streaming with high probability, and the *clusterization* topic has words about clusterization with high probability. Then, the authors generate the words for each document in the collection with a process involving two stages.

1. Randomly choose a distribution over topics
2. For each word in the document
    2.1. randomly choose a topic from the distribution over topics in step 1
    2.2. randomly choose a word from the corresponding distribution over the vocabulary

This statistical model reflects the intuition that documents exhibit multiple topics. Each document exhibits the topics in different proportions (step 1). Additionally, each word in each document is drawn from one of the topics (step 2b), where the selected topic is chosen from the per-document distribution over topics (step 2a).

The distinguishing characteristic of LDA is that all the documents in the collection share the same set of topics. Nonetheless, each document exhibits those topics in different proportions.

3.4.5 TF-IDF

TF-IDF is typically a computationally less complex option to calculate similarity using word counts. Since this counting can be biased toward common words in the documents, some adjustments must be made. Thus, the term counting is weighted by the inverse of the number of appearances of the same term in the documents. We can mathematically formalize this the following way:

- Let D be the set of documents, $d \in D$, a document consisting of a sequence of words (terms), and t a particular term of interest in d. Then the scaled word count based on TF-IDF is

$$tfidf(t,d) = tf(t,d) * idf(t,D) \tag{3.1}$$

Where $tf(t,d)$ is the fraction of times the term t appears in d, and $idf(t,D)$ is the logarithm of the inverse of the proportion of documents containing the term.

3.4.6 TF-IDF with Clustering

The similarity of documents can then be calculated by the cosine similarity of the TF-IDF vectors. After calculating similarity, we can extract the clusters presented in the author's list with the *apcluster* package for R language (Bodenhofer et al., 2011; Frey and Dueck, 2007). Finding clusters, in this case, will signify that we have similar authors grouped. We assume these authors are similar because they approach similar topics of studies. Then, we find the exemplar, i.e., the central author, and assume the exemplar topics represent the cluster topics for each cluster. We validate this assumption by iterating each cluster's authors and then comparing the cluster's topics with the cluster's authors' keywords.

3.4.7 Stemming

For further research about the results from both models, and since we were validating the models with the author's keywords, we were interested in finding the root of the words resulting from the automatic extraction and the root version of the keywords themselves.

It is commonly described that stemming is the term used in linguistic morphology and information retrieval to describe the process for reducing inflected or sometimes derived words to their word stem, base, or root form. Frequently, the stem is not identical to the morphological root of the word. Additionally, it is usually sufficient that related words map to the same stem, even if this stem is not, in itself, a good root.

For example, a stemming algorithm reduces the words "streaming", "streamed", and "streamer" to the root word, "stream". Another example, the words "endue", "endued", "endues", "enduing" reduce to the stem "endu". This last case exhibits a situation where the stem is not a word or root. By using stemming, we expect that both models increase the similarity with the keywords. This results from regular use by the authors of different forms for the same stem in different textual situations like, for example, in titles, abstracts, or keywords. Further developments will be presented in the following sections.

For this task, the R language was used, more specifically, the *tm* package (Feinerer and Hornik, 2018; Feinerer et al., 2008). This package allows executing the stemming of a Corpus of documents.

3.4.8 Evaluation of Topics/Keywords Generated

We use a similarity measure to compare the author's keywords and the model results for topics extraction to validate and compare both LDA and TF-IDF results. The following formula describes the method to find this similarity:

$$similarity_{(K,M)} = \frac{(dim(w_K \cap w_M))^2}{dim(w_K)dim(w_M)} \tag{3.2}$$

where $dim(w_K \cap w_M)$ is the intersection of both groups, i.e., the number of words that appear in both the results and the keywords. $dim(w_K)$ and $dim(w_M)$ are the total number of words the

Table 3.1. Average Similarity Table with LDA and TF-IDF without clustering

Average Similarity			
Topic Extraction Method	**Titles**	**Abstracts**	**Titles + Abstracts**
LDA	0.176	0.040	0.029
TF-IDF	0.217	0.105	0.103

author's keywords group has and the method (LDA or TF-IDF) provides, respectively. The values for the TF-IDF method similarity are calculated with the topics extracted from each author's TF-IDF matrix row and by selecting every term with a value superior to 0. Additionally, with LDA, we used all terms generated by the method for each author's topics.

3.4.9 Results

The results presented in this section were all obtained using the R language. The results are presented to compare LDA and TF-IDF methods applied to our data. We provide a comparison of the LDA method and the TF-IDF method with and without clustering. Finally, we also provide the same experiments but apply stemming to the data in the validation process.

LDA vs TF-IDF

The average similarity results are presented here to compare all methods. This average similarity is calculated with all first authors in the dataset.

Without clustering

This section presents a comparison between LDA and TF-IDF results for each author. Using only the titles, the average similarity for LDA is 0.176 and with TF-IDF is 0.217. For the abstracts with LDA, the average similarity is 0.040. Additionally, with TF-IDF, the value is 0.105. Regarding the titles plus the abstracts, with the LDA method, the average similarity is 0.029. With TF-IDF, this value is 0.103.

Table 3.1 presents results that indicate TF-IDF is better than LDA in the extraction of topics when these topics are compared with the author's keywords.

With Cluster Exemplars

In this section, we provide the comparison between LDA results for each author and the TF-IDF results obtained by generalizing the cluster's exemplar and its extracted topics.

Table 3.2 presents evidence that the TF-IDF method equals LDA with more data in the studied dataset.

Table 3.2. Average Similarity Table with LDA and TF-IDF clustering Exemplars Generalization

Average Similarity			
Topic Extraction Method	**Titles**	**Abstracts**	**Titles + Abstracts**
LDA	0.176	0.040	0.029
TF-IDF	0.066	0.029	0.029

Using Stemming - With and Without clustering

In this section, we provide the results for LDA, TF-IDF with clustering, and TF-IDF without clustering but using stemming when processing the extracted topics and the author's keywords.

Comparing table 3.3 results with previous tables 3.1 and 3.2, we emphasize that, by stemming the topic extraction results and also the keywords, leads to better values of average similarity. This result suggests that topic extraction of this dataset improves by using stemming. This is true for every variant of the method or dataset (titles or abstracts, or even titles plus abstracts).

Discussion of Results

Our results indicate that TF-IDF was the best method to extract topics with our dataset automatically. Additionally, we obtained better similarity values with the titles dataset, which indicates that both models extract more topics/keywords with less uncertain provided terms. Therefore, these topics had higher similarities to the keywords provided by the authors. With abstracts or titles plus abstracts, the similarity values were lower. This happened because both models extract more terms and intersect less with the keywords provided by the authors.

Our results, by using clustering, were not conclusive to provide clear improvements in the extraction of topics compared with the LDA method for each author. We stress that we have conducted a study with one year of data (2014). This short period of data might cause the clustering to have low density. Thus, the elements belonging to the same cluster might not have the expected similarity between them. Additionally, as the amount of data increased, for example, with titles plus abstracts, the similarity decreased and is comparable to the similarity obtained with the LDA method. Thus, the exemplar or central element of the cluster might be representative of the cluster's generalization of topics if we had more data. Since we had only one year of data in these

Table 3.3. Average Similarity Table with stemming and for LDA or TF-IDF, with and without clustering

Average Similarity with Stemming			
Topic Extraction Method	**Titles**	**Abstracts**	**Titles + Abstracts**
LDA	0.206	0.044	0.033
TF-IDF with clustering	0.076	0.031	0.032
TF-IDF without clustering	0.256	0.120	0.118

tests, we had less than two publications per year and each author, which implies more variety of research areas and, therefore, points in data clusters with higher dispersion. So, the question is, if we had more data, TF-IDF with clustering might be better than LDA?

Another exciting result from our experiments is the improvement exhibited by applying stemming to our extracted topics and keywords. This indicates that authors use different words selected for keywords and the same subject inside titles and abstracts text.

3.5 Summary and Contributions

The building of the Affinity Miner prototype revealed solutions for the organization of documents and search for information. We concluded that, with the building of this prototype, although it provided a good sense of ordering, author and/or subject search, it had characteristics that could be improved.

The mentioned preliminary study, to try to further research the possibility of improving text analysis, provided an introduction to the subject of Topic Modeling and local and global keyword search with the intention to better organize or group text documents. This research area is an important task and it was considered a starting point to applications in need of document clustering or summarizing. In this preliminary work, we also introduced the dataset we used in the testing of LDA and TF-IDF with and without clustering. The task to retrieve the topics with both methods from the titles and abstract data was evaluated using the author's keywords. We also tested the methods with stemming processing experiments to check if we obtained better topic modeling. Stemming proved helpful when evaluating our model against the author's keywords.

Main Contributions

The author of this thesis actively contributed to the development of the Affinity Miner prototype and co-authored the following publications related to the research and prototype development:

Brazdil, P., Trigo, L., Cordeiro, J., Sarmento, R., and Valizadeh, M. (2015). Affinity mining of documents sets via network analysis, keywords and summaries. Oslo Studies in Language, 7(1).

Trigo, L., Vita, M., Sarmento, R., and Brazdil, P. (2015). Retrieval, Visualization and Validation of Affinities Between Documents. In Proceedings of the 7Th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management,volume 3 of Kmis, pages 452–459.

Chapter 4

Organizing Large Datasets of Text Documents

In Chapter 3, we discussed the advantages of having an organized set of documents prior to searching for the relevant one. Nonetheless, processing large amounts of documents represents a problem that needs to be addressed. This is why we have investigated sampling which is usually used for this kind of situation. We have experimented with several sampling methods and describe the proposed solution in Section 4.1. This topic is important, as the sampling methods were used extensively further on in this thesis.

Grouping of similar documents into similar groups facilitates the search for a document that satisfies best the user's requirements. The user may go through the elements of the group to select one that matches best his/her requirements. This problem of grouping is addressed as a problem of "community detection" that form part of a complex network. Section 4.2 is dedicated to the description of a particular community detection method and the results obtained when this method is applied to a given network.

Additionally, when exploring a set of documents, knowing which document is more "central" is also of great importance. Section 4.3 is dedicated to the issue of centrality in networks and some measures used for this purpose. As time progresses, it is essential to adapt to changes in the set of documents available, either by alterations to existing documents or new documents arriving in the stream. In this section we have proposed methods that calculate certain centrality measures in an incremental way.

Keywords are important when working with document sets, as they can be used for chacterizing both the individual documents and also the subsets of documents regarded as communities. Section 4.4 is discusses a graph-based approach to keyword extraction, that was updated to be used in a streaming setting. The final section (Section 4.5) discusses network visualization, which enables the user to inspect the network or its parts.

4.1 Methodology for Sampling Networks

In this section, we introduce several sampling methods, as well as effective visualization, of large streaming networks in order to address the above-mentioned problems. More specifically, we introduce the *Top-K* sampling method that focuses on extracting a sample of the most active nodes in the network. This approach is suitable for networks exhibiting a power-law behavior and is able to preserve the same distribution of the original network and the global community structure. Besides, it is highly efficient, either for visualization or analysis purposes, because it relies on the *Space-Saving algorithm*.

Sampling Large Streaming Networks Papagelis et al. (2013) introduced sampling-based algorithms that quickly obtain a near-uniform random sample of nodes in its neighborhood, given a selected node in the social network. The authors also introduce and analyze variants of these basic sampling schemes, aiming at the minimization of the total number of nodes in the visited network by exploring correlations across samples.

Several approaches have been proposed to gather information from streaming graphs. Typical SNA problems, such as triangle counting, centrality analysis, and community detection, have already been implemented in streaming settings. We will delve deeper into these topics further in the chapter.

Network sampling of streaming graphs is still a promising area for future research since, to the best of our knowledge, only a few stream-based sampling methods have been proposed so far. Ahmed et al. (2012) present a novel approach to graph streaming sampling. According to the authors, there was no previous contribution to this topic. The authors propose a novel sampling algorithm, dubbed PIES, based on edge sampling and partial induction by selecting the edges that connect sampled nodes.

4.1.1 Top-K Sampling with Top-K itemsets

Researchers have been trying to achieve efficient ways of analyzing data streams and performing graph summarization. The exact solution implies the knowledge of the frequency of all nodes and edges, which might be impossible to obtain in large-scale networks.

The problem of finding the most frequent items in a data stream S of size N is basically how to discover the elements e_i whose relative frequency f_i is higher than a user-defined support ϕN, with $0 \leq \phi \leq 1$ (Gama, 2010). Given the space requirements that exact algorithms addressing this problem would need (Charikar et al., 2002), several algorithms were already proposed to find the top-k frequent elements, being roughly classified into *counter-based* and *sketch-based* (Metwally et al., 2005). *Counter-based* techniques keep counters for each individual element in the monitored set, which is usually a lot smaller than the entire set of elements. When an element is identified as not currently being monitored, various algorithms take different actions to adapt the monitored set accordingly. *Sketch-based* techniques provide less rigid guarantees, but they do not monitor a subset of elements, providing frequency estimators for the entire set.

Simple *counter-based* algorithms that process the stream in compressed sizes, such as *Sticky Sampling* and *Lossy Counting*, were proposed by Manku and Motwani (2002). Yet, these have the disadvantage of keeping a large number of irrelevant counters. *Frequent* (Demaine et al., 2002) keeps only k counters for monitoring k elements, incrementing each element counter when it is observed, and decrementing all counters when an unmonitored element is observed. Zeroed-counted elements are replaced by new unmonitored elements. This strategy is similar to the one applied by the *Space-Saving algorithm*, proposed by Metwally et al. (2005), which gives guarantees for the *top-m* most frequent elements. *Sketch-based* algorithms usually focus on families of hash functions that project the counters into a new space, keeping frequency estimators for all elements. The guarantees are less strict, but all elements are monitored. The *CountSketch* algorithm (Charikar et al., 2002) solves the problem with a given success probability, estimating the frequency of the element by finding the median of its representative counters, which implies sorting the counters. Also, *GroupTest* method (Cormode and Muthukrishnan, 2005) employs expensive probabilistic calculations to keep the majority elements within a given probability of error. Despite the fact of being generally accurate, its space requirements are large, and no information is given about frequencies or ranking.

Algorithm 4 represents the proposed *Top-K* Method application using the *Space-Saving algorithm*.

This type of application is based on a landmark window model (Gama, 2010), which implies a growing number of inspected events in the accumulating time window. This landmark application is also useful in other contexts, e.g., when the network is relatively small, and the user wants to check all events in it.

Experiments using the landmark window model showed that this model suffers from the problems we would like to avoid, such as exceeding memory limits. This happens when the number of nodes and edges exceeds dozens of thousands of nodes. The *Top-K* algorithm, based on a landmark window model, is an efficient approach for large-scale data. It focuses on the most active nodes and discards the least active ones, which are the most frequent according to the power-law distribution. The alternative option to the landmark window model, i.e., the sliding window model (Gama, 2010), would not be appropriate for the *Top-K* approach since it may remove less recent nodes. Those nodes may yet be included in the *Top-K* list we want to maintain.

In our scenario, the *Top-K* representation of data streams implies knowing the K elements of the simulated data stream from the database. network nodes that have a higher frequency of outgoing connections, incoming connections, or even specific connections between any node A and B may be included in the graph, as well as their connections.

For this application, the user can insert as input a start date and hour and also the maximum number of *Top-K* nodes to be represented (the K parameter), along with their connections.

With the inserted start date and hour, the *Top-K* application is expected to return the evolving network of the *Top-K* nodes. Functions *getTopKNodes* and *updateTopNodesList* in algorithm 4 implement the *Space-Saving algorithm*. As the network evolves over time, new *Top-K* nodes are

Algorithm 4 Top-K algorithm for call graphs

Input: $start$, k_param, $tinc$ ▷ start timestamp, k parameter and time increment
Output: $edges$

```
1:  R ← {}                                                  ▷ data rows
2:  E ← {}                                                  ▷ edges currently in the graph
3:  R ← getRowsFromDB (start)
4:  new_time ← start
5:  while (R <> 0) do
6:      for each edge ∈ R do
7:          before ← GETTOPKNODES(k_param)
8:          UPDATETOPNODESLIST(edge)                        ▷ update node list counters
9:          after ← GETTOPKNODES(k_param)
10:         maintained ← before ⋂ after
11:         removed ← before \ maintained
12:         for each node ∈ after do                        ▷ add Top-K edges
13:             if node ⊂ edge then
14:                 ADDEDGETOGRAPH(edge)
15:                 E ← E ⋃ {edge}
16:             end if
17:         end for
18:         for all node ∈ removed do                       ▷ remove non Top-K nodes and edges
19:             REMOVENODEFROMGRAPH(node)
20:             for all edge ∈ node do
21:                 E ← E \ {edge}
22:             end for
23:         end for
24:     end for
25:     new_time ← new_time + tinc
26:     R ← getRowsFromDB (new_time)
27: end while
28: edges ← E
```

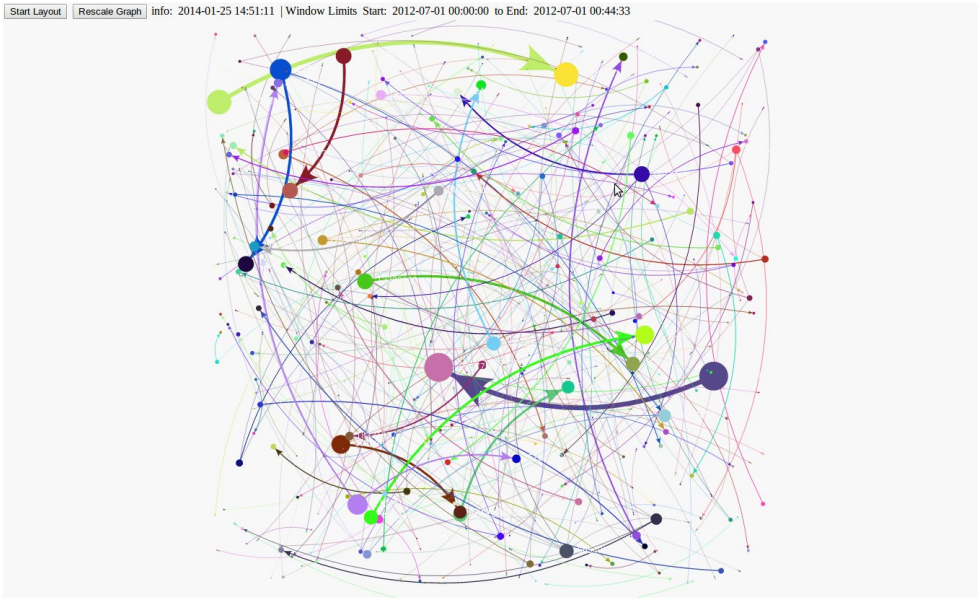


Figure 4.1. Network induced by the top-100 subscribers with the highest number of phone calls and corresponding direct connections. This network was generated without running the layout algorithm.

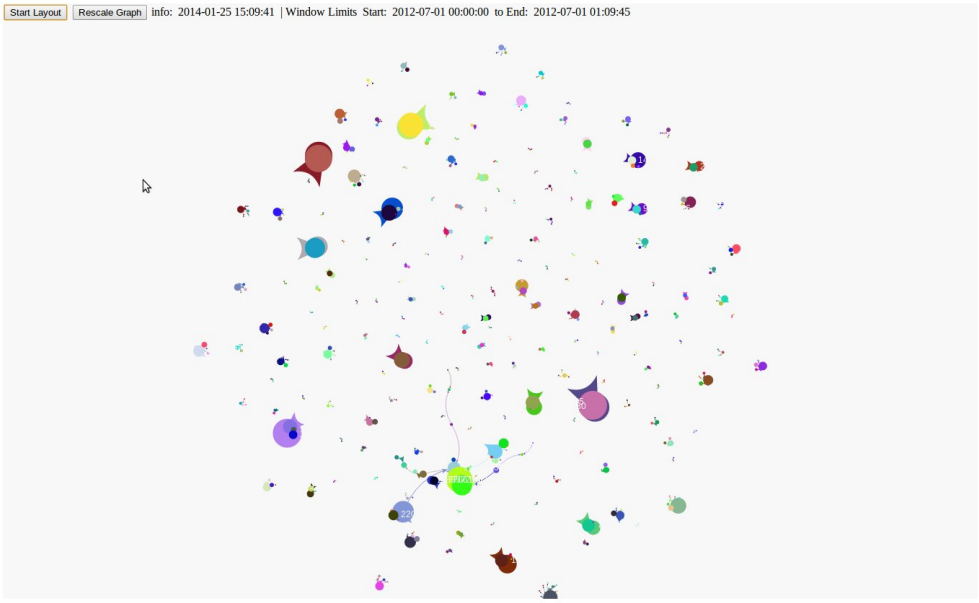


Figure 4.2. Network induced by the top-100 subscribers with the highest number of phone calls and corresponding direct connections. This network was generated after running the layout algorithm.

added to the graph. Nodes that exit the *Top-K* list of numbers are removed from the *Top-K* list and, thus, removed from the graph along with their connections.

Figure 4.1 represents the network induced by the top-100 subscribers with the highest number of phone calls, since the midnight of the first day of July 2012, until 00h44m33s. The algorithm shows the 100 most active phone numbers in that period. Figure 4.2 depicts a similar network but after running the layout algorithm. This time, the output considers results until 01h09m45s.

4.1.2 Open Issues

The sampling of large social networks is still in its infancy, and there are important open research issues and unsolved problems not yet satisfactorily addressed by the scientific community. Firstly, it is necessary to achieve consensus on the definition of *representative sample*. A representative sample is a subgraph of the original network that matches its structural properties based on previous research. However, it is unclear which structural properties (e.g., degree distribution, global clustering coefficient, motifs, community structure) should be preserved in the sample. Another matter of concern is the sample size. Despite the promising results of empirical studies (Leskovec and Faloutsos, 2006), a rigorous formal study of the most appropriate sample size according to the size and characteristics of the original network still has to be done.

The proposed solutions for streaming networks are still quite simple and leave much space for improvement regarding, for example, visualization. A straightforward extension of the presented techniques would include additional information employing visual cues, such as color, size, and shape. An idea would be to integrate the visualization techniques with the incremental algorithms used to compute computationally demanding centrality measures (e.g., closeness and betweenness) and then include this node-level information on the visual output. On the other hand, it is necessary to explore new representations beyond the graph model to visually display interesting patterns in streaming networks.

Community detection on large dynamic social networks faces many of the challenges of static community detection, namely in what regards the lack of a consensual definition of *network community* and the evaluation of the network partitions produced by community detection algorithms. A possible way to circumvent these problems would be to support the definition of community on the specific domain and develop evaluation measures specifically tailored for the application. For instance, telecommunication providers might be interested in finding communities defined by the connections induced by mobile communications and by business variables (e.g., revenue generated by each user), geographical position, and demographic attributes. Depending on the purpose of the community detection task, telecommunication companies could, for instance, perform the evaluation based on the similarity of the users' response to the marketing campaigns targeted to the community they were assigned to. Another important issue, especially for telecommunication providers, is to create models and procedures to characterize and define profiles of communities.

Overall, the major challenge is to devise a system that integrates all the relevant steps involved in extracting useful knowledge from large streaming social networks. This system should be built upon rigorous methods and appropriate algorithms while being user-friendly to encourage

its use by business managers and decision-makers. The use of social network analysis applied to streaming telecommunications networks will benefit society. More specifically, the network users, in terms of service quality, largely due to the rapidity of action that allows operators to benefit from. Service problems, which ultimately may adversely affect the service to the point where the customer will want to leave the operator, will decrease significantly. Otherwise, the problems that will probably arise upon it will necessarily be discussed in more detail in the future, relate to the use of such information for dissemination of network spam, dubious character information, or that might involve the loss of the network user's privacy.

4.1.3 Summary and Contributions

The Top-K algorithm, oriented towards sampling of networks, and particularly to evolving networks represents one of contributions. Sampling with Top-K algorithms is important when dealing with massive amounts of data, as in the case of evolving networks. It enables to cope with the flux of text documents. The main advantage of Top-K sampling algorithms is the great improvement in efficiency and memory usage.

We have published various articles on the topic discussed in this section: Sarmento et al., 2015a; Sarmento et al., 2016a. Sampling with Top-K algorithms forms part of the methods discussed not only in this chapter (Sections 4.2 and 4.5), but also in Chapter 5 (Section 5.3).

4.2 Community Detection

4.2.1 Introduction

Considering we want to organize documents, one main task is the grouping of those documents in similar groups. This important task allows documents to be searched for with more efficiency. The organized structure allows to discover similar documents, that might be of interest.

In this chapter, we deal with the grouping of these similar documents. The grouping of documents implies a method to group these documents that are more or less similar between them. This grouping, is normally designated by community detection, when we are dealing with similarity networks.

Community detection in Social network Analysis (SNA) is a critical research area in an enormous number of unrelated areas. From Psychology to Physics, community detection in SNA is used to find an agglomeration of study objects in a graph. We have witnessed the development of an abundance of algorithms specifically designed to identify communities in graphs of every size, from small graphs with some dozens of vertices and edges to large or very large graphs with millions of vertices and billions of edges.

Very recently, with the growing popularity of SNA, researchers have been migrating concepts and some algorithms to stream setting. This is even more important with the appearance of sources of data that are streamable, such as social networks like Facebook and Twitter, where information arrives as a flow of discrete events that, usually, have a limited existence through time.

4.2.2 Related Work

Cordeiro et al. (2018) show the example of a contact evolving network with instantaneous interactions between vertices. When the interaction between network peers has a time duration, we are in the presence of interval Evolving Networks. If the time T during which a network is observed is finite we can consider the start point $t_{start} = 0$ and the end time as $t_{end} = T$. A dynamic network graph $G^D_{0,T}(V, E_{0,T})$ on a time interval $[0, T[$ consists of a set of vertices V and a set of temporal edges $E_{0,T}$. The evolving network is a set of graphs across the time axis within discrete time points $t_1, t_2, ..., t_{n-1}, t_n$. At time point t_n is observed a graph instance $G(V_n, E_n)$ also denoted as G_n where E_n is the set of temporal edges $(u, v)_{t_n} \in E_{0,T}$ at time point t_n with edges between vertices u and v on time interval $t_n = [t_{n_{begin}}, t_{n_{end}}]$ such that $t_{n_{begin}} \leq T$ and $t_{n_{end}} \geq t_{n_{begin}} \geq 0$. Examples of network changes that may occur between two time points t_{n-1} and t_n are the addition of new edges, i.e.: $E_n \supset E_{n-1}$, and the appearance of additional vertices, i.e.: $V_n \supset V_{n-1}$.

Models of Temporal Representation

The concept of a time-ordered graph, for example, a network for the time interval $[0, 3]$, shows all the time intervals aggregated into a single graph $G_{1,3}$. We use the discretization of the network by converting the temporal information into a sequence of n snapshots, and the evolving network

is represented as a series of static networks $G_1, G_2, ..., G_n$. The time-ordered graph $G = (V, E)$ assumes that a message can be delivered along a single edge at each time step. The time-ordered graph of Kim and Anderson (2012) is the model used in the rest of this document.

Landmark vs Sliding Windows

When the temporal dimension is added to the analysis of networks, methodologies relating to the strategy to deal with the knowledge being analyzed vary. There are three kinds of graph knowledge windowing ways. ***Landmark windows*** (Gehrke et al., 2001) includes all the info from a particular purpose in time, up to the present moment. Within the Landmark window, the model is initialized at a determined point in time, i.e., the landmark that marks the start of the window. The info window grows in ordered snapshots to analyze all the data seen up to now, since the landmark start. ***Sliding windows***, from another point of view, is appropriate when we are not inquisitive about computing statistics over all the past, solely over the recent past (Gama, 2010). The paper of Datar et al. (2002) incorporates a forgetting mechanism, does not consider all the data falling outside the window by keeping the newest data within the window solely. These windows can be defined regarding length in two distinct ways, the time-based length and the sequence-based (Babcock et al., 2002a,b). Sequence-based models, wherever the dimensions of the window are, are set relating to the number of observations. In the Timestamp-based models, the other type of window generation, the dimensions of the window are outlined regarding time sample length. A timestamp window of size t consists of all event elements whose timestamp is within a time interval t since the beginning of the data processing or since the beginning of the current processed data period.

Dynamic Community Detection

Because of both global and local heterogeneity of edge distribution in a graph, specific regions of a graph evidence high concentration of edges within particular regions, called *communities*, whereas inter regions have low concentrations of edges. In the context of networks, these occurrences of groups of vertices in a network that are more densely connected internally than with the rest of the network are called *community structure*. Also known as *modules* or *clusters*, communities can be straightforwardly defined as groups of similar vertices. A complete definition using the concept of density can be the following: communities can be understood as densely connected groups of vertices in the network, with sparser connections between them.

A greedy algorithm based on modularity optimization has been introduced by Blondel et al. (2008) where initially all vertices of the graph are put in different communities. The first step consists of a sequential sweep over all vertices, for each of the neighbors picks the community that yields the largest increase of modularity. At the end of the sweep, one obtains the first level partition. In the second step, communities are replaced by super vertices. The weight of the edge

between the super vertices is the sum of the weights of the edges between the represented communities at the lower level. The algorithm's two steps are then repeated, yielding new hierarchical levels and supergraphs.

Finding Communities in Dynamic Networks When discussing methods for finding communities in dynamic networks, the division of methods for slowly evolving networks and streaming networks is consensual (Aggarwal and Subbian, 2014). In the following section, algorithms for both scenarios will be presented and analyzed.

Slowly Evolving Networks

Cordeiro et al. (2016) presented a modularity-based dynamic community detection algorithm. It modified the original Louvain method where dynamically added and removed vertices and edges only affect their related communities. In each iteration, all the communities that were not affected by modifications to the network maintain unchanged. By reusing community structure obtained by previous iterations, the local modularity optimization step operates in smaller networks. Thus, only affected communities are disbanded to their origin. Compared with the original algorithm, communities' stability is also an improvement. Compared with the original algorithm, with that algorithm run several times, the results in changes on communities or vertex drift from one community to another are easier to follow when presented by the dynamic and incremental algorithm. This is because only parts of the network change during iterations. The algorithm's non-determinism will have a reduced effect on the community assignment, providing better community stability than its counterparts.

Streaming Networks

Streaming graph algorithms are essential to perform community detection with high-frequency data and large or very large networks. In streaming scenarios, the ability to perform deletion of edges in community detection algorithms is important. In short, this will dictate if the method of analysis is to be performed over a sliding window of edges, and therefore edges are deleted from the tail end of the sliding window, or over a landmark window, in case there is no possibility to delete or forget old edges. Several methods were proposed for dynamic community discovery in graph streams. In Wang et al. (2013), researchers motivated by the variability of the underlying social behavior of individuals over different graph regions modeled the problem according to the so-termed *local heterogeneity*, where a Local Weighted-Edge-based Pattern (LWEP) summary is efficiently maintained and used afterward to cluster the graph stream and perform dynamic community detection in weighted graph streams. Taking an almost linear time, Raghavan et al. (2007) investigated and a simple label propagation algorithm that uses the network structure alone as its guide and requires neither optimization of a predefined objective function nor prior information about the communities. By analyzing the problem of real-time community detection in large

networks and having by baseline the algorithm proposed by Raghavan et al. (2007) with linear time-$O(m)$ on a network with m edges-label propagation, or "epidemic" community detection, Leung et al. (2009) proposed a method with near-linear time community detection in graphs. They identified the characteristics and drawbacks of the base algorithm (Raghavan et al., 2007) and extended it by incorporating different heuristics to facilitate reliable and multifunctional real-time community detection. In Yun et al. (2014), the authors proposed two efficient streaming memory-limited clustering algorithms for community detection based on spectral methods. In Yun and Proutière (2014), researchers proposed community detection via random and adaptive sampling. In Sariyüce et al. (2016), the authors proposed SONIC, a find-and-merge type of overlapping community detection algorithm that can efficiently handle streaming updates. Recently, Hollocou et al. (2017) proposed SCoDA, a linear streaming algorithm for community detection in very large networks.

Top-K Communities *Top-K* communities are defined as groups of densely connected nodes in *Top-K* networks (Sarmento et al., 2015c). As the name implies, in this work these communities are detected considering only the *Top-K* nodes and their 1st and 2nd neighborhood-order connections. This method samples the original network in a way that preserves its structural properties and the community structure of the original network. We apply *Top-K* sampling to obtain the nodes that belong to the *Top-K* group. To retrieve their network we query the database so as to collect all connections/edges representing the network with the neighbors of the *Top-K* nodes.

After generating the sampled networks, the *Louvain method* (Blondel et al., 2008) is applied to find the communities. We studied the matching between the community membership obtained for the *top*-10000 network and the community membership of the original network retrieved by the *Louvain method*. This task was performed for an entire day of streaming data. The matching of communities for the two scenarios (sampled network and original network) is performed by retrieving the percentage of matching community members between any *Top-K* network community and the original network communities.

The matching results of the 100 largest communities for the sampled network and the 20 largest communities in the original network showed that the proportion of community member matching varies. There is a considerable matching of the *top*-10000 sampling communities and the 20 largest communities of the original network. These highly active callers and the communities they belong to are therefore represented in the *Top-K* sampling, as expected.

Experiments were also conducted using other days of the dataset. The results are very similar and consistent throughout full-day data comparisons and for the complete dataset of more than 100 days. In all comparisons, it was visible that larger original dataset communities are matched by communities retrieved with the proposed *Top-K* sampling method.

4.2.3 Community Detection and Average Density

The modularity measure was introduced in Newman and Girvan (2004), apart from being the most widely used according to Chen et al. (2015, 2013), when evaluating community division in

community detection algorithms.

Modularity Modularity was considered as the quality measure used in the evaluation of the algorithms for community detection in static networks. Higher values for the modularity Q mean better community structures. Therefore, the objective is to find a community assignment for each node in the network, such that Q is maximized using the modularity function defined by

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j) \tag{4.1}$$

A_{ij} represents the weight of the edge between i and j, $k_i = \sum_j A_{ij}$ is the sum of the weights of the edges attached to vertex i, c_i is the community to which vertex i is assigned, the δ-function $\delta(u,v)$ is 1 if $u = v$ and 0 otherwise and $m = \frac{1}{2}\sum_{ij} A_{ij}$. To calculate the modularity of a specific community, the number of inner edges ($in[n]$) and the total number of edges ($tot[n]$) of a specific node n is used. The modularity of the full network can be calculated using the previous Q function, by considering all the entries of in and tot for all the nodes.

The detection of communities within streams of data and the folding of the network in the representative prototypes/exemplars of the communities is one chance to scale the data analysis and visualization to thousands of authors, like the use of centroids previously mentioned. In Cordeiro et al. (2016), the authors describe an adaptation of the Louvain algorithm for community detection with streaming networks. This is an incremental approach to the problem of community detection. This algorithm was also published in a different format, this time concerning the launch of the DynComm R package (Sarmento and Lemos, 2020). [1]

We should stress the importance of discovering the prototype of each community efficiently. This can be done, for example, by inspecting *eigenvector centrality* measures of the nodes of the community and consider prototypes as the nodes with higher values. The online learning of *eigenvectors* is, for example, studied in Garber et al. (2015). Thus, with this prototype or representative node, we could then present a summarized version of the network with fewer but representative nodes within their communities.

Additionally, we can also use these representative nodes as the providers of global keywords for each community they belong. Different from previously mentioned methods where we looked for centroid clusters, this method regarding communities in the graph might also be effective.

Problem

The community detection with large datasets can be a slow process with thousands or even millions of authors. It is currently done with the random walks batch algorithm.

What should we do?

- use incremental algorithms for community detection?

[1] Available Code at https://github.com/softskillsgroup/DynComm-R-package.

– What algorithms should we use?

Cordeiro et al. (2016) implemented an incremental version of the Louvain algorithm. It is prepared to process incremental community detection with local modularity optimization. This is prepared to cope with large-scale graphs by following some procedures to perform:

- incremental addition of edges
- incremental removal of edges
- incremental modularity optimization of affected communities for each new edge

This results in an improvement of incremental community detection algorithms. In Figure 4.3 we see the results of that algorithm considering modularity, modularity density, and processing time measurements:

	Dynamic Louvain	LabelRankT	GANXiSw	AFOCS
# Nodes	2811	3722	3718	3718
# Edges	11,756	13,423	13,407	13,407
Duration	55	549	1109	544
Cumulative	22,070	219,786	443,488	217,714
Total communities	224	156	150	84
Modularity ($\mathcal{Q}$)	0.5590	0.4337	0.4955	0.1210
Split penalty ($\mathcal{SP}$)	0.4010	0.2211	0.2281	0.7603
Modularity with split penalty ($\mathcal{Q}_s$)	0.1581	0.2127	0.2674	−0.6436
Modularity density ($\mathcal{Q}_{ds}$)	0.0821	0.0478	0.0565	0.0069
Unassigned nodes	37	8	0	6154

Figure 4.3. Cordeiro Incremental Community Detection (reproduced from Cordeiro et al. (2016))

Developed R Package Available since some years ago, a variety of packages in languages as Python or R have been developed to cope with the need for analysis of communities and social networks. Nonetheless, few or no packages that deal with community detection in evolving networks are available right now, in R-CRAN. Thus, this is the right time to develop a way to provide researchers with a package that fulfills the need to explore network streams, particularly community detection of evolving networks. This poses a challenge since algorithms' adaptation is not an easy task and sometimes even impossible due to restrictions in the architecture of the static algorithm.

We developed a framework that we believe will help future developments in this area be included in the package Sarmento and Lemos (2020), with the least effort for the new algorithms' authors [2].

Implemented Algorithms

Questionable Modularity and Density Optimization

[2] Available Code at https://github.com/softskillsgroup/DynComm-R-package.

Modularity-based algorithms used for community detection have been increasing in recent years. Modularity and its application have generated controversy since some authors argue it is not a metric without disadvantages (Lancichinetti and Fortunato, 2011; Arenas et al., 2008). It has been shown that algorithms that use modularity to detect communities suffer a resolution limit and, therefore, it is unable to identify small communities in some situations. In the function in our package, we try to apply a density optimization of communities found by the available algorithms explained in Sarmento (2019). We introduce a metric we call ADC (Average density per Community); we use it to prove our optimization provides improvements to the community density obtained with benchmark algorithms. The results of the optimization algorithm proved to be interesting.

Several developments were made to test the hypothesis. An algorithm was developed, and a metric is introduced in the following sections.

Average density per Community (ADC) measure

Average density per Community (ADC) is the measure that is used to compare the algorithm results and is given by the following formula:

$$ADC = \frac{1}{n_C} \sum_{C_{i=1}}^{n} Density(Ci)$$

Where n_C is the number of communities identified in the graph, $Density(Ci)$ is the density of each community Ci.

Optimization Algorithm

Algorithm 5 provides the sequence of tasks we are doing in the post-processing application [3]. We start by using the results of a community detection algorithm. Then, we try to discover if the communities can be disbanded in smaller communities. These smaller communities are strongly connected components, i.e., groups of nodes with higher density. Then, if the disbanded communities' average community density is higher than the original community, the disbanding is indeed executed. If not, the community founded by the benchmark algorithm is not disbanded and maintains its original id.

4.2.4 Case Study and Evaluation

We test our density optimization algorithm (DENSOPT) with several real-world networks in this case study. We used these networks to test the hypothesis that our algorithm improves the ADC measure in community detection. We also provide a comparison of the modularity results with and without optimization. We provide a comparison with the label propagation algorithm (Raghavan et al., 2007) and also the INFOMAP algorithm (Rosvall and Bergstrom, 2008). Label propagation

[3] Available Code at https://github.com/Sarmentor/Density-based-Community-Detection-Optimization.

Algorithm 5 Algorithm Pseudo-Code for Optimization of Community density

Input: *Comm_Data* ▷ Node List and their Community
Output: *Comm_Res* ▷ New Community Structure

while not at the end of *Original_Communities* list **do**
  if $ncomponents > 1$ **then** ▷ If community has more than 1 component
    $SCC \leftarrow$ STRONG_CONNECTED_COMPONENTS_OF_COMMUNITY$(Comm_i)$
    $mdc \leftarrow$ MEAN_DENSITY_OF_COMPONENTS$(SCCs)$
    if $mdc > Community_Density$ **then**
      for $SCC_i \in Community$ **do**
        $Comm_Res \leftarrow$ COMPONENTNODESFORMNEWCOMMUNITY(SCC_i)
      end for
    else
      do_nothing
    end if
  else
    process *next_community*
  end if
end while

is different in a sense that uses a different maximization measure. We use average of density for the community detection in DENSOPT.

Label propagation, as its name indicates, does the propagation of community labels. At the initial condition, nodes carry a label that denotes their community. The assignment to a community changes based on the neighboring nodes' labels. This change is subject to the maximum number of labels within one degree of the nodes. Every node is initialized with a unique label then the labels diffuse through the network. Consequently, densely connected groups reach a common label quickly. When many such dense (consensus) groups are created throughout the network, they continue to expand outwards until it is possible to do so.

The label propagation algorithm has the advantage in its running time, an amount of a priori information needed about the network structure (no parameter is required to be known beforehand). The main disadvantage is that it produces no unique solution but an aggregate of many solutions. Thus, this algorithm suffers from stability issues, which might be adjusted with our optimization algorithm.

With all community detection algorithms and the post-processing DENSOPT algorithm, we used a modularity measure Q to evaluate the quality of the community structure of a graph. Modularity serves as the objective function, as described in the previous section, during the process of calculating the communities.

Results on Real Networks

We used real networks repositories to test the assumptions with larger networks. For this, we used R package in Schoch (2021), to test the behaviour of the DENSOPT algorithm with dozens of real life networks.

This section presents the results of the experiments with our algorithms and the benchmark algorithms. Thus, we show results for ADC (Average Density per Community) optimization, and the DENSOPT algorithm.

A large amount of real known networks were used to test DENSOPT algorithm. Using two known algorithms for community detection, INFOMAP and Label Propagation algorithm, we noticed that, among some of the networks, the DENSOPT algorithm presented best results with graphs that apparently presented lower modularity of community division.

Table 4.1. Examples with real networks from "networkdata" R Package

Network	**#Nodes**	**#Edges**
pony	17	761
karate_weight	34	231
tailor_work	39	211
movie_26	52	150
highschool_boys	70	506
law_advice	71	892
law_friends	71	575
radoslaw_email	167	5784
covert_33	203	455
hall	217	2672
f2f_infectious	410	17298
dnc_corecipient	906	12085
polblogs	1490	19025
usflights	1574	28236
ucsocial	1899	20296
powergrid	4941	6594

The Table 4.1 shows the dimensions of used networks for these tests. The number of nodes and edges is shown for a variety of available networks.

The Table 4.2 shows the number of communities detected before and after the use of DENSOPT algorithm, for both used community detection algorithms. Some of these results show that, DENSOPT is not always expected to increase the number of communities. Nonetheless it is not expected to reduce this number, as it works by starting with the detected communities of the selected community detection algorithm.

Table 4.2. Number of Communities detected either with INFOMAP or Label Propagation (LP) algorithms, with and without DENSOPT algorithm processing maximization of ADC measure

Network	INFOMAP #Communities	INFOMAP+DENSOPT #Communities	LP #Communities	LP+DENSOPT #Communities
pony	7	7	1	1
karate_weight	11	11	3	3
tailor_work	8	8	2	2
movie_26	7	7	3	6
highschool_boys	20	31	11	12
law_advice	1	4	3	5
law_friends	8	8	5	5
radoslaw_email	12	12	38	38
covert_33	19	19	14	14
hall	16	16	8	8
f2f_infectious	173	173	81	81
dnc_corecipient	157	158	36	36
polblogs	351	370	509	509
usflights	112	112	88	88
ucsocial	228	239	43	43
powergrid	487	487	503	504

Table 4.3. Modularity of Community division, detected either with INFOMAP or Label Propagation (LP) algorithms, with and without DENSOPT algorithm processing maximization of ADC measure

Network	INFOMAP Modularity	INFOMAP+DENSOPT Modularity	LP Modularity	LP+DENSOPT Modularity
pony	-0.021	-0.021	0	0
karate_weight	0.195	0.195	0.434	0.434
tailor_work	0.213	0.213	0	0
movie_26	0.374	0.374	0.363	0.369
highschool_boys	0.207	0.197	0.523	0.535
law_advice	0	0.018	0.238	0.222
law_friends	0.356	0.356	0	0
radoslaw_email	0.076	0.076	0.067	0.067
covert_33	0.603	0.603	0.432	0.432
hall	0.430	0.430	0.265	0.265
f2f_infectious	0.186	0.186	0.798	0.798
dnc_corecipient	0.085	0.085	0.166	0.166
polblogs	0.400	0.399	0.395	0.395
usflights	0.216	0.216	0.070	0.070
ucsocial	0.160	0.160	0.001	0.001
powergrid	0.817	0.817	0.797	0.797

Table 4.3 shows the modularity of detected communities before and after the use of DENSOPT algorithm, for both used community detection algorithms. Some of these results show that, DENSOPT is not always expected to increase the value of modularity. It is considered statistically significant that is expected to approach this value, as it works by starting with the detected communities of the selected community detection algorithm, and solely maximizing density of the communities with few changes in structure.

Table 4.4. ADC of Community division, detected either with INFOMAP or Label Propagation (LP) algorithms, with and without DENSOPT algorithm processing maximization of ADC measure

Network	INFOMAP ADC	INFOMAP+DENSOPT ADC	LP ADC	LP+DENSOPT ADC
pony	3.417	4.167	0.000	0.000
karate_weight	2.703	2.883	1.291	1.291
tailor_work	0.653	0.678	0.150	0.150
movie_26	0.688	0.688	0.268	0.495
highschool_boys	1.158	1.090	0.762	0.838
law_advice	0.179	0.256	0.278	0.391
law_friends	0.511	0.623	0.129	NA
radoslaw_email	0.477	1.000	0.602	0.743
covert_33	0.254	0.254	0.246	0.258
hall	0.511	0.529	0.542	0.635
f2f_infectious	14.391	14.405	29.265	29.265
dnc_corecipient	0.964	0.984	0.909	0.909
polblogs	0.433	0.427	0.540	0.944
usflights	0.501	0.711	0.572	0.818
ucsocial	0.330	0.610	0.603	1.000
powergrid	0.276	0.276	0.342	0.343

Table 4.4 shows the ADC of detected communities before and after the use of DENSOPT algorithm, for both used community detection algorithms. Some of these results show that DENSOPT is expected to increase the value of ADC.

Community Optimization - Example

In the example of Figures 4.4 and 4.5, the network "Karate weight" from package "networkdata" published in Schoch (2021) is a good example where DENSOPT provides an increase of either modularity or ADC measures.

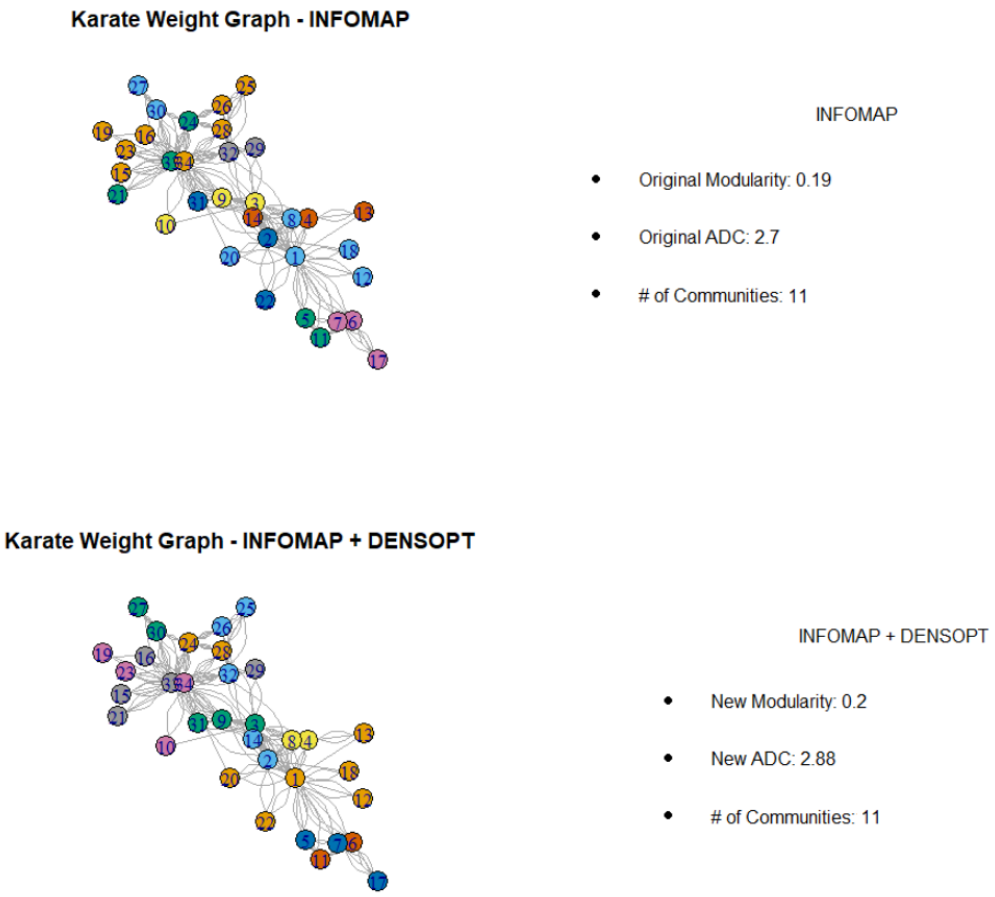


Figure 4.4. INFOMAP algorithm (top) and INFOMAP results with DENSOPT (bottom)

In Figure 4.4, INFOMAP algorithm presented lower values for modularity, and even lower values of ADC. The changes in assignment of nodes to different communities resulted in these increases, while maintaining the number of detected communities.

Considering also Figure 4.5, the Label Propagation algorithm presented minor changes in community attribution label, for only two nodes, 5 and 11. The changes in assignment of nodes to different communities resulted from the DENSOPT optimization of ADC measures.

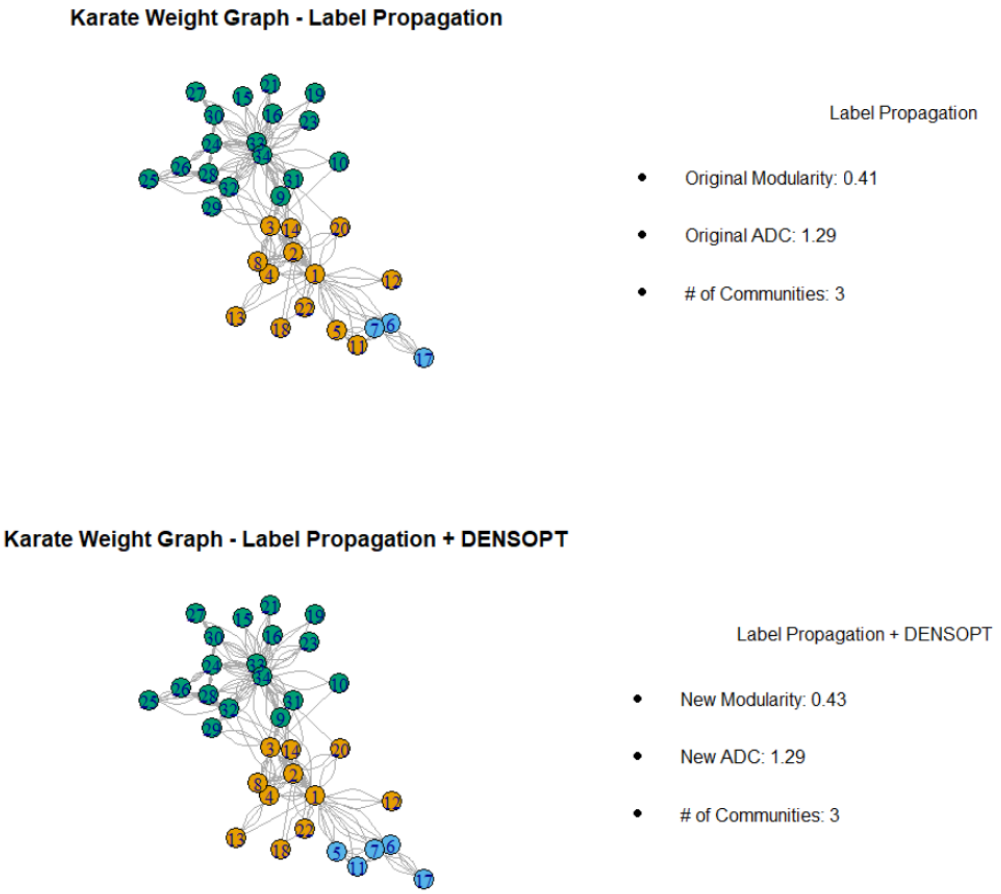


Figure 4.5. Label Propagation algorithm (top) and label propagation results with DENSOPT (bottom)

In Figure 4.5, the Label Propagation algorithm without the DENSOPT algorithm, presented lower values for modularity. The changes in assignment of nodes to different communities, by using DENSOPT, resulted in this increase, while maintaining the number of detected communities and ADC values. Thus, there are visible changes in behaviour with DENSOPT algorithm and chosen algorithm for community detection. Additionally, there was a large amount of networks where DENSOPT made no changes at all, this was specially true for networks and algorithms that presented higher values of modularity for the community divisions. This can be considered a very healthy behaviour of the DENSOPT algorithm, since if the community detection algorithm presents higher values of modularity, it can be considered that it might provide a reasonable community division for the corresponding network, and any change in that division could result in poorer analysis of the community division.

4.2.5 Summary and Contributions

In this chapter we dealt with the problem of grouping of document sets, through the use of community detection algorithms. We have contributed with the research for better ways to deal with

community detection in the context evolving networks. One critical problem is which optimization measure should be used in this process. We have contributed with a design of successful algorithms for density optimization, which maintain also good modularity values, as demonstrated by statistical tests. Resuming, our contributions are:

- Algorithm that uses sampling large networks, which leads to reasonable results and speeds up the process of identifying communities
- Design of an algorithm that exploits the measure ADC in the process of detecting communities.
- ADC is higher on average for all communities identified, without affecting modularity values.

Better ADC implies a better definition of smaller communities in the process of community detection. This leads to an improved document organization, particularly the smaller groups of similar documents.

We have published various articles on the topic discussed in this section: Cordeiro et al., 2016; Sarmento et al., 2017.

4.3 Centrality Analysis

In this section, we deal with centrality measures and the need to use them in evolving context of flux of text documents. This is important and complex. There are several examples of situations this is needed, when considering text document production. This allows the researcher to search for influential or references in a particular subject, inside the information groups, after grouping the documents.

With the explosion of social networks users, for example, like Twitter or Facebook, the networks have grown to the point where the use of batch algorithms cannot handle this data efficiently. Thus, to perform the analysis of large and changing networks, it is necessary to adopt streaming techniques and use of incremental algorithms. This way, researchers try to speed-up the process and use less memory whenever possible, by avoiding to process the full network in each iteration.

4.3.1 Introduction

Our contribution, in this section, is an efficient solution to calculate a particular centrality measure, the Laplacian centrality, in an incremental setting. We present a solution that is accurate, faster than the corresponding batch algorithm for Laplacian centrality on large networks.

4.3.2 Related Work

Incremental Centrality Measures

Due to requirements in size or dynamics of networks, some centrality measures were already adapted to be incremental. The authors of these algorithms argument their solutions are faster than the batch versions. Our objective is to achieve similar improvements, for the batch/non-incremental version of the Laplacian centrality algorithm.

Incremental Betweenness Centrality

The currently widely used *Brandes* algorithm (Brandes, 2001a), runs in $O(mn + n^2 logn)$ time, where $n = |V|$ and $m = |E|$. Nasre et al. (2013) developed an incremental algorithm to perform Betweenness centrality (BC) measures in dynamic networks. The BC score of all vertices in G is updated when a new edge is added to G, or the weight of an existing edge is reduced. Their incremental algorithm runs in $O(m'n + n^2)$ time, where m' is bounded by $m* = |E*|$, and $E*$ is the set of edges that lie on a shortest path in G. The authors explain that, even for a single edge update, their incremental algorithm is the first algorithm that is faster on sparse graphs than recomputing with the well-known static Brandes algorithm. The authors also stress that their algorithm is also likely to be much faster than Brandes on dense graphs since $m*$ is often close to linear in n. The authors explain that, with preliminary experimental results for their basic edge update algorithm on random graphs, generated using the Erdós-Rényi model, they achieve 2 to 15 times speed-up over Brandes' algorithm for graphs with 256 to 2048 nodes, with the larger speed-ups on dense graphs.

Incremental Closeness Centrality

Kas et al. (2013a) developed an incremental Closeness centrality algorithm for dynamic networks. To compute the closeness values incrementally, for streaming, dynamically changing social networks, all-pairs shortest-paths algorithm proposed by Ramalingam and Reps (1996a) was extended, such that closeness values are incrementally updated, in line with the changing shortest path distances in the network. The addition of an edge between X and Y nodes would be processed by discovering affected sources, i.e., nodes that are on the path of the new edge, and the affected sinks, i.e., the nodes that are beyond the added connection. Finally, the authors update the Closeness centrality values just for the affected nodes. The author's argument that, for Incremental algorithms, computation times can benefit from early pruning by updating only the affected parts. While the original algorithm for Closeness centrality can be performed by running an all-pair shortest paths algorithm like Floyd (1962), which results in $O(n^3)$ time complexity, Kas et al. achieve several improvements to their 2-phase algorithm. They argument the time complexity of Phase-1 to be limited by $O(||A_{ffected}||_2)$ where the subscript 2 denotes the size of two hop neighborhood of all affected nodes. The complexity of Phase-2 is dominated by the complexity of priority queue, denoted by $O(||A_{ffected}||log||A_{ffected}||)$. The authors' effort results in significant speed-ups over the most commonly used method, on various synthetic and real-life Datasets, suggesting that incremental algorithm design is a fruitful research area for social network analysts. The speed-ups they achieve with this algorithm vary regarding the topology of the network, and for synthetic networks, they conclude that the incremental algorithm is, on average, six times faster than Dijkstra's algorithm.

4.3.3 Incremental Laplace Centrality Algorithm

The original Laplace centrality algorithm proposed by Qi et al. (2012) was primarily designed for static networks (i.e., networks that do not evolve). Nevertheless, being a static algorithm, it can be used to calculate centralities in changing networks with the penalty of performing a full computation of the centralities in each one of the network snapshots. With our proposal, the same Qi et al. (2012) principles were adapted for an incremental algorithm. The proposed incremental algorithm presents better computational efficiency, by performing selective Laplace centrality calculations only for the nodes affected by the addition of edges in each one of the snapshots (i.e., it reuses information of the previous snapshot to perform the Laplace centrality calculations on the current snapshot). The algorithm is called incremental, because it avoids full calculations, in each one of the snapshots, it is prepared for the addition of edges in each one of the increments (i.e.: *incremental* algorithm).

Locality of the Laplacian Centrality

As already stated before, the Laplacian centrality metric is not a global measure, i.e., is a function of the local degree plus the degree's of the neighbors (with different weights for each). Qi et al. (2012, 2013), and the pseudo-code presented in Algorithm 1, shown that local degree and the 1st

(a) Batch Algorithm (b) Incremental Algorithm

Figure 4.6. Calculated node centralities with edge {(4, 6)} added. Dark grey nodes affected by addition of edges. Light grey nodes centralities need to be calculated due to their neighbourhood with affected nodes.

order neighbors degree is all that is needed to calculate the metric for unweighted networks. We will use the toy network example in Figure 4.6 to show the locality of the Laplacian centrality.

In this toy network we consider two distinct snapshots $G_0 = \{(1,2),(2,3),(3,5),(5,6),(5,4),(4,7),(5,7)\}$, in the second snapshot G_1 a new edge $(4,6)$ is added to the network, so $G_1 = G_0 \cup \{(4,6)\}$. In the fist snapshot (G_0), it will be needed to calculate centralities for all the nodes. In the second snapshot (G_1), once that only local degree and 1st order neighbors degree will affect the centrality values for nodes 4 and 6, The algorithm will just require to calculate the centralities for nodes 4 and 6 (i.e.: affected nodes) and nodes 5 and 7 (1st order neighbors). Due to the determinism of the algorithm, the node centrality results for the Incremental Laplace algorithm are equal to the results of the batch version of the same algorithm. This is explained by the fact that we are not dealing with probabilistic phenomena nor any randomness in the initialization. The results of both versions of the algorithm have the same values for each node in the dynamic graph of Figure 4.6, as we can see in Table 4.5. The obtained values for centrality are the same for both algorithms (expected), but for step 2 in the incremental algorithm, we only calculated the centralities of the affected nodes plus their neighbors. In total, the Batch algorithm performed 14 node centrality calculations (7 for step 1 and 7 for step 2) while the incremental achieved the same result using only 11 node centrality calculations (7 for step 1 and 4 for step 2).

Algorithm Based on the assumptions devised in the Section 4.3.3 and having by reference the batch version of the Laplacian centrality shown in Algorithm 1, a new incremental algorithm is here presented (Algorithm 6). This incremental algorithm achieves better efficiency than the batch version by performing Laplace centrality calculations only in the nodes affected by addition of

Table 4.5. Example of centrality calculation for the network presented in Figure 4.6.

Node	Batch				Incremental			
	step=1		step=2		step=1		step=2	
	centrality	Calculated	centrality	Calculated	centrality	Calculated	centrality	Calculated
1	6	yes	6	yes	6	yes	6	no
2	12	yes	12	yes	12	yes	12	no
3	18	yes	18	yes	18	yes	18	no
4	18	yes	28	yes	18	yes	28	yes
5	34	yes	38	yes	34	yes	38	yes (neighbour)
6	10	yes	20	yes	10	yes	20	yes
7	18	yes	20	yes	18	yes	20	yes (neighbour)
Total:	7 out of 7 nodes		7 out of 7 nodes		7 out of 7 nodes		4 out of 7 nodes	
	14 centrality calculations				11 centrality calculations			

edges and their 1st order neighbors (full dynamic incremental algorithm). In Algorithm 6, the incremental method `LapCentAdd()` receives as parameters a full graph (network of the previous snapshot), the lists of edges that will be added from the graph in the current iteration (A_{dd}), and the previously calculated centralities to be updated in the current iteration ($C_{entralities}$). Previously to the beginning of the `for each` cycle, the algorithm will calculate the set of nodes affected by the addition of nodes ($\mathcal{V}_s$) and the list of 1st order neighbours of affected nodes ($\mathcal{V}_f$). Laplace centrality will only be calculated for this two sets of nodes using the same centrality calculation function employed in the batch algorithm. Function `LapCentAdd()` returns the updated graph G, the updated centrality list $C_{centralities}$, and $\mid \mathcal{V}_f \mid$ the number of nodes for which new centralities were calculated for the current iteration. In the main function, initially, in the first iteration, the centralities are calculated for the full network, and this information if reused in the following iterations. In each of the incremental steps, the function `LapCentAdd()` only receives the list of edges that changed from the previous iteration. By analysis of the proposed algorithm, we can conclude that obtained efficiency can be related to the number of edges that change in each of the snapshots, and also the degree of its nodes. Higher degrees will require performing more computations of 1st order neighbours.

The complexity of the original algorithm in Qi et al. (2012) is $O(n*\Delta^2)$, where n is the number of vertices and Δ as the maximum degree ($\Delta = max_{v \in V(G)} d_v$). Thus, the total complexity for computing Laplacian centrality for network G with n vertices, m edges and maximum degree Δ would be no more than $O(n*\Delta^2)$. Nonetheless, according to Wheeler (2015) it should be something like $O(n*a)$, where a is the average number of neighbours for the entire graph, and n are the number of nodes. In the worst case, a is the maximum number of neighbours any node has in the graph. Our improvement of this algorithm brings the use of locality features of the original algorithm to lower the complexity to $O(n'*a)$, where n' are the affected nodes in each snapshot of the evolving changes of the networks. The affected nodes by addition of m' edges is given by $n' = (2*m' + 2*m'*a)$, i.e.: the sum of 2 nodes per modified edge $(2*m')$ and their respective 1st order neighbours $(2*m'*\Delta)$ or $(2*m'*a)$ with Wheeler (2015) assumption. Final time complexity is $O(2*m'*a + 2*m'*a^2)$.

4.3.4 Text Node Centrality - Laplace Centrality Solution

Results The Incremental Laplace centrality algorithm was evaluated in incremental network setups (Sarmento et al., 2017). Results for incremental networks were obtained for the High-energy physics theory citation network (Leskovec et al., 2005), using the original Laplace centrality (now on called batch) and the proposed Laplace centrality in an incremental setting configuration. The original Laplace centrality served as a baseline. This setup considered the 136 snapshots of the dataset with snapshots built by aggregating timestamps of citations monthly. In the Batch algorithm, the centralities were calculated with the full network as input for every snapshot. In the first snapshot, the full network is passed as input for the Incremental algorithm. In the following snapshots, the algorithm only receives the edges added to the network in that snapshot (incremental). The performed empirical evaluation consisted mainly of comparing each increment's run

Algorithm 6 Incremental Laplace centrality Algorithm

$V \leftarrow \{u_1, u_2, .., u_v\}$, $E \leftarrow \{(i_1, j_1), (i_2, j_2), .., (i_e, j_e)\}$
$A_{dd} \leftarrow array\{(i_1, j_1), .., (i_n, j_n)\}$
procedure LAPCENTADD($G \leftarrow (V, E), A_{dd}, \mathcal{C}_{entralities}$)
    $\mathcal{V}_s \leftarrow \{\}$
    for each *edge* **in** A_{dd} **do**
        $\mathcal{V}_s \leftarrow \mathcal{V}_s \cup edge.source() \cup edge.destination()$
        $G.add_edge(edge)$
    end for
    $\mathcal{V}_f \leftarrow \{\}$
    for each *node* **in** $\mathcal{V}_s$ **do**
        $\mathcal{V}_f \leftarrow \mathcal{V}_f \cup G.neighbors(node)$
    end for
    $\mathcal{D}_{egrees} \leftarrow G.degrees()$
    for each v **in** $\mathcal{V}_f$ **do**
        $\mathcal{N}_{eighbors} \leftarrow G.neighbors()$
        $loc \leftarrow \mathcal{D}_{egrees}[v]$
        $nei \leftarrow 2. \sum_{i=1}^{\mathcal{N}_{eighbors}} \mathcal{D}_{egrees}[i]$
        $\mathcal{C}_{entralities}[v] \leftarrow (loc^2 + loc + nei)$
    end for
    return $\mathcal{C}_{entralities}$, $\mid \mathcal{V}_f \mid$, G
end procedure
procedure MAIN
    $\mathcal{D}_{ataset} \leftarrow \{G_0, G_1, ..., G_n\}$, $A_{dd} \leftarrow \{A_0, A_1, ..., A_n\}$
    $\mathcal{C}_{entralities}, \mathcal{N}_{um}\mathcal{C}_{entralities} \leftarrow$ LAPCENT(G_0) ▷ initial step in the full network
    $G \leftarrow G_0$, $i \leftarrow 1$
    while $(i \leq |\mathcal{D}_{ataset}|)$ **do** ▷ calculate centralities for the increments
        $\mathcal{C}_{entralities}, \mathcal{N}_{um}\mathcal{C}_{entralities}, G \leftarrow$ LAPCENTADD($G, A_{dd}[i], \mathcal{C}_{entralities}$)
    end while
end procedure

times (duration of each increment and cumulative execution time). Additionally, the size of the network (number of nodes and edges), the number of added edges in each snapshot, and the total number of calculated centralities for each snapshot was also registered. Remarks that the batch algorithm will always need to calculate centralities for all nodes in the snapshot. It is expected that the incremental algorithm only performs centrality calculations for the affected nodes. In the end, an analysis of the total speed-up ratio obtained in each of the steps is added. We used an Intel (R) Core (TM) i7-4702MQ processor computer with 8 GBytes and SSD HDD for all the experimentation and development. Three runs per algorithm/ dataset were performed with values presented in graphs as the average values of each of those three runs.

Incremental Networks In this subsection, we introduce the reader to the results obtained by the incremental setup of the algorithm. The results are presented in the following Figure 4.7 and are related to the High-energy physics theory citation network (Leskovec et al., 2005) in an incremental network setting where no edges are removed from previous snapshots. Figure 4.7 – network Size shows the variation of the network over the 136 snapshots regarding the number of nodes and number of edges. Figure 4.7 – # Added Edges, shows the number of added edges in each snapshot. Notice that the total number of edges in this dataset increases over time. A few

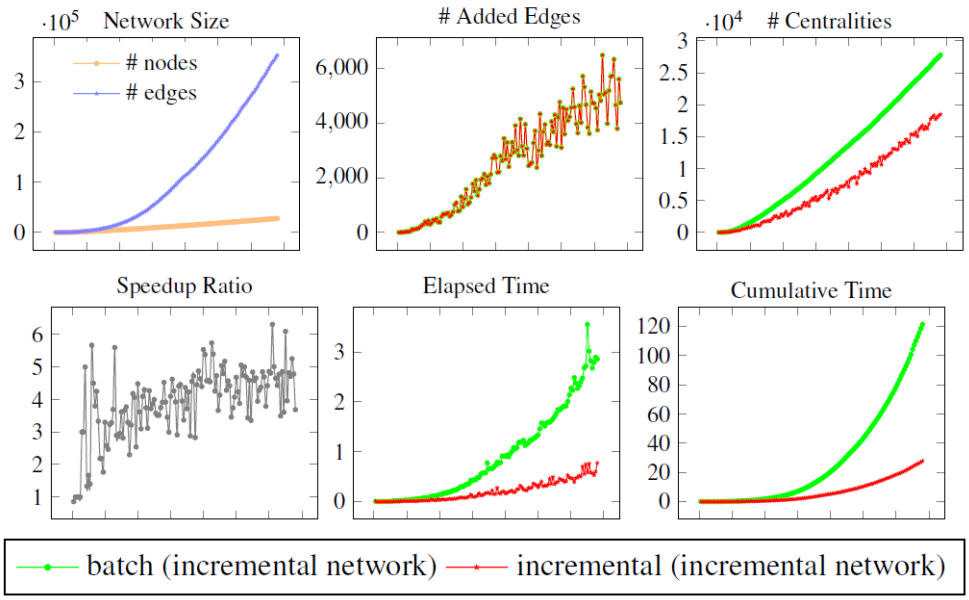


Figure 4.7. Results for the Cit-HepTh network (incremental only)

edges are added to the network in the first snapshots, but at the final snapshots, more than 6000 edges are added in each snapshot.

Figure 4.7 – Speed-up Ratio shows that the number of calculated centralities in the incremental version is much lower than in the batch version. The batch version requires to compute centralities for all nodes in the snapshot as can be seen on Figure 4.7 – # Centralities. Figure 4.7 – Elapsed Time shows the time required to perform the centralities measurements in each increment. This Figure also shows that the incremental version is more efficient and deals better with both increases in the size of the network and an increase in the number of added edges. This is confirmed by the total time required for processing the whole network: 27,806 seconds of the incremental version compared to the 121,345 seconds of the batch.

It is clear that, as the number of added edges increases, the processing elapsed time of the batch version of the algorithm grows much faster than the incremental algorithm. Thus, regarding the speed-up ratio, we achieve a speed-up of up to 6 times the processing time of the batch version with an incremental network (Figure 4.7 – Speed-up Ratio).

4.3.5 Summary and Contributions

The methods that deal with centrality analysis in evolving networks are normally considered to be rather time consuming and with most networks, of high complexity.

Our contribution with Incremental Laplacian centrality proved to be of lower complexity and, at the same time, very efficient and effective.

Having an efficient algorithm helps also to design better and more efficient methods for the task of folding, i.e., substituting a network by a node) and the keywords that characterize the nodes representing groups of documents. Besides, these algorithms can also be used to identify central

author of a particular group of documents, which is useful in the search for the right document that satisfies users' criteria. We have published various articles on the topic discussed in this section: Cordeiro et al., 2016; Sarmento et al., 2017.

4.4 Incremental Graph-based Keyword Representation

4.4.1 Introduction

Automatic keyword extraction from text is an essential area of research. It has been used to automatically summarize documents. From enterprises' document production to the analysis of users' social networks posts, for example, in Twitter or Facebook, the appropriate summarizing of text brings new possibilities to better target advertising and recommendations to users. It is also used to find similarities between authors and improve enterprise departments workflow. The majority of current systems are prepared to be used statically. Thus, with the explosion of large-scale document sets, researchers felt a considerable increase in the need for systems to analyze this evolving and even more extensive datasets. The improvement of previous algorithms or the adaptation of these systems to cope with incoming text is, therefore, imperative nowadays. Although not many researchers were concerned with the issue of incremental keyword extraction, some of them are starting to implement new algorithms for this issue like in Shin et al. (2014). Incremental algorithms have been developed extensively in several areas of Data Analysis. Notably, those algorithms that are graph oriented have been proving to be faster or more efficient when their incremental adaptations are compared to their static versions. Our contribution, in this section, is the transformation of a well-known algorithm for keyword extraction, TextRank (Mihalcea and Tarau, 2004a). Taking into account that this algorithm is a graph-based approach to keyword extraction, we updated it to be used in a streaming setting. We validate the results by comparing the behavior and processing time with the original algorithm.

4.4.2 Related Work

Space Saving Top-K

The problem of finding the most frequent items, in our case, keywords, in a data stream S of size N is the problem of discovering the elements e_i whose relative frequency f_i is higher than a user-specified support ϕN, with $0 \leq \phi \leq 1$ (Gama, 2010). Given the space requirements that exact algorithms addressing this problem would need (Charikar et al., 2002), several algorithms have been proposed to find the top-K frequent elements, being roughly classified into *counter-based* and *sketch-based* (Metwally et al., 2005). *Counter-based* techniques keep counters for each element in the monitored set, which is usually a much smaller than the entire set of elements. When an element is identified as not currently being monitored, various algorithms take different actions to adapt the monitored set accordingly. *Sketch-based* techniques provide less rigid guarantees, but they do not monitor a subset of elements, providing frequency estimators for the entire set. Simple *counter-based* algorithms, such as *Sticky Sampling* and *Lossy Counting*, were proposed by Manku and Motwani (2002), which process the stream in compressed size. They have the disadvantage of keeping a lot of irrelevant counters. *Frequent* (Demaine et al., 2002) keeps only k counters for monitoring k elements, incrementing each element counter when it is observed, and decrementing all counters when an unmonitored element is observed. Zeroed-counted elements are replaced by

new unmonitored elements. This strategy is similar to the one applied by *Space-Saving* (Metwally et al., 2005), which gives guarantees for the *top-m* most frequent elements. This was the algorithm we selected to extract the most frequent keywords in the stream of text.

TextRank for Text Streams

In this section, we explain our proposal for two versions of an Incremental TextRank. We propose a Window-based version and a more complex incremental version. In Figure 2.1 we showed the sequence of operations the text goes through, in the original TextRank. Although some parts of it are eminently evolving as, for example, the graph construction from the text words, other parts are not prepared for streaming or incremental approach.

Keywords Extraction

Problem

The current process of extracting keywords from the authors' text is a slow process and it is performed with TextRank batch algorithm.

What should we do?

- should we update text rank to be incremental?
- should we use Online Topic Modeling (OLDA) to extract authors and communities keywords?

Local or Global "Online Topic Modeling"

One of the publications that study online topic modeling, more specifically the semantic model smoothing with stream data, is Liu et al. (2008). In this publication, the authors stress that many previous approaches use the word extraction method and single word vector as the document features. However, they suffer from the context-insensitivity problem. The terms in these models may have ambiguous meanings. In contrast, the semantic smoothing model uses multiword phrases as topic signatures (document features). The authors in Liu et al. (2008) continue and give an example, the multiword phrase *"fixed star"* (denotes a celestial body), and has clearer meaning than the single word *"star"* (denotes either a celestial body or a pop star). After phrase extraction, the training process determines the probability of translating the given multiword phrase to terms in the vocabulary. For example, if the word *"planet"* frequently appears in the documents whose topic contains *"fixed star"*, then *"fixed star"* and *"planet"* must have some specific relationship. The translation model finds such a relationship and assigns a degree to describe it. In the following process (e.g., clustering), if we encounter a document that contains the topic signature *"fixed star"* (but not *"planet"*), we can also assign a rational probability count to the word *"planet"* for the document. For each phrase t_k, it would have a set of documents (D_k) containing that phrase. Since not all words in D_k center on the topic signature t_k, we assume D_k is generated by a mixture language

model (i.e., all terms in the document set are either translated by the given topic signature model $p(w|t_k)$ or generated by the background collection model $p(w|C)$). Equations 4.2 to 4.4 are used to iteratively compute the translation probabilities like in Zhang et al. (2006); Zhou et al. (2006).

$$p(w|D_k) = (1-\beta)p(w|t_k) + \beta p(w|C) \tag{4.2}$$

$$p^{(n)}(w) = \frac{(1-\beta)p^{(n)}(w|t_k)}{(1-\beta)p^{(n)}(w|t_k) + \beta p(w|C)} \tag{4.3}$$

$$p^{(n+1)}(w|t_k) = \frac{c(w,D_k)p^{(n)}(w)}{\sum_i c(w_i,D_k)p^{(n)}(wi)} \tag{4.4}$$

Here, β is a coefficient accounting for the background noise, $p(w|t_k)$ denotes the translation model for topic signature t_k, $c(w,D_k)$ is the frequency count of term w in document set D_k (means the number of appearance of w in D_k), n is the number of iterations, and C denotes the background collection, which is the set of all word occurrences in the corpus. In practice, the algorithm is used to estimate the translation model in equations 4.3 and 4.4. The cluster model with semantic smoothing (or referred to as semantic smoothing model) is estimated using a composite model $p_{bt}(w|c_j)$, which means the likelihood of each vocabulary word w generated by a given document cluster c_j after smoothing. It has two components: a simple language model $p_b(w|c_j)$ and a topic signature (multiword phrase) translation model $p_t(w|c_j)$. The influence of two components is controlled by the mixture model's translation coefficient (λ).

$$p_{bt}(w|c_j) = (1-\lambda)p_b(w|c_j) + \lambda p_t(w|c_j) \tag{4.5}$$

$$p_b(w|c_j) = (1-\alpha)p_{ml}(w|c_j) + \alpha p(w|C) \tag{4.6}$$

$$p_t(w|c_j) = \sum_k p(w|t_k)p_{ml}(t_k|c_j) \tag{4.7}$$

In simple language model 4.6, α is a coefficient controlling the influence of the background collection model $p(w|C)$ and $p_{ml}(w|c_j)$ is a maximum likelihood estimator cluster model. They can be computed using equations 4.8 and 4.9. In translation model 4.7, t_k denotes the topic signatures (multiword phrases) extracted from documents in cluster c_j. The probability of translating t_k to individual term (word) is estimated using equations 4.3-4.4. The maximum likelihood estimator of t_k in cluster c_j can be estimated with 4.10. In equations 4.8, 4.9, and 4.10, $c(w,c_j)$ denotes the frequency count of word w in cluster c_j, $c(w,C)$ is the frequency count of word w in the background corpus, and $c(t_k,c_j)$ is the frequency count of topic signature t_k (multiword phrase) in cluster c_j. The function of the translation model is to assign reasonable probability to an unseen core word in a given cluster.

$$p_{ml}(w|c_j) = \frac{c(w,c_j)}{\sum_{w_i \in c_j} c(w_i,c_j)} \tag{4.8}$$

$$p(w|C) = \frac{c(w,C)}{\sum_{w_i \in C} c(w_i,C)} \tag{4.9}$$

$$p_{ml}(t_k|c_j) = \frac{c(t_k,c_j)}{\sum_{t_i \in c_j} c(t_i,c_j)} \tag{4.10}$$

Due to the likelihood of word w, generated by a given cluster $p(w|c_j)$ can be obtained by the cluster model with semantic smoothing, the remaining problem for the clustering of a text document is how to estimate the likelihood of a document d generated by a cluster. The log-likelihood of document d generated by the j-th multinomial cluster model is described in equation 4.11, where $c(w,d)$ denotes the frequency count of word w in a document d and V denote the vocabulary.

$$\log p(d|c_j) = \sum_{w \in V} c(w,d) \log p(w|c_j) \tag{4.11}$$

Although OLDA is a feasible solution for the problem of finding keywords that are significant for an individual author or globally significant for a given group or cluster of authors, the problem we have is mainly directed to complex networks and their evolution. Thus, keeping coherent the original thought of research in graph-based representation of text and documents, we considered that implementing an Incremental TextRank algorithm would provide a similar approach to the currently used solution of the Affinity Miner prototype, which uses the batch version of the TextRank algorithm. Nonetheless, the incremental approach would allow for more efficient retrieval of automatic keyword extraction since the runtime of the batch TextRank algorithm is considerably high. The incremental approach would, by using sliding windows or forgetting factors, allow to work with considerably less information than by using all the data, typical in a batch approach.

Incremental PageRank

The reader should note that the original TextRank algorithm uses the original PageRank algorithm in one of its phases. We will show, later in this document, that a significant improvement of TextRank might be possible by transforming some of its phases algorithms to a streaming approach. Revisions of these algorithms were already developed as we will see. Particularly, there have been several attempts to improve the original PageRank algorithm. The purpose of several of these improvements was to adapt this algorithm for streaming data. In Desikan et al. (2005), the authors provided a solution for the update of nodes' PageRank values in evolving graphs. The algorithm explores the fact that the web evolves incrementally and with small changes between updates. Thus, according to the author's proposal, we should focus only on the pages that change their PageRank values due to the addition or removal of new pages/nodes and the correspondent change in the affected nodes by this change. In Desikan et al. (2005), researchers continued with

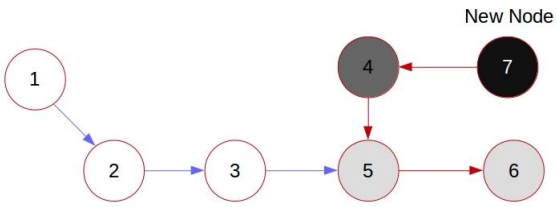


Figure 4.8. Incremental PageRank workflow - Addition of a Node

this work and stress the need to find the nodes that are affected by the changes in the graph and find three categories of nodes:

1. Nodes that are directly affected by changes in the graph, e.g., the nodes that are outgoing nodes of added or removed nodes;
2. Nodes that are indirectly affected. For example, the neighbors of the nodes in 1, and consequently their outgoing neighbors also;
3. The nodes not affected by the changes, but that have their PageRank value updated anyway, due to a simple graph scale update.

Figure 4.8 shows a small network with these types of nodes. In this figure, changes in the graph are represented by added node 7. In this figure, we can also see nodes of type 1 colored dark gray, nodes of type 2 colored light gray, and nodes of type 3 with white color.

Window-based Streaming TextRank

The first version we developed was based on the concept of using a sliding window to iterate through the stream of data. Thus, with this version, the current snapshot processing is unrelated to the previous snapshots processing. Therefore, each chunk of data from the stream is treated independently from snapshot to snapshot. In each iteration of the Window-based TextRank, we only process the current piece of data, we pre-process this data, and we do the standard word selection by considering only nouns and adjectives. Additionally, when we do the graph construction we just build it considering the selected words for the current chunk of data. Thus, the next phase of the algorithm, i.e., the use of PageRank algorithm to detect strong words in the text also only processes the words in the current snapshot graph. Again, regarding post-processing, it is done considering the words in the text available just for the current snapshot. As previously stated, the *Top-K* phase of this Window-based streaming version, uses the *Top-K* space saving implementation that is based on the concept of landmark windows. Thus, the *Top-K* list of keywords is the only information that transits from snapshot to snapshot processing for the streaming data.

4.4.3 Incremental TextRank

Figure 4.9 shows our proposal for the upgraded Incremental TextRank. We have changed the workflow of the original algorithm to be able to cope with evolving text streams. The input of a

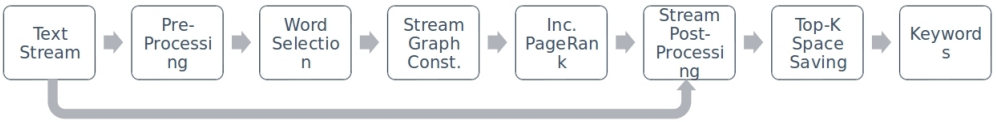


Figure 4.9. Incremental TextRank workflow

new stream of text starts to be pre-processed in the same way that was used for the original algorithm. At this stage, as in the original algorithm, we filter only the desired words as, for example, nouns and adjectives. Then, the workflow proceeds with the word graph update, considering the previous graph from previous updates, incrementally, with edge weight updates for the repetition of words sequences, and also the addition of new nodes/words if new words happen to exist in the evolving text stream. Then, we use Incremental PageRank algorithm to retrieve the words that have a higher importance in the text stream. These keywords are applied yet to a final post-processing stage of the stream to obtain a reasonable combination of keywords as in the original algorithm. Finally, as we do not need to accumulate unnecessary keywords from previous stream updates, and as we want the proposed system to be sensitive to the occurrence of new keywords in the text stream, we process the keywords in a stage where we maintain and update a *Top-K* list of keywords with the space-saving algorithm previously described.

The *Top-K* algorithm application, based on landmark window, enables an efficient approach for large-scale text datasets. It focuses on the relevant keywords and discards less extracted keywords through time or keywords extracted from earlier snapshots. The alternative option for sliding windows (Gama, 2010) would not be appropriate for the *Top-K* approach, since it may remove relevant keywords from the extracted *Top-K* keywords. Those keywords might yet be included in the *Top-K* keyword list we wish to maintain. In our scenario, *Top-K* representation of text data streams implies knowing the K keywords of the simulated data stream from the database of publications and received after the post-processing stage of the streaming TextRank algorithms.

Functions *getTopKkeywords* and *updateTopkeywordsList* in 7 implement the *Space-Saving* algorithm. As the network evolves over time, new *Top-K* keywords are added to the list. keywords that exit the *Top-K* list of keywords are removed from the *Top-K* list.

Algorithmic Analysis

Algorithm 9 is a conceptual description of the procedure each document of a input stream would be subject to when ITR is employed. For ODS (Algorithm 8), each input document is unprecedented, so that the condition in 1 is always true;

In the several documents stream (SDS) variant, there are some updates necessary regarding *PageRank* and $Top-K$ keywords.

To assess the asymptotic computational complexity of the algorithm, the following variables should be considered: PageRank algorithm has a complexity of $O(n+m)$ and Top-K has a complexity of $O(K*(logk))$, where K is the minheap window size and k is the number of keywords to retrieve from the stream. These are the dominant complexity parcels of the ITR algorithm,

Algorithm 7 *Top-K* Pseudo-Code for Keyword Extraction

Input: *start*, *k_param*, *keywords* ▷ k parameter and previous Phase' extracted keywords
Output: *keywords*
1: $R \leftarrow \{\}$ ▷ Current keywords
2: $E \leftarrow \{\}$ ▷ keywords currently in the keywords' bucket
3: $R \leftarrow$ getkeywordsFromPost-ProcessingStage(*start*)
4: **while** ($R <> 0$) **do**
5: **for all** *keywords* $\in R$ **do**
6: $before \leftarrow$ GETTOPKKEYWORDS(*k_param*)
7: UPDATETOPKEYWORDSLIST(*keyword*)
8: $after \leftarrow$ GETTOPKKEYWORDS(*k_param*)
9: $maintained \leftarrow before \bigcap after$
10: $removed \leftarrow before \setminus maintained$
11: **for all** *keywords* $\in after$ **do** ▷ add *Top-K* keywords
12: **if** *keyword* $\subset$ *words* **then**
13: ADDKEYWORDTOBUCKET(*words*)
14: $E \leftarrow E \bigcup \{keyword\}$
15: **end if**
16: **end for**
17: **for all** *keyword* $\in$ *removed* **do** ▷ remove non *Top-K* keywords
18: REMOVEKEYWORDFROMBUCKET(*keyword*)
19: **end for**
20: **end for**
21: **end while**
22: *keywords* $\leftarrow E$

nonetheless, for SDS, PageRank complexity can be simplified to n' changed $tokens$ and m' changed edges, respectively, for the changed edges in Dgt. Therefore, the proposed algorithm has a time complexity of $O(n+m+logk)$ for ODS, and $O(n'+m'+K*(logk))$ for SDS.

4.4.4 Case Study and Evaluation

This section presents the method to analyze both window-based and Incremental TextRank, regarding its comparison with the original TextRank. We introduce the reader to the data used for the tests. Finally, we describe the methodology to perform the comparison tests. We also introduce the metrics used to compare the algorithms regarding obtained keywords or keyphrases.

Description of the Data

In this case study, for algorithms efficiency measurements, we selected a dataset publicly available for research purposes by Corney et al. (2016). This dataset has a high amount of information, including Reuters news and several articles extracted from Blogs. The data contains articles titles, content, type of publication (which might include news or Blog posts), the source institution, and also date and time of publishing. This structured data's high quality and high organization make it a good source for text mining or NLP tasks. We selected all Reuters news, from the 1st to the 30th of September 2015. This corresponds to 400 news articles. We could choose between the news Titles or the news content to analyze, regarding the used text. We choose the news content in all

Algorithm 8 ITR document Processing Pseudocode (ODS)

Require: doc, the document to be processed
Require: Dgt, a graph per document with each documents' text tokens
Require: $Dtopk$, a Top-K strutcture to store each documents' Top-K keywords
if there is no node regarding doc in Dgt **then**
    Add node $token$ to Dgt
    Calculate the initial PageRank of $token$ regarding doc
    Calculate Top-K keywords of doc considering each document initial PageRank table
end if
Preprocess doc
for each token $t \in doc$ **do**
    if there is no node regarding t in Dgt **then**
        Add node t to Dgt
    end if
    if there is no edge in Dgt between nodes t and tn **then**
        Add an edge between t and tn to Dgt
    end if
    Calculate PageRank of t regarding doc
    Calculate list of k keywords retrieved from the PageRank table for tokens of doc
end for

our studies. For qualitative measurements, we selected R&D publications from Czech Republic researchers. The dataset was publicly available in RDICCR (2015) when accessed online in 2015. This is a complete source of information, and there was information for dozens of years from around 1985 until 2015. This dataset provides annotated keywords for each publication. During 2015, while this dataset was publicly available, we exported the data for several years. We selected conference papers and book chapters only. Therefore, the number of publications was reduced from around 25000 to 5110.

Methodology

The Window-based TextRank and the Incremental TextRank algorithm were evaluated in an incremental setup for both the developed versions. Additionally, both algorithms were tested and evaluated regarding Processing Efficiency and quality.

Processing Efficiency Both streaming TextRank versions were obtained from several increments of text news, compared with the original batch algorithm, using the original TextRank (now on called batch). The proposed TextRank versions, the Window-based implementation (now on called Window-based), and the incremental (now on called incremental) were tested regarding efficiency in an incremental setting configuration. The original batch results served as a baseline. In this setup, six snapshots of news were considered. In total, the first 20 days of September 2015, corresponding to 300 news articles, were passed as input. The snapshots were built by aggregating publications daily by using the timestamps available in the dataset. In the Batch algorithm, the keywords were extracted for every snapshot, having all daily news from day 1 to the current day as input. For the incremental algorithm, the first snapshot receives only 15 days of news articles as input. The algorithm only receives the set of publications text added to the Corpus in that particular

Algorithm 9 ITR document Processing Pseudocode (SDS)

Require: *doc*, the document to be processed
Require: *Dgt*, a graph per document with each documents' text tokens
Require: *Dtopk*, a Top-K strutcture to store each documents' Top-K keywords
if there is no node regarding *doc* in *Dgt* **then**
    Add node *token* to *Dgt*
    Calculate the initial PageRank of *token* regarding *doc*
    Calculate Top-K keywords of *doc* considering each document initial PageRank table
end if
Preprocess *doc*
for each token $t \in doc$ **do**
    if there is no node regarding t in *Dgt* **then**
        Add node t to *Dgt*
    end if
    if there is no edge in *Dgt* between nodes t and tn **then**
        Add an edge between t and tn to *Dgt*
    end if
    Update the Incremental PageRank of t regarding *doc* considering only the new and changed tokens
    Update the list of keywords retrieved from the PageRank table for tokens of *doc*
    Update Top-K keywords of *doc* considering each document seen changes in keywords
end for

day snapshot (incremental). The empirical evaluation performed consisted mainly of comparing run times of each increment (duration of each increment and cumulative execution time) for all versions of TextRank. Additionally, the size of the Corpus (number of unique words) and the total number of extracted keywords for each snapshot was also registered. Note that the batch algorithm will always need to extract keywords for all the text. At the same time, it is expected that the Window-based version only works with the current snapshot data, and not regarding previous snapshots information except the *Top-K* list of keywords that needs to be updated between snapshots. Additionally, the incremental algorithm only performs calculations and keyword extraction for the current snapshot, particularly for the affected words. Nonetheless, as previously explained, the incremental versions take graph and corpus information from previous snapshots into account. In the end, an analysis of the total speed-up ratio obtained in each of the steps is added.

Qualitative Comparison We used the Czech Republic R&D Dataset with annotated keywords for qualitative results. For this purpose, we selected only articles in proceedings for 2013. Then, we aggregated the publications by month in 12 snapshots. We selected the abstracts for our automatic keyword extraction tasks for this amount of publications. On average, the abstracts have 240 words. Finally, we calculated the 12 measures of similarity between annotated and extracted keywords of all the versions of TextRank. We use a similarity measure to compare the author's keywords and the model results for keywords extraction.

This similarity measurement is more appropriate to test unsupervised keyword extraction algorithms. For example, imagine the annotated keyword "car maintenance" for a text. We automatically extracted the keywords "car washing" or "car cleaning" from the text. In this situation, if we used Precision, Recall, and F measure, we would have no idea that the extraction algorithm is

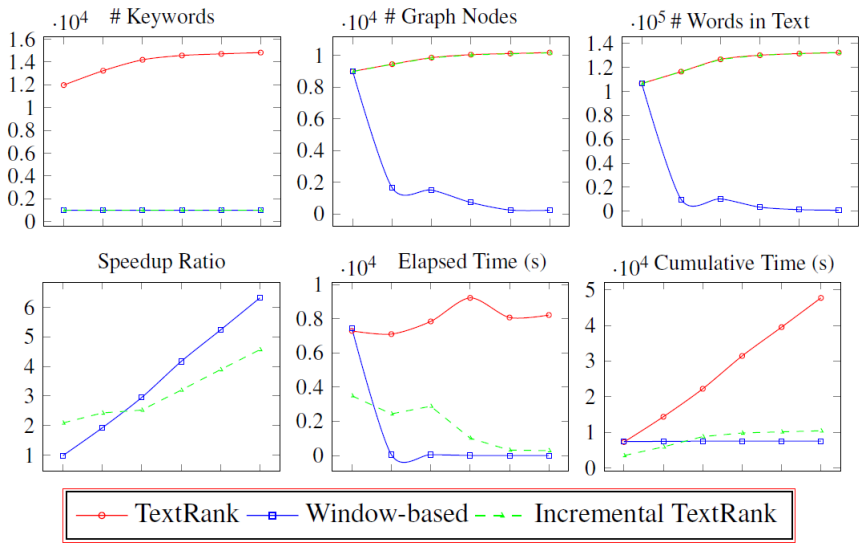


Figure 4.10. Efficiency results with the Reuters News Dataset (for all algorithms)

extracting concept keywords that are not that much far from the annotated keywords. Thus, by decomposing n-gram keywords, we can detect these normal situations when dealing with annotated keywords. Particularly for this example, we would have a value of similarity of:

$$similarity_{K,M} = \frac{(dim(w_K \cap w_M))^2}{dim(w_K)dim(w_M)} = \frac{1^2}{2*2} = \frac{1}{4} = 0.25 \tag{4.12}$$

where $dim(w_K \cap w_M)$ is the intersection of both groups, i.e., the number of words that appear in both the results and the keywords. $dim(w_K)$ and $dim(w_M)$ are the total number of words the author's keywords group has and the method (for all TextRank algorithms) provide, respectively. We used an Intel® Core™ i7-7700HQ CPU @ 2.80GHz x 8 processors computer with 16 GBytes of RAM, SSD HDD, and Ubuntu for all the experimentation and development 16.04.3 LTS 64-bit OS. Three runs per algorithm were performed. The values presented in the following charts in this section are the average values of those three runs.

Results

We present the results for all algorithm versions here in this section. All algorithms are compared with the Batch version of TextRank. The comparisons regard qualitative measurements and efficiency.

Processing Efficiency

Results on Reuters News dataset (Figure 4.10)

Figure 4.10 shows the results for the Reuters News data. From "#Graph Nodes" and "#Words in Text" we can, as expected, conclude that the number of TextRank graph nodes is similar for the batch and incremental version and variable for the Window-based version of the algorithm.

(#Extracted keywords) shows that the batch version retrieves a growing number of keywords while our streaming implementations retrieve the top-1000 keywords.

(Elapsed Time) shows that all algorithms have variable elapsed time for each of the iterations. Nonetheless, both streaming algorithms have a clear decreasing elapsed time compared to the batch algorithm, which tends to increase elapsed time as the quantity of the data increases. Otherwise, both streaming versions tend to achieve much lower values. For example, on iteration 4, the Window-based algorithm delivers a very low elapsed time, almost instantly processing incoming data. The incremental version takes around 100 seconds to perform the same incoming data processing, and the batch version takes approximately 900 seconds.

(Cumulative Time) shows a clear advantage of using incremental or Window-based versions of the algorithm regarding total elapsed time/cumulative time when the data is growing, and there is a need to analyze all data since day 1. In the last iteration, both Window-based and the incremental versions take around 10000 seconds to process the 20 days of data. Otherwise, if we used the original TextRank algorithm to process all the data and the incoming text, it would take almost 50000 seconds to achieve results.

(Speed-up Ratio) shows that both streaming algorithms, the incremental and the Window-based, show growing and significant speed-ups when we compare cumulative time. At the end of the stream, i.e., the last iteration, the Window-based approach provides a speed-up of more than six times. The incremental version achieves a speed-up of almost five times the speed of processing provided by the original TextRank algorithm.

Qualitative Comparison Regarding qualitative comparison, the Table 4.6 shows the similarity values between extracted keywords and accumulated annotated keywords as the abstracts text data evolves. We selected the top-1000 keywords of the batch algorithm regarding keyword weight. Therefore, we compare the similarity values for these 1000 keywords obtained from the batch, Window-based and incremental algorithms. To measure the sensitivity to capture recent keywords, we repeat the test, but this time we measure the similarity of extracted keywords with each snapshot annotated keywords. This way, we can measure the sensibility of all algorithms to the current keywords in the stream. Table 4.7 shows these results. As we expected, in both tables of results, the incremental version of the algorithm presents a higher similarity average. This is even more pronounced in the second table as expected. As expected in the second table, we should also note that the maximum similarity values for the streaming versions of the algorithms are obtained in the last snapshots. This is explained by the higher sensibility of both proposed algorithms to more recent keywords.

Incremental TextRank Regarding processing efficiency, it is clear that if the TextRank user is dealing with streaming data, the use of incremental or Window-based versions of the algorithm

Table 4.6. Qualitative measures of similarity with accumulated annotated keywords

Algorithm	Avg.	Sd	iter1	iter2	iter3	iter4	iter5	iter6	iter7	iter8	iter9	iter10	iter11	iter12
TextRank	0.168	0.108	**0.023**	0.025	0.032	0.067	0.130	0.254	0.308	0.300	0.260	**0.218**	0.204	**0.193**
Window-based	0.114	0.059	**0.023**	0.045	0.042	0.069	0.125	0.173	0.186	0.207	0.136	0.127	0.134	0.102
Incremental	**0.198**	**0.125**	0.011	**0.046**	**0.046**	**0.124**	**0.189**	**0.308**	**0.350**	**0.398**	**0.304**	0.217	**0.206**	0.181

Table 4.7. Qualitative measures of similarity with current snapshot annotated keywords

Algorithm	Avg.	Sd	iter1	iter2	iter3	iter4	iter5	iter6	iter7	iter8	iter9	iter10	iter11	iter12
TextRank	0.160	0.118	**0.023**	0.008	0.003	0.026	0.086	0.221	0.280	0.254	0.267	0.250	0.270	0.235
Window-based	0.155	0.103	**0.023**	**0.043**	0.016	0.047	0.099	0.183	0.202	0.267	0.211	0.239	0.299	0.237
Incremental	**0.254**	**0.181**	0.011	**0.032**	**0.023**	**0.105**	**0.133**	**0.311**	**0.331**	**0.480**	**0.336**	**0.371**	**0.425**	**0.487**

is imperative. The Window-based version presents a more pronounced speed-up, and this is expected as the processed data is variable instead of incremental. Still, regarding cumulative time, the incremental version of the algorithm has a learning curve. Thus, as the data arrives and the graph construction and update happens, the algorithm will eventually need fewer updates of graph structure in the future. This typically results in a logarithmic type of cumulative time curve. This results from less and less needed addition of nodes, resulting in the need to only update the weight of edges between nodes, i.e., the counting of words pairs. Regarding qualitative results of both streaming proposals, we would like to note that, if the researcher using our proposed algorithms is not too preoccupied with the small decrease of quality of extracted keywords from the stream, he/she can use the Window-based version the algorithm for fast extraction. Nonetheless, if the researcher needs assured quality of extracted keywords, the incremental version proved to be a better solution when compared to the batch and the Window-based approach for keyword extraction. Additionally, the effect of increasing K in both these *Top-K* approaches would intuitively increase the quality results as it is expected that, in Table 4.6, as the annotated keywords amount increase, we continue to test only 1000 keywords extracted with the algorithms. Accordingly, in Table 4.6, the quality values decrease due to this characteristic, approximately after iteration 8, until the end of the stream, and for all algorithms.

4.4.5 Summary and Contributions

This section introduces a Window-based and an incremental version of the TextRank system (Sarmento et al., 2018b). The Window-based version proved similar in quality and much faster than the original TextRank algorithm in a streaming context. Through the careful transformation of the original algorithm to an incremental version, we achieve a more efficient version with better quality. Additionally, we introduce a *Top-K* feature in our improvement, which allows storing the more relevant keywords. We have published various articles on the topic discussed in this section, we should underline the following article: Sarmento et al., 2018.

4.5 Network Visualization

The folding and unfolding of Graph Nodes is directly related to previously mentioned developments and might help to use the folding in centroids or prototypes (more information in Chapters 4.2 and 4.3) in a situation of visualization of thousands of authors. Thus, the visualization would provide a summary of the full network.

In a previous section regarding the Affinity Miner prototype (Chapter 3), we present a figure about the folding and unfolding of communities in nodes and vice-versa. The problem here is that this new version would be prepared to process data streams. The software already implemented is a batch and static version of what we intend to develop.

Another possible approach as we fold and unfold nodes is the possibility of having, for example, the unfolding of the ego-network of the centroid or prototype. A streaming approach to avoid massive Ego-Networks as the streaming evolves would be to use node forgetting factors like, for example, the one we proposed in our previous work (Sarmento et al., 2015b).

Problem

The visualization of large datasets and their similarity matrix can have low comprehensibility. We wish to show the evolution of the network or a summarized representation of the original network.

What should we do in the streaming approach?

- use folding and unfolding of network nodes and the detected communities?
- display only the centroids of communities?
- display only the connections with more weight?
- use network pruning or Ego-Networks?
- what should be the window temporal scale for the visualization? One year? one day? one month?

4.5.1 Introduction

The analysis of social networks that emerge from a set of phone calls using regular services from a telecommunication service provider is a demanding problem. In these networks, a node represents a user, and a phone call is represented by an edge between two nodes. Common networks in telecommunication services have millions of nodes and billions of edges where data from started phone calls flows at high speed. This is the reason why these networks are complex and difficult to analyze. Sampling from a large network is known to be a hard problem to solve with typical hardware or software. State-of-the-art software and hardware reveal some limitations to dealing with networks with more than a few thousand nodes and edges. Computational memory and power are

the main constraints to perform the visualization of large networks. Even if the software is capable of representing a network of millions of nodes on the screen, the user may struggle to gather some valuable information from the visual outcome. In this work, we propose processing the data as a stream of networked data with Landmark, Sliding Windows, and *Top-K* algorithm applications to enhance the network visualization enabling knowledge acquisition from the output. The main goal is to sample the data stream by highlighting the *Top-K* nodes, providing a clear insight about the most active nodes in the network. We also present a case study of our methods applied to Telecommunication network data with several millions of nodes and edges. Results were obtained with a common commodity machine. In the following section, we present an overview of previous work on the subject of social network visualization and summarization, focusing on the *Top-K* algorithms.

4.5.2 Related Work

Analysis and visualization of Streaming Social Networks The main goal of the techniques of graph visualization is to optimize the node's geographic arrangement on the screen. Thus, for example, with the "force directed" and "spring-embedder" techniques the nodes that have stronger connections appear closer to each other in the visualization. Nonetheless, according to Chen and Morris (2003), although these techniques are visually interesting in some situations, the great unsolved problem is the scalability of these solutions.

Additionally, in Chen and Morris (2003), the authors emphasize that the information retrieval and visualization structural patterns can be jeopardized by a greater number of connections. In many situations, the vast number of connections between nodes makes it difficult to inspect the connections that are significant in the network. Nonetheless, the amount of connections can be reduced by eliminating them from the network.

The basic version for cutting out non-significant edges in a graph is to eliminate every connection that is represented by a low value of importance. These edges are removed if this value is inferior to an established threshold. The great advantage of this solution is its simplicity. Nonetheless, the great disadvantage is that this removal might not preserve the structural features of the networks. Thus, the resulting network might not be representative of the original network.

Another strategy is based on the topology of the networks, for example, "Minimum Spanning Tree" (MST) or "Pathfinder". MST is a tree that connects the nodes by weighting the nodes and selecting the tree with higher value nodes. MST is a particular case of Pathfinder, because Pathfinder is the group of all MSTs possible to achieve from a network (Schvaneveldt et al., 1988; Schvaneveldt, 1990).

Chen and Morris (2003), stress that MST has less computational complexity, but Pathfinder is superior when capturing topological patterns of the dynamic networks. Thus, it better captures the dynamic properties of the evolving networks.

According to Feldman and Sanger (2006) the grouping can deliver more precise searches. Therefore, they emphasize that the goal is to augment the intuitiveness of human visualization of results of the text/document processing.

For the improvement of visualization of similarity networks, a development of visualization of large-scale networks methods for streaming is required. Probably, the use of sliding windows or forgetting factors to obtain scalability in the visual output for large data size is required. These methods were introduced in some previous publications (Sarmento, 2013; Sarmento et al., 2015b, 2014a). Additionally, we will need the development and application of visualization methods of communities or affinity groups for streaming data. The analysis of these communities or groups, as well as the possibility of obtaining information or knowledge from the visualization, will be a challenge. This is largely due to the fact they are constantly changing. We will need to explore methods that use information buffers that allow unfolding of communities or groups detected in the data and that enable the extraction of a social network from the data.

4.5.3 Window-Based Visualization

Resorting to time window models is an useful strategy to limit the amount of data available for analysis, since it is based on setting a fixed point in time (the so-called *landmark*) from which the data starts being observed (Sarmento et al., 2014b). A disadvantage of this method is that the amount of data inside the window quickly grows to a prohibitive size. Other way of limiting data is by using a fixed sliding window model. These windows are bounded by the number of data points or the number of time units, being both constant.

Introduction

Visualization of large networks is known to be a hard problem to solve with typical hardware or software. Networks with more than a few thousands nodes and edges can not be computed in limited standard systems. The software and the user himself are the main constraints in visualization tasks of large networks like explained in Rafiei (2005). Even if the software is capable of outputting a network of millions of nodes on the screen it is a very hard task for the user to grasp valuable information from the visual outcome or from its analysis. In this paper we discuss and propose a new way of outputting the data as a network data stream to help the observer visualize the network and enable knowledge acquisition from the output. We present a summarization method for large scale online network streaming focusing on any specific node. Moreover, we assemble existing and novel algorithms for visualization or analysis to get a more efficient method. The results were obtained by simulation of data streaming, originated from databases. All experiences were executed with an ordinary commodity machine.

Related Work

Several studies have already addressed Ego-Networks. This field covers and relates with many varied subjects, from biology to sociological and criminal networks. The section introduces related work with a more generic approach including an overview of research on social networks.

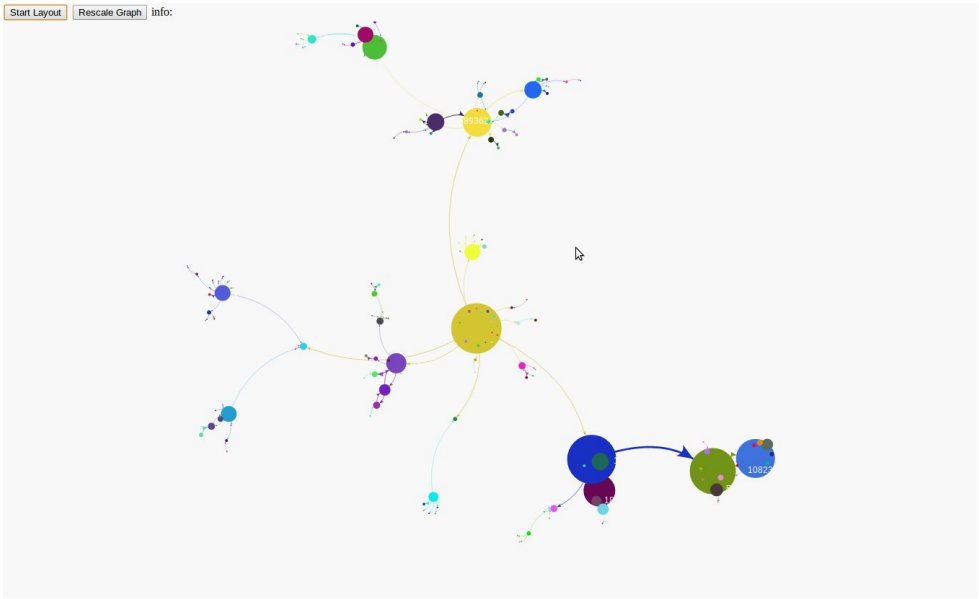


Figure 4.11. Visualization of an Ego-Network

Ego-Networks In Hanneman and Riddle (2005a), a throughout exposition about Social Networks is made and a full chapter is dedicated to Ego-Networks. Hanneman et al. define "Ego" as an individual "focal" node in a network. "Neighborhood" defines the boundaries of ego networks and includes all the direct connections and egos that tie with an ego. DeJordy and Halgin (2009), introduce the network perspective and the differences between socio-centric and ego-centric analysis. The ego-centric approach fits studies about phenomena or entities across different networks. The socio-centric approach is more suitable for studying interaction within a defined network. Wasserman et al. provide a complete study about social networks with several models in Wasserman and Faust (1994a). Some important studies address the social structure of competition. For Burt (1992), social structure of competition addresses the consequences of voids in relational and resource networks. Competitive behavior can be understood in terms of player access to "holes" in the social structure of the competitive arena. Those "structural holes" create entrepreneurial opportunities for information access, timing, referrals and control. Ego-Networks analysis provides an answer to this sensible information or properties that are are also studied in the case study section of this document.

Streaming Algorithm In this section we describe the algorithm used for streaming. For the Ego-Networks streaming representation of the data we used the landmark window implementation (Gama, 2010), focusing on the ego node's 1^{st} and 2^{nd} order connections. This algorithm represents an alternative to full networks SNA model. Figure 4.11 shows the example Ego-Network for the network node in light green colour at the centre of the picture, its direct connections and its 2^{nd} order connections.

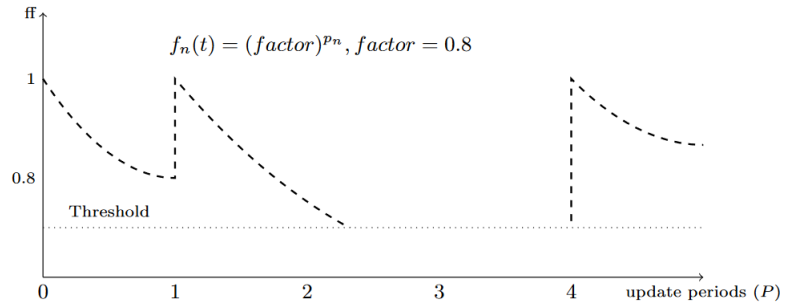


Figure 4.12. Evolving inclusion or exclusion of a given node based on the forgetting factor function. P_n is the number of update periods considering that node n did not receive any connection

The developed application enables the visualization of the events centred on some specific network node, instead of the evolution in time of the full network events. The input is a start date and hour, as well as the end date and hour in the landmark window. The algorithm returns the visualization of the evolving Ego-Network over time. New connections arising from the central node or their 1st order connections are plotted in the screen.

Ego-Networks with Forgetting Factor Ego-Networks with node forgetting factor algorithm, using streaming simulation, implies the update of a node structure with a *forgetting factor* variable per node. This variable value is the same for each new node represented on the graph. After some estimated streaming time period, forgetting factor values are updated for all nodes currently present in the structure. Those nodes with lower than the threshold forgetting factor are deleted from the graph along with their direct outgoing connections. The following expression presents the forgetting factor update that was used

$$f_n(t) = (factor)^{p_n}$$

where $factor$ is the selected initial forgetting factor, with $0 < factor < 1$, and p_n is the number of update periods for node n. p_n will be set to zero every time node n establishes a new outgoing connection. Thus, nodes that have recent related events are kept using the data timestamps.

Figure 4.12 describes the behaviour of the forgetting factor function for a specific node. In this example, the represented node establishes connections for the update periods 0, 1 and 4, meaning that for these update periods the number of periods without being updated will be zero ($P_n = 0$), and therefore $f_n(t) = (0.8)^0 = 1$. For the other update periods, ex.: 3, the value of the function will be $f_n(t) = (0.8)^2 = 0.64$, with $P_n = 2$. This means that at the update period 3, the node did not established a connection for two update periods. The figure also shows the threshold value for considering, or not, a particular node. When the function crosses bellow the threshold for a particular update period, the node is removed from the graph. In the example, right after the update period 2, the node will be removed from the network and will only be considered again in update period 4, when the node establishes a new connection.

Landmark Windows Algorithm 10 in Sarmento et al. (2014b), regarding streaming landmark window models, provides the representation of all the events (e.g., edge and node addition or removal) that occur in the network, starting at a specific time stamp, for example, 01h48m09s of 1st of January 2012.

Algorithm 10 Algorithm based on a landmark window model

Input: *start*, *wsize*, *tinc* ▷ start timestamp, window size and time increment
Output: *edges*
1: $R \leftarrow \{\}$ ▷ data rows
2: $E \leftarrow \{\}$ ▷ edges currently in the graph
3: $R \leftarrow$ getRowsFromDB (*start*)
4: *new_time* $\leftarrow$ *start*
5: **while** ($R <> 0$) **do**
6: **for all** *edge* $\in R$ **do**
7: ADDEDGETOGRAPH(*edge*)
8: $E \leftarrow E \bigcup \{edge\}$
9: **end for**
10: *new_time* $\leftarrow$ *new_time* + *tinc*
11: $R \leftarrow$ getRowsFromDB (*new_time*)
12: **end while**
13: *edges* $\leftarrow E$

This type of window model is not very useful in a streaming scenario, because it implies a growing number of events outputted on the screen and the comprehensibility lowers as this number reaches, or exceeds, a few thousands of events. This landmark application is however useful in other contexts, for example, if the network is relatively small and the analyst is interested in checking all events in the evolution of the network. Nevertheless, if the analyst wants to follow the evolution of a large streaming network, the application described in the next subsection is more appropriate.

Sliding Windows For this large data stream, we generate a dynamic sample representation of the data by using a sliding window model. In Algorithm 11, from Sarmento et al. (2014b), this sliding window is defined as a data structure with fixed number of registered events. Each event is a phone call between pairs of subscribers. Since these events are annotated with time stamps, the time period between the first call and the last call in the window is easily computed. The input parameters of this algorithm are (i) starting date and time, and the (ii) maximum number of events/calls the sliding window can have. The SNA model used in this application is a full weighted directed network, since all the nodes and edges in the network are outputted for a particular instance of the sliding window (Hanneman and Riddle, 2005b).

An example of the obtained results is provided in Sarmento et al. (2014b), in Figure 4.13. Nodes are sized according to their weighted degree. Thus, larger nodes correspond to more active subscribers, i.e., subscribers associated with a higher number of phone calls (either received or made). This is the representation of a window with 1000 events/calls, for a time period starting

Algorithm 11 Algorithm based on a sliding window model

Input: *start*, *wsize*, *tinc* ▷ start timestamp, window size and time increment
Output: *edges*

```
 1: R ← {}                                              ▷ data rows
 2: E ← {}                                              ▷ edges currently in the graph
 3: V ← {}                                              ▷ buffer to manage removal of old edges
 4: R ← getRowsFromDB (start)
 5: new_time ← start
 6: p ← {}
 7: while (R <> 0) do
 8:     for all edge ∈ R do
 9:         ADDEDGETOGRAPH(edge)
10:         E ← E ⋃ {edge}
11:         k ← 1 + (p mod wsize)
12:         old_edge ← V[k]
13:         REMOVEEDGEFROMGRAPH(old_edge)
14:         E ← E \ {old_edge}
15:         V[k] ← edge
16:         p ← p + 1
17:     end for
18:     new_time ← new_time + tinc
19:     R ← getRowsFromDB (new_time)
20: end while
21: edges ← E
```

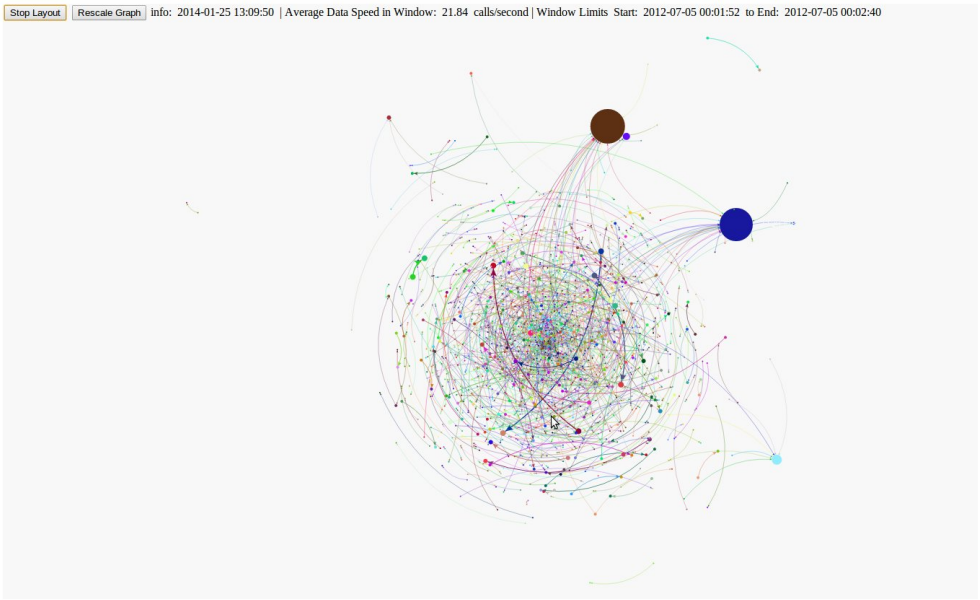


Figure 4.13. Visualization of the call graph using a sliding window approach

at 00h01m52s and ending at 00h02m40s. The evolution of the network is visually and immediately conclusive. There are three nodes with the largest number of connections/phone calls in the network.

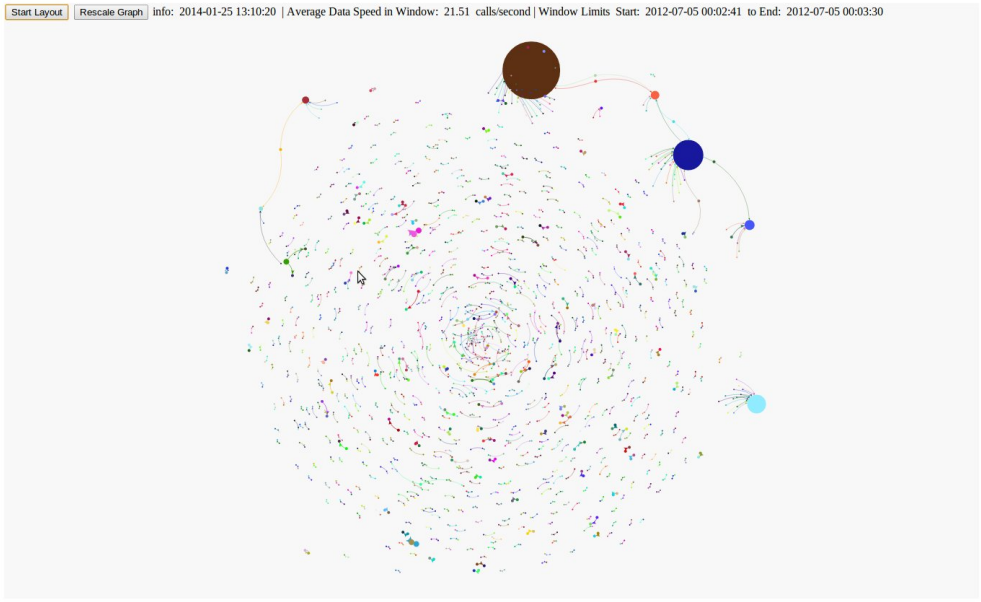


Figure 4.14. Visualization of the call graph using a sliding window approach (2nd version)

From the figure, we can also see the connection established between two of these three largest nodes. Figure 4.13 also displays the average data speed in the window (approximately 22 calls per second). This average data speed is computed by counting the number events (i.e., phone calls) inside the window, which comprises all the events observed during the time period associated with the window length. Under different experimental conditions, namely when analysing the window obtained for mid-day, the data speed increases, with more phone calls per second. After several experiments with different window sizes, and considering that the data speed changes, we concluded that this speed should not be smaller than approximately 100 events and also not larger than approximately 1000 events. With the minimum data speed conditions, 100 events represents a window period of around 10 seconds of events. With the maximum data speed and a window of 1000 events, it represents around 5 seconds of data. Using this data, less than 100 events represents changes in the window that are too fast to be visually comprehensible, and more than 1000 events represents too many events, reducing the visual comprehensibility of the output.

In Sarmento et al. (2014b), Figure 4.14 represents the next window instance, starting at 00h02m41s and ending at 00h03m30s. Considering the previous Figure 4.13, we can visually observe the evolution of the network and observe that there is a new smaller node establishing connection to the most active nodes identified before, in this window of 1000 events.

4.5.4 Ego Networks - Visualization of Text Nodes

Case Study

The proposed forgetting factor method was tested in large-scale telecommunications Ego-Networks in this case study. The aim was to determine if the method results were representative of the original data Ego-Network as the data streaming and forgetting factor value evolved. Thus, we used our novel algorithms either with or without forgetting factors. We discarded any events related to voicemail numbers for these tests, which biased these studies.

Description of the Data Call Detail Records (CDR) log files were retrieved from equipment distributed in different geographic locations. The network data has an average of 10 million calls (edges in the social network) per day. The phone numbers were changed to different identifiers to preserve users' anonymity. A call between A and B phones is represented as an edge in the social network. Because some individuals receive and make more than one call, the full network has an average of 6 million unique users/nodes per day. The dataset contains anonymous data for 135 days. For each edge/call, timestamp information shows the date and hour of the beginning of the call. The number of calls per second varies from around ten at midnight and peaks at mid-day with 280.

Results

Ego-Networks with Forgetting Factor Figure 4.15 illustrates the test of parameters with the same anonymous number's Ego-Network presented in Figure 4.11. The *update time period* was set to 3 hours and the *forgetting factor threshold* set to 0.6. The *initial forgetting factor* value was 0.95.

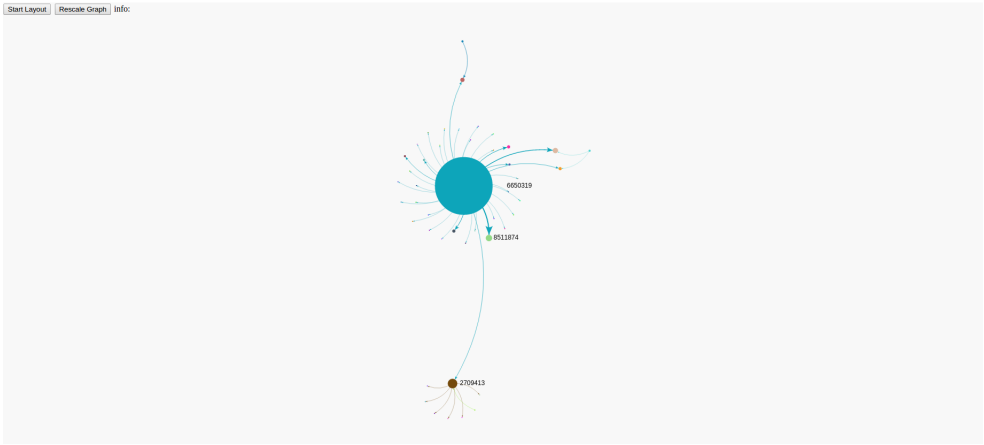


Figure 4.15. Ego-Network with forgetting factor

For the same period of streaming, Figure 4.15 presents much fewer connections in the graph, meaning that some of the connections of the central node were deleted as an outcome of a forgetting factor lower than the threshold value of 0.6.

Node and Edges counter variation with time After the previous experiment, we randomly selected three phone/nodes in the set of approximately 10000000. This was made for one week period of the available data. We run the experiments with five different update periods, three different thresholds, and three different forgetting factor values with these nodes. The used values for the experiments were 3, 6, 12, 24, and 48 hours for the *update time period*. The selected values for the *initial forgetting factor* were 0.65, 0.8 and 0.95. The *forgetting factor threshold* values were 0.1, 0.3 and 0.6.

We then studied the variation of the counters for nodes and edges. The variation regarded time and comparison between Ego-Network with and without forgetting factor for the same update periods. It only considered 1st order outgoing connections for the experiments. Thus, the focus is more on active node events and less on passive ones.

The following figures represent the variation when the *forgetting factor* is 0.95 and the *forgetting factor threshold* is 0.6 and by changing only the *update time period* parameter.

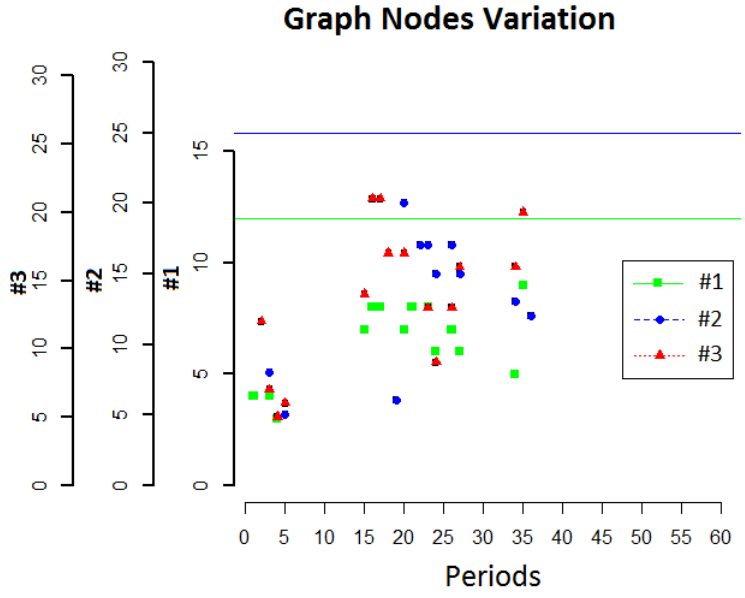


Figure 4.16. Ego-Network nodes counter

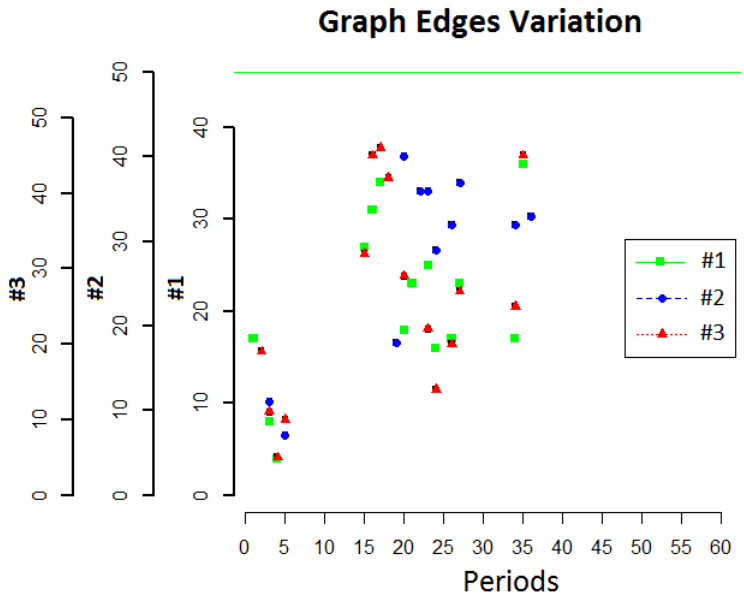


Figure 4.17. Ego-Network edges counter

It is known that the values of both counters increase with time for the original Ego-Network without forgetting factor. In Figure 4.16 and Figure 4.17, the horizontal lines represent each node network's maximum number of nodes and edges in graph. With the forgetting factor version, in Figure 4.16 and 4.17 with an update period of 21600 seconds, there is evidence, for the majority of the three selected numbers, the same counters are lower and with the tendency for stability. Moreover, the maximum number of nodes and edges is much lower than the original network maximum values for both counters. Thus, some horizontal lines are off the scale and therefore not visible in these figures. It proves this streaming method implies memory-saving characteristics by discarding older events in the streaming.

In this short section, we propose a new type of application for large-scale telecommunications networks visualization, streaming, and analysis. With data time stamps, we approach the data with a streaming point of view to visualize samples of data. Thus, it is comprehensible to the user and enables knowledge extraction from the visual output.

Variations of the Landmark Window streaming algorithm were developed to output Ego-centered networks,. Ultimately, we propose a novel streaming method for this social network analysis by using the forgetting factor in the Ego-Network output. The weight and importance of recent events are higher for the majority of the test cases. We consider and prove this to be an effective visualization method for Ego-Networks SNA.

Finally, we believe this method for evolving network visualization and analysis is a light method to visualize massive Ego-Networks. Thus, an ordinary commodity machine can achieve the simulation of a data stream and the visualization results very close to the node-link level.

4.5.5 Summary and Contributions

In Chapter 3, we stressed that one of the most problematic features of the Affinity Miner prototype was the visualization of the network of documents. Unless this issue is attended to, the graph is cluttered with too much information, even for a modest number of documents of the order of hundreds. We have described various methods in this section that can provide better ways for scalability of the representation of documents and their analysis. These include folding and unfolding of groups of documents, the use of Ego-Networks, to visualization of central nodes among others. We have thus contributed to the mitigation of this important problem. We have published various articles on the topic discussed in this section: Sarmento et al., 2014a; Sarmento et al., 2014b; Sarmento et al., 2015b; Sarmento et al., 2016a, Sarmento, 2017.

Chapter 5

Incremental Similarity In Text-Streams

5.1 Introduction

Considering we want to organize documents, one main task is the grouping of those documents in similar groups. This important task allows documents to be searched with more efficiency and with a more organized structure that allows the reader to discover similar documents, and that might be of interest. Similarity measure of documents is the key measure we want to calculate at this point of the procedure to find information in document sets.

Problem

The Similarity matrix elements grows quadratically with the number of authors. This is a high dimensional and also a computationally complex problem to solve.

What should we do? How to work with TF-IDF large matrices?

- Should we work with Sparse Matrices techniques?
- Should we use feature selection or topic modeling to lower the number of columns (tokens) in the matrix?
- Should we use clusterization of authors to reduce the number of instances?
- Compact those matrices? (reduce the dimension of the matrix)

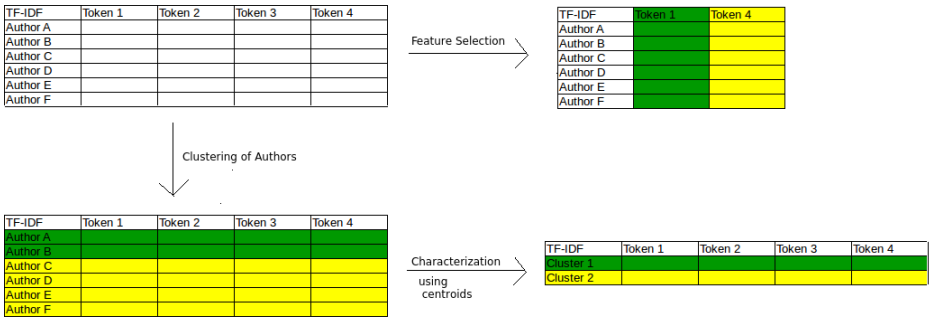


Figure 5.1. Compacting document-term matrix

Figure 5.1 represents the need to represent large document-term matrices in compact ways. This could be achieved by performing either feature selection, i.e., word selection and/or grouping of documents either by clustering or community detection. The representation of these groups could lead to an estimation of central documents or points of data in the calculated groups or clusters.

5.2 Incremental Similarity Measures

It is necessary to test methods and solutions for data stream pre-processing regarding text pre-processing. Additionally, the development and application of summarization methods to efficiently treat large-scale data is important. Thus, by obtaining a feasible method, it is possible to carry out the necessary text similarity analysis in a streaming mode. Moreover, this method allows the use of computing resources efficiently, regarding the processing and the use of memory. An example is the use of a method similar to indicated in Carmona Cejudo (2013) which describes the concept of STFSIDF. Obtaining the STFSIDF is an approach for obtaining a modified version of the TF-IDF matrix through the use of *sketches* and *Bloom filters* as tools for increased efficiency in the calculations. This allows to maintain dynamic attributes spaces with limited resources and reduces processing time and memory usage. Thus, this method or a similar one could help resolve the scalability problem presented by the large streaming data.

We need to incorporate clustering methods capable of accepting data streams to identify affinity groups. These groups represent a summary of all the elements that would otherwise be individually visualized in the form of social networking affinities. Thus, we can compact the obtained visual result and increase the user's understanding of what is displayed. A research of methods described in Aggarwal et al. (2004); Aggarwal and Yu (2006) on the analysis of text data streams and clustering may lead to adjustments to the problem we wish to solve. In Aggarwal et al. (2004); Aggarwal and Yu (2006) the authors created the HPStream that is appropriate for use with large data streaming. As the number of dimensions increases, the major problem in large data clusterization is that the distance calculation to evaluate the similarity between groups/clusters is increasingly less accurate. This is because the distance between different points tends to be the same. The authors propose and exploit the selection of attributes and subspace groups to solve the problem. It can, therefore, be a good starting point for the resolution of the need for scalability of the data. It will be necessary to reduce the need for attributes or groups with a high number of dimensions.

Due to the fast-expanding of high throughput text data sources, like social networks or news gateways, it is necessary to test methods and solutions for data stream pre-processing. Additionally, we need the development and application of summarizing methods that achieve efficient treatment of large-scale data.

In previous Chapter 3 about the prototype, Figure 3.5 mentioned the folding of the full network in existing affinity groups. The network is shown in a higher hierarchical snapshot of the full network on the right side. The size of nodes is, in this context, proportional to the number of elements the affinity group has, and the mean of the similarity weights gives the width of the connection/edges for each inter-group element connection.

For this section, we used several R packages. To compute graph-based operations, we used the "igraph" package (Csardi and Nepusz, 2006), and "fastmatch" package (Urbanek, 2017), providing required fast index search for repeated look-ups, extremely fast on any subsequent look-up as it keeps the hashtable in memory. Text mining and pre-processing were done with the "tm" package (Feinerer and Hornik, 2018; Feinerer et al., 2008). “Rcpp” package was used to provide libraries

for packages built with C++ (Eddelbuettel and François, 2011; Eddelbuettel, 2013; Eddelbuettel and Balamuta, 2017). To take measurements of processing and memory use, we used package "microbenchmark" (Mersmann, 2018).

5.2.1 IS-TFIDF for Text Streams

One of the major problems we have is the scalability of the similarity matrix of $n * n$ authors regarding its graph representation.

Additionally, TF-IDF matrix also grows to become a high-dimensional problem to solve and compute. Regarding this important issue with coping with large document sets and corresponding text flux, we researched efficient ways to analyze text data.

We experimented with several concepts regarding text streams analysis. We tested the implementation of Incremental Sparse TF-IDF (IS-TFIDF) and Incremental Cosine Similarity (ICS) using bipartite graphs.

We use bipartite graphs - one type of node are documents, and the other type of nodes are words - to know what documents are affected with a word arrival at the stream (the neighbors of the word in the graph). Thus, we leverage optimized algorithms used for graph-based applications with this information. The concept is similar to, for example, the use of hash tables or other computer science concepts used for fast access to information in memory.

In the following sections of this chapter, we explain the method and our settings for the tests for two versions of IS-TFIDF, an algorithm concept based on text and documents, through the use of evolving networks.

5.2.2 IS-TFIDF and ICS Method

IS-TFIDF can be seen as a method with an updatable list structure of documents (LIL - List of lists type of Sparse matrix storing). Each entry per document has a vector of words that the document has after typical pre-processing (removal of stop words, numbers, etc.). This vector contains the TF-IDF values for each word. These values are updated in each stream iteration. In each new iteration or input of a new chunk of text documents, we update a bipartite graph by adding new words and documents and establishing the connection (edges) between words and documents. Thus, this graph has two types of nodes, words, and documents. New words that appear in the stream are connected to their documents.

Regarding our goal, the efficient update of the similarity between documents (ICS), we use the bipartite graph first order neighbors for new or updated words in the stream to check which pairs of documents' changes similarity. Then, we recalculate these pairs' similarities due to changes. This way, we avoid the recalculation of the similarity of all pairs of documents in the stream in each iteration of the stream.

Figure 5.2 shows an example of a bipartite graph used by our method. This example has two snapshots or iterations of a stream. Consider the following example:

A sample of two documents enters the stream in a larger network of documents and words.

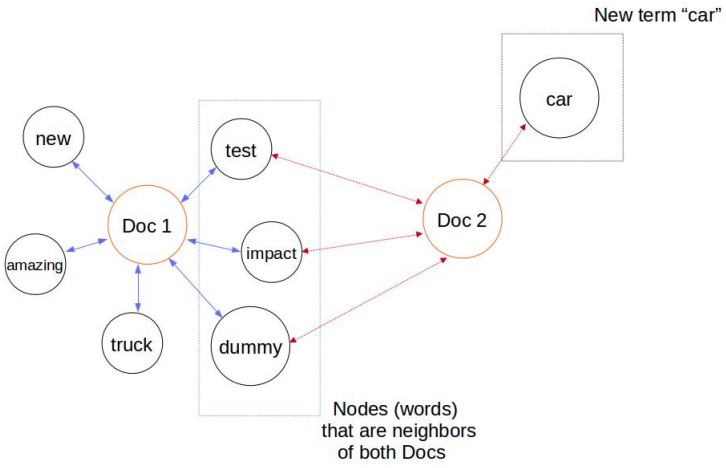


Figure 5.2. Bipartite Graph Method

- First Snapshot of text (Doc 1): “New Amazing Truck Impact Test Dummy”;
- Second Snapshot of text (Doc 2): “Car Impact Test Dummy”.

The word "Car" is a new term added to the graph with Doc 2. Additionally, the words that are simultaneously common to both documents also change the TF-IDF values. If Doc 2 only had the word "Car" we did not need to update the similarity between Doc 1 and Doc 2 since they were not connected (they were not both neighbors of the term "Car"). Instead, as we have the neighbor words "Impact", "Test", and "Dummy" changing their TF-IDF values in the second snapshot, we have to recalculate the similarity between Doc 1 and Doc 2 in this iteration.

One Document Streaming (ODS)

IS-TFIDF ODS is based on the concept of using a sliding window to iterate through the stream of data, where each snapshot is considered a new document. Thus, this way, the input is not incremental because any new chunk of text is not considered an update or increment in an already processed document but a new one. This is the most basic architecture of our solution, with the advantage of only recalculating TF-IDF and similarity for the stored graph-affected documents, only regarding those with common words in their content. On the occasion of a new document in the stream with new words only, calculations are avoided, and efficiency increases.

Several Documents Streaming (SDS)

IS-TFIDF SDS is based on the concept of using a sliding window to iterate through the stream of data, where each snapshot is considered a flow of several documents. Consider, as an example, that

in the first snapshot, we had the text for doc1, doc2, and doc3. Thus, if in the third snapshot we had more information added to doc1, we would incrementally update and proceed with the addition of information to doc1. This can happen, for example, in a real-world dataset like described in this section for the temporal-based dataset of scientific publications of different scientists. SDS was used to calculate similarity changes between scientists through evolving time. Relatable modeling for a non-streaming scenario is known as UserAsDocument (Cossu et al., 2016): the proximity between two entities, each of which owns a collection of documents, can be assessed by the textual similarity between the two super-documents resulting from concatenating each of their collections.

Algorithmic Analysis

Algorithm 12 is a conceptual description of the procedure each document of an input stream would be subject to when IS-TFIDF is employed. This same procedure covers both ODS and SDS despite differences during its execution: for ODS, each input document is unprecedented, so that the condition in algorithm 12 is always true; That is not the case for SDS, as two different input items may belong to the same document, which is perceived incrementally through streaming excerpts of it.

Algorithm 12 IS-TFIDF document Processing Algorithm

Require: doc, the document to be processed
Require: g, a bipartite graph relating documents to text tokens
if there is no node regarding doc in g **then**
    Add node doc to g
end if
Preprocess doc
Let S be an empty set
for each token $t \in doc$ **do**
    if there is no node regarding t in g **then**
        Add node t to g
    end if
    if there is no edge in g between nodes doc and t **then**
        Add an edge between doc and t to g
    end if
    Calculate the frequency of t regarding doc
    Calculate IDF of t considering each document seen
    for each document u neighboring t in g **do**
        Calculate TF-IDF of t regarding u
        Add u to S
    end for
end for
for each pair of distinct documents $\{u, v\} \subseteq S$ **do**
    Calculate the similarity between u and v
end for

In order to assess the asymptotic computational complexity of the algorithm, the following variables should be considered: let k be the expected number of tokens in a document after pre-processing; let m be the expected number of documents that have a given token in common; let n be the expected number of documents that have any token in common with a given document;

let p be the number of documents seen up to a given time instant; let q be the number of tokens seen up to a given time instant. The loops of algorithm 12 would take k, m, and $\binom{n}{2}$ iterations, respectively. Besides these, the operations in all other lines have a negligible cost, except for algorithm 12, which would require $O(q)$ steps for cosine similarity, as an example. Therefore, the proposed algorithm has a time complexity of $O(km+\binom{n}{2}q) = O(n^2q)$. This can be compared to the naive approach, which would have a substantially greater time complexity of $O(k+pq+\binom{p}{2}q) = O(p^2q)$, as the similarity between each pair of documents would be reevaluated every time a stream item is processed. At last, the space complexity of both alternatives is $O(nq+n^2)$, being dominated by the TF-IDF matrix and the pairwise similarity matrix of all documents.

5.2.3 Case Study and Evaluation

This section presents the method to analyze both ODS and SDS implementations of IS-TFIDF regarding its comparison with the original batch algorithm. First, we describe the data used for the tests. Then, we describe the methodology to perform the comparison tests.

Description of the Data

Reuters News Dataset In this case study, for algorithm efficiency measurements, we selected a dataset publicly available for research purposes by Corney et al. (2016). This dataset has a high amount of information, including Reuters news and several articles extracted from Blogs. The data contains article titles, content, type of publication (which might include news or Blog posts), the source institution, and also date and time of publishing. The high quality of this structured data and high organization make it a good source for text mining or NLP tasks. We selected all Reuters news, from the 1st to the 30th of September 2015. This corresponds to 400 news articles. Regarding the used text, we could choose between the news Titles or the news content to analyze. We choose the news content in all our studies.

Thus, we used this particular dataset to test the ODS method in our experimental setup. Each document presented to the input in the flow of the stream of documents was considered a new document.

INESC-TEC Scientific Publications Dataset This dataset consists of publications by scientific researchers associated with the INESC-TEC scientific research institution. This dataset has more than 2500 text documents. This dataset is temporal regarding keywords, titles, abstracts, authors' names, and co-authors of several publications, either published in international conference proceedings or Journals of scientific content.

Thus, we used this particular dataset to test the SDS method in our experimental setup. Each document presented to the input in the flow of the stream of documents was considered a document that should be checked for its existence in the already stored documents, to have a potential need for an incremental update, in the occurrence of a new publication of a particular author already registered in the previous iterations of the algorithm.

Experiments

The ODS IS-TFIDF and ICS implementation, the SDS IS-TFIDF, and ICS implementation were evaluated in an incremental setup for both the developed versions. Additionally, both algorithms were implemented and evaluated regarding Processing Efficiency and compared with the batch version.

Processing Efficiency

Results for the ODS IS-TFIDF and ODS ICS were obtained from several increments of text news, compared with the original batch algorithm, using the algorithms in the tm package of R (Feinerer and Hornik, 2018; Feinerer et al., 2008). Both versions were evaluated regarding efficiency in an incremental setting configuration. The original batch results served as a baseline. In this setup, six snapshots of news were considered. In total, the first 20 days of September 2015, corresponding to 300 news articles, were passed as input. The snapshots were built by aggregating publications daily by using the timestamps available in the dataset. In the Batch algorithm, the input text has all daily news from day 1 to the current day for every snapshot. For the incremental algorithm (ODS IS-TFIDF and ICS), the first snapshot receives only 15 days of news articles as input. In the following snapshots, the algorithm only receives the set of text publications added to the corpus, in that particular day snapshot (incremental).

SDS IS-TFIDF and SDS ICS were obtained from several increments of data concerning researchers' publications. Each increment had several author's publication titles, with five research titles per author. In the end, 22 snapshots of data were sent as input.

The empirical evaluation performed consisted mainly of comparing the run times of each increment (duration of each increment and cumulative execution time) for all versions of TF-IDF and ICS. Note that the batch algorithm will always need to process all the accumulated text. In the end, an analysis of the total speed-up ratio obtained in each of the steps is added.

Results

We present the results for all versions of the algorithm in this section. All algorithms are compared with the Batch version of TF-IDF, with subsequent cosine similarity calculation. The comparisons regard the processing efficiency. The charts show us the total number of nodes and edges of our bipartite graph in each iteration and the metrics used in this comparison. Those metrics include elapsed time for each iteration, cumulative time on each iteration, and speed-up ratio, i.e., the speed-up presented by the IS-TFIDF when compared with the baseline TF-IDF algorithm and similarity calculations.

Processing Efficiency

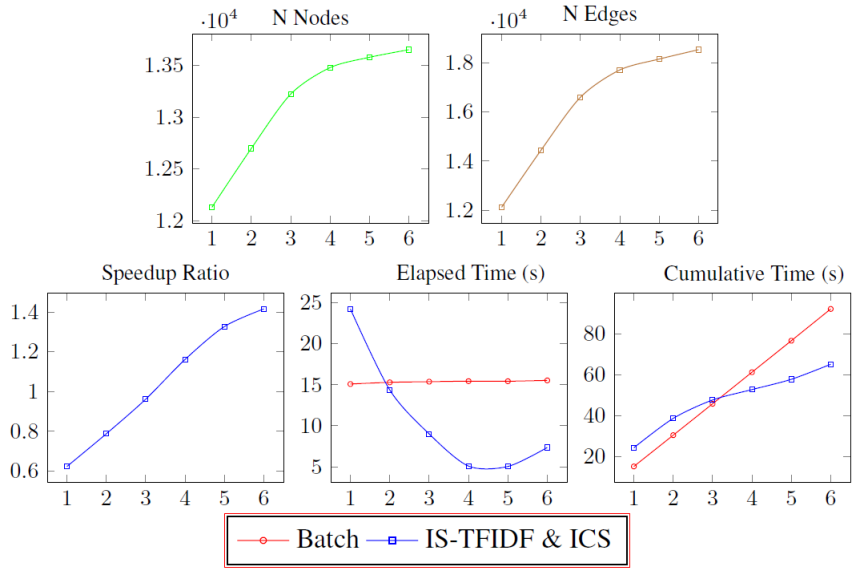


Figure 5.3. Efficiency results with the Reuters News Dataset (for all algorithms)

Results on Reuters News dataset (Figure 5.3)

(Speed-up Ratio) shows that, although the IS-TFIDF implementation is slower in the first iterations, it starts being faster after some iterations.

(Elapsed Time) shows that the Batch algorithm has a slight increase in elapsed time. Regarding our implementation, the update time decreases as the stream evolves.

(Cumulative Time) shows a clear advantage of using IS-TFIDF & ICS, as the cumulative time rises at a slower pace when compared with the batch version.

Results on INESC TEC research dataset (Figure 5.4)

(Speed-up Ratio) shows that, although the IS-TFIDF implementation is slower in the first iterations, it starts being faster after some iterations.

(Elapsed Time) shows that the Batch algorithm has a pronounced exponential rise with the evolving stream. Otherwise, our implementation slightly increases elapsed time, behaving more linearly as the stream evolves.

(Cumulative Time) shows a clear advantage of using IS-TFIDF & ICS, as the cumulative time rises at a slower pace when compared with the batch version. This is expectedly more pronounced as the stream evolves with time and more data arrives at the system.

5.2.4 Summary and Contributions

This section deals with initial parts of the process of organizing of text documents. Similarity between pairs of documents plays an important role in many methods that deal with this issue. Our contribution to this task is in the design of a method that calculates this similarity in a streaming

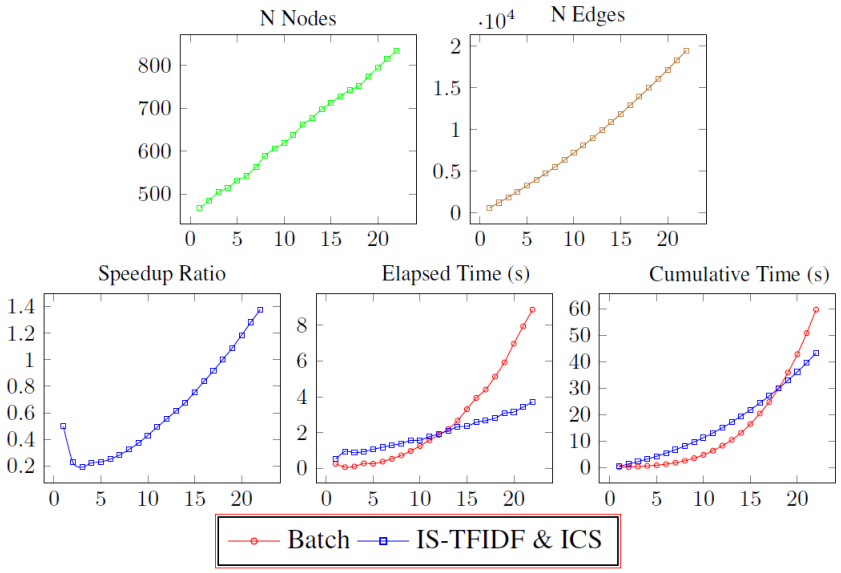


Figure 5.4. Efficiency results with the INESC TEC research Dataset (for all algorithms)

setting, that is, when we have a flux of text documents coming in. With the use a mapping of text documents and respective words in bipartite networks. Then we calculate evolving measures of similarity between the documents in a stream, with or without discarding previous information about the documents. We conclude that we can achieve more efficient algorithms (according to e.g., the speed-up ratio) to calculate a measure that was already considered of high complexity in batch setting. We have published various articles on the topic discussed in this section and the most focused on this issue is: Sarmento, et al., 2021.

5.3 Keyword Extraction

The organization of documentation is expected to serve a goal for better and higher levels of information retrieval from text. After introducing the procedures to better organize documentation in Chapters 5, 4.2, 4.3 and 4.5, we now deal with the issues associated with the identification of important keywords that define the content and context of the documentation we wish to analyze and search in. In the context of the developed prototype, explained in Chapter 3, keyword selection is intimately related to feature selection procedures, might even be considered similar. Since we are dealing with features of text documents, i.e., words, we can do several paralelisms with keyword selection and feature selection.

5.3.1 Contextualization and Clustering

Similarity and Hierarchical Clustering

By using the Stats R Package by the (R Core Team, 2020), we performed hierarchical clustering of the similarity matrices retrieved from previously explained algorithms. Hierarchical clustering is a type of clustering algorithm that performs clustering based on proximity of cluster points, with several different methods, i.e., different optimization measures for cluster forming.

Hierarchical clustering is a well-known method for clustering data, as explained in, for example, Gordon (1999). It was updated and transformed numerous times in the past, and it is described in its various forms in publications like, for example, in Murtagh (1985); Legendre and Legendre (2012); Murtagh and Legendre (2014).

Regarding Clustering with Contextualization - Word2Vec

Word2Vec model is the representation for words in a vectorized distribution of values. The general characteristic of these representations consists in that semantically similar words have proximate representations in the target vector space.

Word2Vec model comes from the idea of predicting neighbors of a word using the theory of neural networks. The main attribute of distributed representations was proposed initially by Rumelhart et al. (1986). The author proposed that the representations of semantically similar words are close in the vector space. The vector representations of words are learned using the distributed Skip-gram or Continuous Bag-of-Words (CBOW), by Mikolov et al. (2013). The CBOW idea predicts the word "in the middle" from the surrounding words. In contrast, in the Skipgram model, the training objective is to learn vector representations that are good at predicting their context in the same sentence.

Many previous works used bag-of-words in conjunction with Word2Vec to inspect the similarity between text chunks or documents.

Regarding Embedding with FastText

After keyword extraction from documents, this time with the use of previously developed IS-TFIDF or ITR, we needed to find similarities between documents to be able to build a similarity graph.

To do this, we could use the mentioned IS-TFIDF for Incremental Similarity and ITR to retrieve a list of keywords per document in the flux. We could try to contextualize the similarity between documents with external use of word embedding and the Word2Vec model.

FastText (Bojanowski et al., 2016) was used, in a context of word representation, to reach for a compound similarity of two documents, and with both calculated with the extracted significant keywords at any particular moment in the stream of documents.

5.3.2 Case Study and Evaluation

This section presents the method to analyze ODS implementations of IS-TFIDF or ITR regarding both comparisons. We introduce the reader to the data used for the tests. Finally, we describe the methodology to perform the comparison tests.

Description of the Data

In this case study, for the algorithms used in retrieving keywords, similarity for clustering, we selected a dataset publicly available for research purposes by Corney et al. (2016) and to obtain corresponding measurements for evaluation of quality. We also selected another dataset with ground truth labels, the Covid-19 emotions dataset by Kleinberg et al. (2021). Additionally, we start by using Kaggle (2021) for the comparison of IS-TFIDF with and without contextualization by using Word2Vec and fastText embedding.

TED Dataset TED talks dataset (Kaggle, 2021) was used to test the hypothesis that IS-TFIDF would gain with the use of Word2Vec embedding regarding clustering quality. These datasets contain information about all audio-video recordings of TED Talks uploaded to the official TED.com website until September 21, 2017. The TED main dataset contains information about all talks, including the number of views, number of comments, descriptions, speakers, and titles. The TED transcripts dataset contains the transcripts for all talks available on TED.com. We selected the complete transcriptions for analysis in the experimentation testbeds.

Reuters Dataset The first dataset we selected was from Corney et al. (2016), which has a high amount of information, including Reuters news and also several articles extracted from Blogs. The data contains article titles, content, type of publication (which might include news or Blog posts), the source institution, and also date and time of publishing. This structured data's high quality and high organization make it a good source for text mining or NLP tasks. We selected Reuters news, corresponding to circa 2000 news articles. Regarding the used text, we could choose between the news Titles or the news content to analyze. We choose the news content in all our studies.

Thus, we used this particular dataset to test the ODS method in our experimental setup. Each document presented to the input in the flow of the stream of documents was considered a new document.

Covid-19 emotions Dataset The COVID-19 emotions dataset by Kleinberg et al. (2021) is a ground truth dataset of emotional responses to COVID-19. Participants were asked to indicate their emotions and express these in written text. This resulted in the Real World Worry Dataset of 5000 text documents, 2500 short text, and 2500 long texts. We used only long texts in our studies.

Methodology

We started the tests with the hypothesis that IS-TFIDF would gain using Word2Vec to improve clustering quality. We tested this assumption with de TED dataset and for hierarchical clustering with and without using Word2Vec for 15 evaluation metrics and 30 data stream points.

Then, we proceeded with the comparison between IS-TFIDF and ITR, both with Word2Vec. The ODS IS-TFIDF and ICS implementation or the ITR implementation were then evaluated in an incremental setup for both the developed versions, with the Word2Vec, i.e., FastText embedding, being used in the majority of the time. The only time the original similarity between documents with IS-TFIDF was used - i.e., by using cosine/euclidean distance similarity - was when there was no possible calculation using Word2Vec. This can occur when words are not present in the available Word2Vec dictionaries. Therefore, we used a hybrid solution for similarity update, with contextualization given by Word2Vec, to calculate the similarity between extracted keywords from both algorithms.

To avoid disparities between algorithms, eight keywords with higher values of TF-IDF were selected per document, in the case of IS-TFIDF. Similarly, for the ITR algorithm, eight keywords were selected from the Top-K table of keywords, in the case of this second algorithm for testing and comparison purposes and per document. With ITR, in the case Word2Vec model with FastText did not return a similarity value; this was considered 0.

Both algorithms were tested with 30 stream points within a distance of 25 documents. This is equal to saying that this represents the passing of more than 1000 documents in the stream sample for similarity (continuously updated for each new document) retrieval and corresponding clustering tasks.

Resuming, in Table 5.1, with 30 stream points, we had the following number of different classes per point, and for each dataset:

Hierarchical clustering We used the "complete" method for hierarchical clustering of the obtained similarity matrices and corresponding distances between documents. Although we tested some other methods, the differences were insignificant and proved to be minor when selecting a method for this type of clustering algorithm.

Table 5.1. Number of groundtruth classes for each dataset

Data Stream Point	TED	REUTERS	COVID-19
1	16	10	5
2	26	15	5
3	30	16	6
4	42	18	6
5	52	20	7
6	53	24	7
7	57	27	7
8	61	31	8
9	68	32	8
10	70	32	8
11	71	35	8
12	74	35	8
13	78	36	8
14	81	36	8
15	83	37	8
16	84	37	8
17	85	37	8
18	86	39	8
19	89	42	8
20	92	42	8
21	95	42	8
22	98	42	8
23	99	42	8
24	99	43	8
25	108	43	8
26	110	43	8
27	115	43	8
28	118	43	8
29	120	43	8
30	120	44	8

Quality Measurements - External Validation To calculate evaluation metrics, we used the ClusterR R package by Mouselimis (2020). This package returns several metrics to evaluate and validate clusters with ground truth classes. Although there are numerous published research papers and documentation that provide results using external validation metrics, there is crescent care with selecting these metrics within the research community. There is also a crescent fear within the community that these metrics offer some bias in results that might prove them not very reliable, for example, as written by Lei et al. (2017). Thus, we choose the option to use most metrics available to make conclusions of which algorithm would be better to achieve document similarity. The list of metrics used in this study is as follows:

- purity (PUR);
- entropy (E);
- normalized mutual information (NMI);
- variation of information (VI);
- normalized variation of information (NVI);
- specificity (SPE);

- sensitivity (SENS);
- precision (P);
- recall (R);
- F-measure (F1);
- accuracy OR rand-index (RI);
- adjusted-rand-index (ARI);
- jaccard-index (JI);
- fowlkes-mallows-index (FMI);
- mirkin-metric (MM)

Results

In this section, we summarize the results by stating average values for the 15 metrics, taken with the 30 point sample of the documents stream. The higher values are underlined with bold digits for easier reading of these tables.

IS-TFIDF and IS-TFIDF with Word2Vec - TED DATASET In Table 5.2, we show the results for 15 evaluation metrics, for the comparison between having and not having Word2Vec (W2V) in IS-TFIDF algorithm, and what the implications are considering clustering evaluation metrics.

Comparing IS-TFIDF algorithm, the following table Table 5.3 shows the Mann-Whitney-Wilcoxon test values of mean differences, for the 15 metrics available, for the TED dataset experiments:

Table 5.2. Quality Measurements Comparison - TED DATASET

Measure	IS-TFIDF	IS-TFIDF (W2V)
PUR	**0.382**	0.318
E	**0.329**	0.327
NMI	**0.544**	0.495
VI	4.649	**4.987**
NVI	0.616	**0.667**
SPE	**0.882**	0.857
SENS	0.121	**0.125**
P	**0.050**	0.030
R	0.121	**0.125**
F1	**0.062**	0.048
RI	**0.857**	0.832
ARI	**0.015**	-0.006
JI	**0.032**	0.025
FMI	**0.072**	0.061
MM	**36353.667**	32856.333

As seen in Table 5.3, we have statistically significant equality of metrics for almost half of the evaluation metrics, providing some evidence that IS-TFIDF might not gain nor decrease quality

Table 5.3. IS-TFIDF and IS-TFIDF with Word2Vec - TED DATASET

Measure	Significantly different?
PUR	**Yes**
E	No
NMI	**Yes**
VI	**Yes**
NVI	**Yes**
SPE	No
SENS	No
P	**Yes**
R	No
F1	**Yes**
RI	No
ARI	**Yes**
JI	**Yes**
FMI	**Yes**
MM	No

with the use of Word2Vec. We then proceeded with a comparison using other datasets for both ITR and IS-TFIDF using contextualization and Word2Vec embedding.

Quality Measurements Comparison - Reuters Dataset Comparing both algorithms, in the Table 5.4, we show the average value of the 15 metrics available for the Reuters dataset experiments:

Table 5.4. Quality Measurements Comparison - Reuters Dataset

Measure	IS-TFIDF	ITR
PUR	**0.652**	0.478
E	**0.410**	0.140
NMI	**0.434**	0.183
VI	**3.838**	3.533
NVI	0.721	**0.896**
SPE	**0.780**	0.226
SENS	0.207	**0.782**
P	0.195	**0.210**
R	0.207	**0.782**
F1	0.196	**0.326**
RI	**0.663**	0.338
ARI	-0.012	**0.004**
JI	0.109	**0.196**
FMI	0.199	**0.401**
MM	66403.467	**128908.000**

Quality Measurements Comparison - Covid-19 Dataset Comparing both algorithms, the Table 5.5 shows the average value of the 15 metrics available for the Covid-19 emotions dataset experiments:

Comparison with Statistical Tests After checking the results for 15 metrics of evaluation, we concluded that the ITR algorithm provided higher metrics for the evaluation of clustering, with both datasets, in 10 of 15 possible metrics.

The list of 10 metrics are the following:

- purity

Table 5.5. Quality Measurements Comparison - Covid-19 Dataset

Measure	IS-TFIDF	ITR
PUR	0.603	**0.614**
E	**0.302**	0.069
NMI	**0.072**	0.054
VI	**2.613**	1.992
NVI	0.963	**0.972**
SPE	**0.269**	0.060
SENS	0.714	**0.946**
P	0.390	**0.398**
R	0.714	**0.946**
F1	0.500	**0.560**
RI	**0.445**	0.412
ARI	-0.015	**0.005**
JI	0.335	**0.389**
FMI	0.525	**0.613**
MM	105590.533	**109987.133**

- normalized variation of information
- sensitivity
- precision
- recall
- F-measure
- adjusted-rand-index
- jaccard-index
- fowlkes-mallows-index
- mirkin-metric

All these tests were performed using Mann-Whitney-Wilcoxon with paired samples, for the 30 sample stream points in the flux of documents, within 95% of confidence level. Correspondingly, IS-TFIDF presented higher results of the following metrics:

- entropy;
- normalized mutual information;
- variation of information;
- specificity;
- accuracy OR rand-index;

5.3.3 Discussion

Although results are routed to ITR as a better choice for clustering with the same amount of extracted keywords, several questions arise with this research.

One doubt is related to thresholds for the evolving networks, particularly the algorithms discarding lower values of similarity between documents. Although some ad-hoc tests were done, with IS-TFIDF and similarity threshold, to discard lower similarity between documents, the value was thought to be around 0.12 on a scale from 0 to 1. This is equal to say that similarity below or equal to these values was considered 0, for any particular purpose of using the similarity value. Previous literature mentioned 0.06 (as in Martinez et al. (2018)) to obtain better community detection from networks generated from similarity values, for example, in the affinity miner prototype. Nonetheless, we discovered that, with our test conditions, this value of 0.12 threshold provided better evaluation metrics for clustering for our datasets in these experiments. These values should be better defined to have a clear notion of differences between using community detection or clustering from similarity values.

Another doubt that is equally related to thresholds is the one that comprises the use of Word2Vec and FastText. Adhoc tests showed that a threshold of similarity between documents below 0.6 and the corresponding setting of similarity value to 0 represented better metrics for clustering quality. This also represented better values for modularity, with the tested community detection algorithms, for example, the one in Cordeiro et al. (2016), which detects communities by optimizing modularity, and is available in the DynComm R package by Sarmento and Lemos (2020). This is equal to saying that, this time, the setting to 0.6 is equally related to better results either for modularity or general clustering evaluation metrics. Nonetheless, a better notion - for example, with a considerable amount of different datasets - how these thresholds change with different techniques for community detection or clustering analysis is yet to be discovered.

Another discussion point in this research comprises the model selected to test the hypothesis that contextualization gives better quality, even for the worst-case scenario of our algorithms. Either with IS-TFIDF or ITR, we are currently, in this part of the research, using the ODS model for the streaming flux, i.e., one document processing at any given moment of the flux of documents. This option is considered to have the lowest quality in achieving keywords from streaming documents. Thus, ITR algorithms' Top-K phase behaves as a Top-K selection of keywords, and Incremental PageRank is not being incremented at each new document, simply because every new document in the flux is what it is, a new document and not an update of an already known document. Either way, IS-TFIDF, and ITR show that, even with zero historical background of any kind, for any dataset, achieve and maintain along with the flux, reasonable evaluation metrics for the clustering quality. This is related to the contextualization with Word2Vec and FastText vector dictionaries.

5.3.4 Summary and Contributions

In this section we are concerned with the problem of generating keywords in stream setting, i.e., when we have a flux of documents coming in. One method uses focuses on keywords of the central documents in a set. Their respective keywords are also representative for other documents belonging to the respective group. Another method relies also on an evolving network. The evolving network is mapped into keywords, as this was done with ITR method, described in this section. Using algorithms for the calculations of word weights in particular documents and their changes as other documents stream in, we achieve a keyword/feature selection of important keywords or keyphrases in the evolving document stream. We have published various articles on the topic discussed in this section: Sarmento, et al., 2018; Sarmento, et al., 2021; Sarmento, et al., 2022.

Chapter 6

Discussion, Future Work and Conclusions

Our aim in this thesis was how to facilitate the processing of a relatively large set of documents that make a change in time. Our starting point was the existing Affinity Miner prototype, which did not really cope well with dynamically changing document streams. To upgrade this prototype, we have re-designed various methods to be able to continuously absorb new information and process it in an incremental manner. One such area involves the representation of documents with recourse to their attributes or keywords. Consequently, this makes the selected attributes change over time, and hence the methods must be able to work with dynamic and evolving attribute space, providing faster, and at the same time reliable search and analysis of the existing documents. This work is by no means complete. Various points regarding future research are discussed in Section 6.1, while Section 6.2 summarizes the main achievements of our work.

6.1 Future Work

Our work covered various aspects of document organization exemplified by the prototype system Affinity Miner. Improvements could be done to various parts of our work. In this part, we mention just a few directions that could be followed. One such direction involves keywords that play an important role both in our work and the prototype system, as they form the basis of calculating similarity among documents. Improvements could be made to how keywords are represented, and also, how they are identified.

In Chapter 4 we have discussed the method called Incremental Text Rank (ITR) that obtains the most important keywords from the flux of documents. An alternative to this would be to use the method of Latent Dirichlet Allocation (LDA) incrementally, following the work of Zhai and Boyd-Graber (2013). These authors created Online LDA with Infinite Vocabulary to extract topics from documents. They used a Top-K approach to store evolving vocabulary over time and, therefore, maintained a dictionary of terms. The incremental update of a vocabulary significantly improves the quality of the topic/keyword automatic extraction process. Another possible approach to consider would be the method for feature selection (in our case, features are words) proposed by Zhai and Boyd-Graber.

We note that many words have different meanings, depending on the context. So this situation could be addressed by improving the representation of keywords. This could be done by storing not only the keywords but also the associated context, as it is done for instance in Word2Vec, and mentioned in Chapter 5. Improvements in this could lead to another area involving studies that are oriented towards the usage of the existing system, incorporating various developed enhancements in specific tasks. This could be analyzed with a group A/B test and subjective qualitative and quantitative results could be retrieved. Below, we mention some questions that could be investigated:

- How easy it is to identify some target document(s) when compared to an unorganized list of documents in some directory?
- Does the system enable one to gain a better and faster understanding of the scientific production of a given institution (e.g. some University or one of its Faculties)?
- What are its strong subjective points and how can this be characterized?

6.2 Conclusions

One of the central themes in this thesis is that it is advantageous of having an organized set of documents prior to searching for the relevant ones. In recent research, the focus was mostly on relatively limited amounts of documents and static settings. Processing large amounts of documents represents in a stream setting represents a problem that needs to be addressed.

This is why one of our aims was to investigate sampling methods which are usually used when we have large amounts of documents. We have introduced the Top-K sampling method that focuses on extracting a sample of the most active nodes in the network. This is particularly important when dealing with massive amounts of data, as in the case of evolving networks. It enables us to cope with the flux of text documents. The main advantage of Top-K sampling algorithms is the great improvement in efficiency and memory usage.

Grouping similar documents into similar groups facilitates the search for a document that satisfies best the user's requirements. The user may go through the elements of the group to select one that matches best his/her requirements. This problem of grouping is addressed as a problem of "community detection" in a complex network. We have contributed to this area by designing a method for community detection in the context of evolving networks. One critical issue is which optimization measure should be used in this process. We have suggested using density optimization through ADC (Average density per Community), which maintain also good modularity values, as demonstrated by statistical tests. Higher values of ADC result in a better definition of smaller communities in the process of community detection. This leads to an improved document organization, particularly the smaller groups of similar documents.

The algorithms that help to identify documents with high centrality value are useful in the search for the right document that satisfies users' criteria. Many current methods that identify central elements are normally considered to be rather time-consuming due to high complexity. Our contribution in this area is Incremental Laplacian centrality that proved to be of lower complexity and, at the same time, rather efficient and effective. Having an efficient algorithm helps also to design better and more efficient methods for the task of folding, i.e., substituting a network with a node, and generation of keywords characterizing them.

Keywords play an important role because they enable the characterization of both individual documents and groups of documents. We have contributed to this area by designing a Window-based, incremental version of the graph-based algorithm TextRank (ITR). This version achieved similar quality but is much faster than the original algorithm in a streaming context. Additionally, we introduce a Top-K feature, which allows storing the most relevant keywords. We have designed a method that generates keywords in an evolving network. The evolving network is mapped into keywords, as this is done in the incremental version of the TextRank algorithm (ITR). Using algorithms for the calculations of word weights in particular documents and their changes as other documents stream in, we achieve a keyword/feature selection of important keywords or key phrases

in the evolving document stream.

One of the most problematic features of a graph-based representation of documents is its visualization. Unless this issue is attended to, the graph is cluttered with too much information, even for a modest number of documents. We have discussed various methods that can provide better ways for this task. These include folding and unfolding of groups of documents, the use of Ego-Networks (networks that include focal nodes), and visualization of central nodes, among others. We have thus contributed to the mitigation of this important problem.

The similarity between pairs of documents plays an important role in many methods of document organization. Our contribution to this task is in the design of a method that calculates the similarity in a streaming setting, that is when we have a flux of text documents. We have shown that it is possible to exploit a mapping of text documents and respective words in bipartite networks. This enables us to calculate the evolving measures of similarity between the documents in a stream. These include Incremental Sparse TF-IDF (IS-TFIDF) and Incremental Cosine Similarity (ICS). We can thus achieve more efficient algorithms (according to e.g., the speed-up ratio) when calculating a measure that was already considered of high complexity in a batch setting.

Author's Publications

The production by the author, related to this thesis, is presented here.

Sarmento, R., Cordeiro, M., and Gama, J. (2014a). visualization for streaming networks. In *Proceedings of the 3rd Workshop on New Frontiers in Mining Complex Patterns (NFMCP 2014)*, pages 62–74.

Sarmento, R., Cordeiro, M., and Gama, J. (2014b). visualization for streaming telecommunications networks. In *New Frontiers in Mining Complex Patterns*,pages 117–131. Springer International Publishing.

Sarmento, R., Cordeiro, M., and Gama, J. (2015a). Streaming networks sampling using Top-K networks. In *Proceedings of the 17th International Conference on Enterprise Information Systems*, pages 228–234.

Sarmento, R., Cordeiro, M., and Gama, J. (2015b). visualization of evolving large scale Ego-Networks. In *Proceedings of the 30th Annual ACM Symposiumon Applied Computing*, pages 960–962. ACM.

Sarmento, R., Oliveira, M., Cordeiro, M., Tabassum, S., and Gama, J. (2016a). Social network analysis in streaming call graphs. In *Big Data Analysis: New Algorithms for a New Society*, pages 239–261. Springer International Publishing.

Sarmento, R. and Costa, V. (2017). Comparative Approaches to Using R and Python for Statistical Data Analysis. IGI Global.

Sarmento, R. P. (2017). Metrics of evolving Ego-Networks with forgetting factor. In *International Journal of Social and Organizational Dynamics in IT (IJ-SODIT)*, 6(1):35–47.

Sarmento, R. P., Cordeiro, M., Brazdil, P., and Gama, J. (2017). Efficient Incremental Laplace centrality algorithm for dynamic networks. In *International Workshop on Complex Networks and*

their Applications, pages 341–352.Springer, Cham.

Sarmento, R. P., Cordeiro, M., Brazdil, P., and Gama, J. (2018). Incremental TextRank - automatic keyword extraction for text streams. In *Proceedings of the 20th International Conference on Enterprise Information Systems - Volume 1: ICEIS*, pages 363–370. INSTICC, SciTePress.

Sarmento, R. P., Lemos, L., Cordeiro, M., Rossetti, G., and de O. Cardos, D.(2019). Dyncomm R package - dynamic community detection for evolving networks.CoRR, abs/1905.01498.

Sarmento, R. P., Cardoso, D. O., Dearo, K., Brazdil, P., and Gama, J. (2021). Text documents streams with improved incremental similarity. In *Social network Analysis and Mining*, 11(1):113.

Sarmento, R. P., Cardoso, D. O., Brazdil, P., and Gama, J. (2022). Contextualization for the organization of text document streams. To be Published.

Brazdil, P., Trigo, L., Cordeiro, J., Sarmento, R., and Valizadeh, M. (2015). Affinity mining of documents sets via network analysis, keywords and summaries. In *Oslo Studies in Language*, 7(1).

Cordeiro, M., Sarmento, R. P., and Gama, J. (2016). Dynamic Community Detection in evolving networks using locality modularity optimization. In *Social network Analysis and Mining*, 6(1):1–20.

Cordeiro, M., Sarmento, R. P., Brazdil, P., Kimura, M., and Gama, J. (2019). Identifying, ranking and tracking community leaders in evolving social networks. In Cherifi, H., Gaito, S., Mendes, J. F., Moro, E., and Rocha, L. M.,editors, Complex Networks and Their Applications VIII - Volume 1 Proceedings of the Eighth International Conference on Complex Networks and Their Applications COMPLEX NETWORKS 2019, Lisbon, Portugal, December 10-12, 2019, In *volume 881 of Studies in Computational Intelligence*, pages 198–210.Springer.

Trigo, L., Vita, M., Sarmento, R., and Brazdil, P. (2015). Retrieval, visualization and Validation of Affinities Between documents. In J., A. D. J. J. K. A.,editor, Ic3K 2015 - In *Proceedings of the 7Th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management,volume 3 of Kmis*, pages 452–459.

References

James Abello and Frank van Ham. Matrix zoom: A visual interface to semi-external graphs. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS 2004)*, pages 183–190. IEEE Computer Society, 2004.

Charu Aggarwal and Karthik Subbian. Evolutionary network analysis: A survey. *ACM Computing Surveys (CSUR)*, 47(1):1–36, 2014. ISSN 03600300. doi: 10.1145/2601412. URL http://dl.acm.org/citation.cfm?id=2601412.

Charu C. Aggarwal and Philip S. Yu. A framework for clustering massive text and categorical data streams. In *Proceedings of the Sixth SIAM International Conference on Data Mining, April 20-22, 2006, Bethesda, MD, USA*, pages 479–483, 2006. doi: 10.1137/1.9781611972764.44. URL http://dx.doi.org/10.1137/1.9781611972764.44.

Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. A framework for projected clustering of high dimensional data streams. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, pages 852–863. VLDB Endowment, 2004. ISBN 0-12-088469-0. URL http://dl.acm.org/citation.cfm?id=1316689.1316763.

Nesreen K Ahmed, Jennifer Neville, and Ramana Kompella. Space-efficient sampling from social activity streams. In *Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications (BigMine 2012)*, pages 53–60. ACM, 2012.

Loulwah AlSumait, Daniel Barbará, and Carlotta Domeniconi. On-line lda: Adaptive topic models for mining text streams with applications to topic detection and tracking. In *ICDM*, pages 3–12. IEEE Computer Society, 2008. URL http://dblp.uni-trier.de/db/conf/icdm/icdm2008.html#AlSumaitBD08.

A Arenas, A Fernández, and S Gómez. Analysis of the structure of complex networks at different resolution levels. *New Journal of Physics*, 10(5):053039, may 2008. doi: 10.1088/1367-2630/10/5/053039. URL https://dx.doi.org/10.1088/1367-2630/10/5/053039.

Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and Issues in Data Stream Systems. In *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '02, pages 1–16, New York, NY, USA, 2002a. ACM. ISBN 1-58113-507-6. doi: 10.1145/543613.543615. URL http://doi.acm.org/10.1145/543613.543615.

Brian Babcock, Mayur Datar, and Rajeev Motwani. Sampling from a Moving Window over Streaming Data. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '02, pages 633–634, Philadelphia, PA, USA, 2002b. Society for Industrial

and Applied Mathematics. ISBN 0-89871-513-X. URL http://dl.acm.org/citation.cfm?id=545381.545465.

David M. Blei. Probabilistic topic models. *Commun. ACM*, 55(4):77–84, April 2012. ISSN 0001-0782. doi: 10.1145/2133806.2133826. URL http://doi.acm.org/10.1145/2133806.2133826.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=944919.944937.

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

Ulrich Bodenhofer, Andreas Kothmeier, and Sepp Hochreiter. Apcluster: an r package for affinity propagation clustering. *Bioinformatics*, 27:2463–2464, 2011. doi: 10.1093/bioinformatics/btr406.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

D. B. Bracewell, F. Ren, and S. Kuriowa. Multilingual single document keyword extraction for information retrieval. In *2005 International Conference on Natural Language Processing and Knowledge Engineering*, pages 517–522, Oct 2005. doi: 10.1109/NLPKE.2005.1598792.

Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25:163–177, 2001a.

Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163–177, 2001b.

Pavel Brazdil, Luís Trigo, João Cordeiro, Rui Sarmento, and Mohammadreza Valizadeh. Affinity mining of documents sets via network analysis, keywords and summaries. *Oslo Studies in Language*, 7(1), 2015.

S. Bugla. Name identification in scientific publications. Master's thesis, FCUP, University of Porto, Portugal, 2009.

Ronald S Burt. *Structural holes: The social structure of competition.* Harvard University Press, 1992.

J. M. Carmona Cejudo. *Nuevas tendencias en fundamentos teóricos aplicaciones de la minería de datos aplicada a la clasificación de textos en lenguaje natural.* PhD thesis, U. Salamanca, 2013.

Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming*, ICALP '02, pages 693–703, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-43864-5. URL http://dl.acm.org/citation.cfm?id=646255.684566.

Chaomei Chen and Steven Morris. Visualizing evolving networks: Minimum spanning trees versus pathfinder networks. In *Proceedings of the Ninth Annual IEEE Conference on Information Visualization*, INFOVIS'03, page 67–74, USA, 2003. IEEE Computer Society. ISBN 0780381548.

Mingming Chen, Tommy Nguyen, and Boleslaw K. Szymanski. On measuring the quality of a network community structure. In *2013 International Conference on Social Computing*, pages 122–127, 2013. doi: 10.1109/SocialCom.2013.25.

Mingming Chen, Tommy Nguyen, and Boleslaw K. Szymanski. A new metric for quality of network community structure. *CoRR*, abs/1507.04308, 2015. URL http://arxiv.org/abs/1507.04308.

Several Contributors. Python recipes - igraph, Oct 2021. URL http://igraph.wikidot.com/python-recipes#toc4. [Online; accessed 11. Oct. 2021].

Mário Cordeiro, Rui Portocarrero Sarmento, and João Gama. Dynamic community detection in evolving networks using locality modularity optimization. *Social Network Analysis and Mining*, 6(1):1–20, 2016. ISSN 1869-5469. doi: 10.1007/s13278-016-0325-1. URL http://dx.doi.org/10.1007/s13278-016-0325-1.

Mário Cordeiro, Rui P Sarmento, Pavel Brazdil, and João Gama. Evolving networks and social network analysis methods and techniques. In *Social Media and Journalism-Trends, Connections, Implications*. IntechOpen, 2018.

Graham Cormode and S Muthukrishnan. What's hot and what's not: tracking most frequent items dynamically. *ACM Transactions on Database Systems*, 30(1):249–278, 2005.

David Corney, Dyaa Albakour, Miguel Martinez, and Samir Moussa. What do a million news articles look like? In *Proceedings of the First International Workshop on Recent Trends in News Information Retrieval co-located with 38th European Conference on Information Retrieval (ECIR 2016), Padua, Italy, March 20, 2016.*, pages 42–47, 2016. URL http://ceur-ws.org/Vol-1568/paper8.pdf.

Jean-Valère Cossu, Vincent Labatut, and Nicolas Dugué. A review of features for the discrimination of twitter users: application to the prediction of offline influence. *Social Network Analysis and Mining*, 6(1):25, May 2016. ISSN 1869-5469. doi: 10.1007/s13278-016-0329-x. URL https://doi.org/10.1007/s13278-016-0329-x.

Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006. URL http://igraph.org.

M Datar, A Gionis, P Indyk, and R Motwani. Maintaining stream statistics over sliding windows. *Proceedings of the thirteenth annual {ACM-SIAM} symposium on Discrete algorithms*, pages 635–644, 2002. ISSN 0097-5397. doi: 10.1137/S0097539701398363.

Rich DeJordy and Dan Halgin. Introduction into ego network analysis, 2009.

Erik D Demaine, Alejandro López-Ortiz, and J Ian Munro. Frequency estimation of internet packet streams with limited space. In *Algorithms-ESA 2002*, pages 348–360. Springer, 2002.

Prasanna Desikan, Nishith Pathak, Jaideep Srivastava, and Vipin Kumar. Incremental page rank computation on evolving graphs. In *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web*, WWW '05, pages 1094–1095, New York, NY, USA, 2005. ACM. ISBN 1-59593-051-5. doi: 10.1145/1062745.1062885. URL http://doi.acm.org/10.1145/1062745.1062885.

Dirk Eddelbuettel. *Seamless R and C++ Integration with Rcpp*. Springer, New York, 2013. doi: 10.1007/978-1-4614-6868-4. ISBN 978-1-4614-6867-7.

Dirk Eddelbuettel and James Joseph Balamuta. Extending extitR with extitC++: A Brief Introduction to extitRcpp. *PeerJ Preprints*, 5:e3188v1, aug 2017. ISSN 2167-9843. doi: 10.7287/peerj.preprints.3188v1. URL https://doi.org/10.7287/peerj.preprints.3188v1.

Dirk Eddelbuettel and Romain François. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011. doi: 10.18637/jss.v040.i08. URL http://www.jstatsoft.org/v40/i08/.

Niklas Elmqvist, Thanh-Nghi Do, Howard Goodell, Nathalie Henry, and Jean-Daniel Fekete. ZAME: Interactive Large-Scale Graph Visualization. In *IEEE Pacific Visualization Symposium 2008 (PacificVIS 2008)*, pages 215–222, Kyoto, Japan, 2008. IEEE Computer Society.

Ingo Feinerer and Kurt Hornik. *tm: Text Mining Package*, 2018. URL https://CRAN.R-project.org/package=tm. R package version 0.7-5.

Ingo Feinerer, Kurt Hornik, and David Meyer. Text mining infrastructure in r. *Journal of Statistical Software*, 25(5):1–54, March 2008. URL http://www.jstatsoft.org/v25/i05/.

Ronen Feldman and James Sanger. *Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, New York, NY, USA, 2006. ISBN 0521836573, 9780521836579.

Ronen Feldman and James Sanger. *The text mining handbook: advanced approaches in analyzing unstructured data*. Cambridge University Press, 2007.

Robert W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345–, June 1962. ISSN 0001-0782. doi: 10.1145/367766.368168. URL http://doi.acm.org/10.1145/367766.368168.

C. L. Fowlkes, E. B. & Mallows. A Method for Comparing Two Hierarchical Clusterings. *Journal of the American Statistical Association*, 78(383):553–569, 1983. ISSN 01621459. doi: 10.2307/2288117. URL http://dx.doi.org/10.2307/2288117.

Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. *Science*, 315:972–977, 2007. doi: 10.1126/science.1136800.

Ines Färber, Stephan Günnemann, Hans peter Kriegel, Peer Kröger, Emmanuel Müller, Erich Schubert, Thomas Seidl, and Arthur Zimek. On using class-labels in evaluation of clusterings. In *In MultiClust: 1st International Workshop on Discovering, Summarizing and Using Multiple Clusterings Held in Conjunction with KDD 2010*, 2010.

Joao Gama. *Knowledge Discovery from Data Streams*. Chapman & Hall/CRC, 1st edition, 2010. ISBN 1439826110, 9781439826119.

Emden R. Gansner, Yifan Hu, and Stephen C. North. Interactive visualization of streaming text data with dynamic maps. *J. Graph Algorithms Appl.*, 17(4):515–540, 2013. URL http://dblp.uni-trier.de/db/journals/jgaa/jgaa17.html#GansnerHN13.

Dan Garber, Elad Hazan, and Tengyu Ma. Online Learning of Eigenvectors. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *JMLR Proceedings*, pages 560–568. JMLR.org, 2015.

Luit Gazendam, Christian Wartena, and Rogier Brussee. Thesaurus based term ranking for keyword extraction. In *Database and Expert Systems Applications, DEXA, International Workshops, Bilbao, Spain, August 30 - September 3, 2010*, pages 49–53, 2010. doi: 10.1109/DEXA.2010.31. URL https://doi.org/10.1109/DEXA.2010.31.

Johannes Gehrke, Flip Korn, and Divesh Srivastava. On computing correlated aggregates over continual data streams. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data - SIGMOD '01*, pages 13–24, 2001. ISBN 1581133324. doi: 10.1145/375663.375665. URL http://portal.acm.org/citation.cfm?doid=375663.375665.

Leo A Goodman. Snowball sampling. *The Annals of Mathematical Statistics*, 32(1):148–170, 1961.

A.D. Gordon. *Classification, 2nd Edition*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1999. ISBN 9781584880134. URL https://books.google.pt/books?id=jW41nwEACAAJ.

M. Granovetter. Network sampling: Some first steps. *American Journal of Sociology*, 81(6): 1267–1303, 1976.

Oded Green, Robert McColl, and David A Bader. A fast algorithm for streaming betweenness centrality. In *Proceedings of the 2012 International Conference on Privacy, Security, Risk and Trust (PASSAT 2012) and 2012 International Conference on Social Computing (SocialCom 2012)*, pages 11–20. IEEE Computer Society, 2012.

T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl. 1):5228–5235, April 2004.

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

David Hall, Daniel Jurafsky, and Christopher D. Manning. Studying the history of ideas using topic models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 363–371, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1613715.1613763.

Frank Ham, Hans-Jörg Schulz, and Joan M. Dimicco. Honeycomb: Visual analysis of large scale social networks. In *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part II (INTERACT 2009)*, pages 429–442. Springer-Verlag, 2009.

R.A. Hanneman and M. Riddle. *Introduction to Social Network Methods*. University of California, 2005a. URL http://books.google.pt/books?id=wAHaygAACAAJ.

Robert A. Hanneman and Mark Riddle. *Introduction to Social Network Methods*. University of California, Riverside, Riverside, CA, USA, 2005b. URL http://www.faculty.ucr.edu/~hanneman/nettext/index.html.

Alexandre Hollocou, Julien Maudet, Thomas Bonald, and Marc Lelarge. A linear streaming algorithm for community detection in very large networks. *CoRR*, abs/1703.02955, 2017. URL http://arxiv.org/abs/1703.02955.

L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985. URL http://scholar.google.de/scholar.bib?q=info:IkrWWF2JxwoJ:scholar.google.com/&output=citation&hl=de&ct=citation&cd=0.

Christian Hubler, H-P Kriegel, Karsten Borgwardt, and Zoubin Ghahramani. Metropolis algorithms for representative subgraph sampling. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008)*, pages 283–292. IEEE Computer Society, 2008.

D. Iacobucci. *Graphs and Matrices. In: Wasserman, S. (eds), Social network analysis: methods and applications. PP. 92-166.* Cambridge University Press, New York, 1994.

INESC-TEC. Inesc tec. http://www.inesctec.pt/, 2015.

Kaggle. TED Talks, Dec 2021. URL https://www.kaggle.com/rounakbanik/ted-talks. [Online; accessed 12. Dec. 2021].

Miray Kas, Kathleen M. Carley, and L. Richard Carley. Incremental closeness centrality for dynamically changing social networks. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '13, pages 1250–1258, New York, NY, USA, 2013a. ACM. ISBN 978-1-4503-2240-9. doi: 10.1145/2492517.2500270. URL http://doi.acm.org/10.1145/2492517.2500270.

Miray Kas, Kathleen M Carley, and L Richard Carley. Incremental closeness centrality for dynamically changing social networks. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*, pages 1250–1258. IEEE Computer Society, 2013b.

Miray Kas, Matthew Wachs, Kathleen M Carley, and L Richard Carley. Incremental algorithm for updating betweenness centrality in dynamically growing networks. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*, pages 33–40. IEEE Computer Society, 2013c.

Sushant S Khopkar, Rakesh Nagi, Alexander G Nikolaev, and Vaibhav Bhembre. Efficient algorithms for incremental all pairs shortest paths, closeness and betweenness in social network analysis. *Social Network Analysis and Mining*, 4(1):1–20, 2014.

Hyoungshick Kim and Ross Anderson. Temporal node centrality in complex networks. *Physical Review E*, 85(2):026107, 2012.

Bennett Kleinberg, van der Vegt Isabelle, and Maximilian Mozes. Papers with Code - RWWD Dataset, Dec 2021. URL https://paperswithcode.com/dataset/rwwd. [Online; accessed 7. Dec. 2021].

Nicolas Kourtellis, Gianmarco De Francisci Morales, and Francesco Bonchi. Scalable online betweenness centrality in evolving graphs. *arXiv preprint arXiv:1401.6981*, 2014.

Andrea Lancichinetti and Santo Fortunato. Limits of modularity maximization in community detection. *Phys. Rev. E*, 84:066122, Dec 2011. doi: 10.1103/PhysRevE.84.066122. URL https://link.aps.org/doi/10.1103/PhysRevE.84.066122.

Bongshin Lee, Catherine Plaisant, Cynthia Sims Parr, Jean-Daniel Fekete, and Nathalie Henry. Task taxonomy for graph visualization. In *Proceedings of the 2006 AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization (BELIV 2006)*, pages 1–5. ACM, 2006.

Min-Joong Lee, Jungmin Lee, Jaimie Yejean Park, Ryan Hyun Choi, and Chin-Wan Chung. QUBE: a Quick algorithm for Updating BEtweenness centrality. In *Proceedings of the 21st International Conference on World Wide Web*, pages 351–360. ACM, 2012.

Pierre Legendre and Louis Legendre. *Numerical ecology*. Elsevier, 2012.

Yang Lei, James C. Bezdek, Simone Romano, Nguyen Xuan Vinh, Jeffrey Chan, and James Bailey. Ground truth bias in external cluster validity indices. *Pattern Recogn.*, 65(C):58–70, may 2017. ISSN 0031-3203. doi: 10.1016/j.patcog.2016.12.003. URL https://doi.org/10.1016/j.patcog.2016.12.003.

Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006)*, pages 631–636. ACM, 2006.

Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining - KDD '05*, page 177, New York, New York, USA, August 2005. ACM Press. ISBN 159593135X. doi: 10.1145/1081870.1081893.

Ian X Y Leung, Pan Hui, Pietro Liò, and Jon Crowcroft. Towards real-time community detection in large networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 79(6):1–10, 2009. ISSN 15393755. doi: 10.1103/PhysRevE.79.066107.

Wei Li and Andrew McCallum. Pachinko Allocation: DAG-structured Mixture Models of Topic Correlations. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 577–584, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2. doi: 10.1145/1143844.1143917. URL http://dx.doi.org/10.1145/1143844.1143917.

Chin-Yew Lin and Eduard H. Hovy. Identifying topics by position. In *ANLP*, pages 283–290, 1997. URL http://dblp.uni-trier.de/db/conf/anlp/anlp1997.html#LinH97.

Marina Litvak, Mark Last, Hen Aizenman, Inbal Gobits, and Abraham Kandel. Degext — a language-independent graph-based keyphrase extractor. In Elena Mugellini, Piotr S. Szczepaniak, Maria Chiara Pettenati, and Maria Sokhn, editors, *Advances in Intelligent Web Mastering – 3: Proceedings of the 7th Atlantic Web Intelligence Conference, AWIC 2011, Fribourg, Switzerland, January, 2011*, pages 121–130, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-18029-3. doi: 10.1007/978-3-642-18029-3_13. URL https://doi.org/10.1007/978-3-642-18029-3_13.

Yu-Bao Liu, Jia-Rong Cai, Jian Yin, and Ada Wai-Chee Fu. Clustering text data streams. *Journal of Computer Science and Technology*, 23(1):112–128, 2008. ISSN 1860-4749. doi: 10.1007/s11390-008-9115-1. URL http://dx.doi.org/10.1007/s11390-008-9115-1.

Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 257–266, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-59-6. URL http://dl.acm.org/citation.cfm?id=1699510.1699544.

G. S. Manku and R. Motwani. Approximate frequency counts over data streams. In *Proceedings of the 28th International Conference on Very Large Data Bases*, 2002. URL citeseer.ist.psu.edu/manku02approximate.html.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715.

André Martinez, Pavel Brazdil, and Luís Trigo. Analysis of Publications of FEP and their Affinities. FEP Working Papers 599, Universidade do Porto, Faculdade de Economia do Porto, January 2018. URL https://ideas.repec.org/p/por/fepwps/599.html.

Olaf Mersmann. *microbenchmark: Accurate Timing Functions*, 2018. URL https://CRAN.R-project.org/package=microbenchmark. R package version 1.4-6.

Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. Efficient computation of frequent and top-k elements in data streams. In *Proceedings of the 10th International Conference on Database Theory*, ICDT'05, pages 398–412, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 3-540-24288-0, 978-3-540-24288-8. doi: 10.1007/978-3-540-30570-5_27. URL http://dx.doi.org/10.1007/978-3-540-30570-5_27.

R. Mihalcea and P. Tarau. TextRank: Bringing order into texts. In *Proceedings of EMNLP-04and the 2004 Conference on Empirical Methods in Natural Language Processing*, July 2004a.

Rada Mihalcea and Paul Tarau. TextRank: Bringing Order into Texts. In *Conference on Empirical Methods in Natural Language Processing*, Barcelona, Spain, 2004b. URL http://acl.ldc.upenn.edu/acl2004/emnlp/pdf/Mihalcea.pdf.

Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. URL http://arxiv.org/abs/1301.3781.

Lampros Mouselimis. *ClusterR: Gaussian Mixture Models, K-Means, Mini-Batch-Kmeans, K-Medoids and Affinity Propagation Clustering*, 2020. URL https://CRAN.R-project.org/package=ClusterR. R package version 1.2.2.

Fionn Murtagh. *Multidimensional clustering algorithms*. Physica-Verlag, 1985.

Fionn Murtagh and Pierre Legendre. Ward's hierarchical agglomerative clustering method: which algorithms implement ward's criterion? *Journal of classification*, 31(3):274–295, 2014.

Meghana Nasre, Matteo Pontecorvi, and Vijaya Ramachandran. Betweenness centrality - incremental and faster. *CoRR*, abs/1311.2147, 2013. URL http://dblp.uni-trier.de/db/journals/corr/corr1311.html#NasrePR13.

M E J Newman and M Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113+, 2004. doi: 10.1103/physreve.69.026113.

Márcia D. B. Oliveira, Américo Guerreiro, and João Gama. Dynamic communities in evolving customer networks: an analysis using landmark and sliding windows. *Social Netw. Analys. Mining*, 4(1):208, 2014. doi: 10.1007/s13278-014-0208-2. URL http://dx.doi.org/10.1007/s13278-014-0208-2.

L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. In *Proceedings of the 7th International World Wide Web Conference*, pages 161–172, Brisbane, Australia, 1998. URL citeseer.nj.nec.com/page98pagerank.html.

Manos Papagelis, Gautam Das, and Nick Koudas. Sampling online social networks. *IEEE Transactions on Knowledge and Data Engineering*, 25(3):662–676, 2013.

Michael J. Paul and Roxana Girju. Topic modeling of research fields: An interdisciplinary perspective. In Galia Angelova, Kalina Bontcheva, Ruslan Mitkov, Nicolas Nicolov, and Nikolai Nikolov, editors, *RANLP*, pages 337–342. RANLP 2009 Organising Committee / ACL, 2009. URL http://dblp.uni-trier.de/db/conf/ranlp/ranlp2009.html#PaulG09.

Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. In *Proceedings of the 20th International Conference on Computer and Information Sciences*, ISCIS'05, pages 284–293, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 3-540-29414-7, 978-3-540-29414-6. doi: 10.1007/11569596_31. URL http://dx.doi.org/10.1007/11569596_31.

Xingqin Qi, Eddie Fuller, Qin Wu, Yezhou Wu, and Cun-Quan Zhang. Laplacian centrality: A new centrality measure for weighted networks. *Inf. Sci.*, 194:240–253, July 2012. ISSN 0020-0255. doi: 10.1016/j.ins.2011.12.027. URL http://dx.doi.org/10.1016/j.ins.2011.12.027.

Xingqin Qi, Robert D. Duval, Kyle Christensen, Edgar Fuller, Arian Spahiu, Qin Wu, Yezhou Wu, Wenliang Tang, and Cunquan Zhang. Terrorist Networks, Network Energy and Node Removal: A New Measure of Centrality Based on Laplacian Energy. *Social Networking*, 02(01):19–31, 2013. ISSN 2169-3285. doi: 10.4236/sn.2013.21003.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020. URL https://www.R-project.org/.

Davood Rafiei. Effectively visualizing large networks through sampling. In *Proceedings of the IEEE Visualization 2005 (VIS 2005)*, pages 375–382. IEEE Computer Society, 2005.

Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3):036106, 2007. ISSN 15393755. doi: 10.1103/PhysRevE.76.036106. URL http://link.aps.org/doi/10.1103/PhysRevE.76.036106{%}5Cnhttp://journals.aps.org/pre/abstract/10.1103/PhysRevE.76.036106.

G. Ramalingam and Thomas Reps. An incremental algorithm for a generalization of the shortest-path problem. *J. Algorithms*, 21(2):267–305, September 1996a. ISSN 0196-6774. doi: 10.1006/jagm.1996.0046. URL http://dx.doi.org/10.1006/jagm.1996.0046.

Ganesan Ramalingam and Thomas Reps. On the computational complexity of dynamic graph problems. *Theoretical Computer Science*, 158(1):233–277, 1996b.

W.M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

RDICCR. Published data from r&d is of the czech republic - research, development and innovation council of the czech republic (rdiccr). http://www.isvav.cz/, 2015. Accessed: 2015-09-30.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. Automatic keyword extraction from individual documents. In Michael W. Berry and Jacob Kogan, editors, *Text Mining. Applications and Theory*, pages 1–20. John Wiley and Sons, Ltd, 2010. ISBN 9780470689646. doi: 10.1002/9780470689646.ch1. URL http://dx.doi.org/10.1002/9780470689646.ch1.

Martin Rosvall and Carl T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008. doi: 10.1073/pnas.0706851105. URL http://www.pnas.org/content/105/4/1118.abstract.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

Ahmet Erdem Sariyüce, Bugra Gedik, Gabriela Jacques-Silva, Kun-Lung Wu, and Ümit V. Çatalyürek. SONIC: streaming overlapping community detection. *Data Min. Knowl. Discov.*, 30(4):819–847, 2016. doi: 10.1007/s10618-015-0440-z. URL https://doi.org/10.1007/s10618-015-0440-z.

Rui Sarmento. Large scale social network analysis. Master's thesis, School of Economics and Business, U. of Porto, 2013.

Rui Sarmento, Tiago Cunha, João Gama, and Albert Bifet. Large scale social networks analysis. In *JOCLAD 2013*, 2013.

Rui Sarmento, Mário Cordeiro, and João Gama. Visualization for streaming telecommunications networks. In *New Frontiers in Mining Complex Patterns*, pages 117–131. Springer International Publishing, 2014a.

Rui Sarmento, Mário Cordeiro, and João Gama. Visualization for streaming networks. In *Proceedings of the 3rd Workshop on New Frontiers in Mining Complex Patterns (NFMCP 2014)*, pages 62–74, 2014b.

Rui Sarmento, Mário Cordeiro, and João Gama. Streaming networks sampling using top-k networks. In *ICEIS 2015 - Proceedings of the 17th International Conference on Enterprise Information Systems, Volume 1, Barcelona, Spain, 27-30 April, 2015*, pages 228–234, 2015a. doi: 10.5220/0005341402280234. URL http://dx.doi.org/10.5220/0005341402280234.

Rui Sarmento, Mário Cordeiro, and João Gama. Visualization of evolving large scale ego-networks. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 960–962. ACM, 2015b.

Rui Sarmento, Mário Cordeiro, and João Gama. Streaming networks sampling using top-k networks. In *Proceedings of the 17th International Conference on Enterprise Information Systems (ICEIS 2015)*, page to appear. INSTICC, 2015c.

Rui P. Sarmento and Luís Lemos. *DynComm: Dynamic Network Communities Detection and Generation*, 2020. URL https://CRAN.R-project.org/package=DynComm. R package version 2020.1.6.

Rui Portocarrero Sarmento. Density-based Community Detection/Optimization. *arXiv*, Apr 2019. URL https://arxiv.org/abs/1904.12593.

Rui Portocarrero Sarmento, Mário Cordeiro, Pavel Brazdil, and João Gama. Efficient incremental laplace centrality algorithm for dynamic networks. In *Complex Networks & Their Applications VI - Proceedings of Complex Networks 2017 (The Sixth International Conference on Complex Networks and Their Applications), COMPLEX NETWORKS 2017, Lyon, France, November 29 - December 1, 2017.*, pages 341–352, 2017. doi: 10.1007/978-3-319-72150-7_28. URL https://doi.org/10.1007/978-3-319-72150-7_28.

Rui Portocarrero Sarmento, Mário Cordeiro, Pavel Brazdil, and João Gama. Incremental textrank - automatic keyword extraction for text streams. In *Proceedings of the 20th International Conference on Enterprise Information Systems, ICEIS 2018, Funchal, Madeira, Portugal, March 21-24, 2018, Volume 1.*, pages 363–370, 2018a. doi: 10.5220/0006639703630370. URL https://doi.org/10.5220/0006639703630370.

Rui Portocarrero Sarmento, Mário Cordeiro, Pavel Brazdil, and João Gama. Incremental textrank - automatic keyword extraction for text streams. In *Proceedings of the 20th International Conference on Enterprise Information Systems - Volume 1: ICEIS,*, pages 363–370. INSTICC, SciTePress, 2018b. ISBN 978-989-758-298-1. doi: 10.5220/0006639703630370.

Rui Portocarrero Sarmento, Douglas O. Cardoso, Kemmily Dearo, Pavel Brazdil, and João Gama. Text documents streams with improved incremental similarity. *Soc. Netw. Anal. Min.*, 11(1):113, 2021. doi: 10.1007/s13278-021-00826-z. URL https://doi.org/10.1007/s13278-021-00826-z.

David Schoch. *networkdata: Repository of Network Datasets*, 2021. URL https://github.com/schochastics/networkdata. R package version 0.1.9.

Roger W. Schvaneveldt, editor. *Pathfinder Associative Networks: Studies in Knowledge Organization*. Ablex Publishing Corp., Norwood, NJ, USA, 1990. ISBN 0-89391-624-2.

R.W. Schvaneveldt, D.W. Dearholt, and F.T. Durso. Graph theoretic foundations of pathfinder networks. *Computers & Mathematics with Applications*, 15(4):337 – 345, 1988. ISSN 0898-1221. doi: http://dx.doi.org/10.1016/0898-1221(88)90221-0. URL http://www.sciencedirect.com/science/article/pii/0898122188902210.

S. Sendhilkumar, Nachiyar S N, and G. S. Mahalakshmi. Novelty detection via topic modeling in research articles, 2013.

Lei Shi, Nan Cao, Shixia Liu, Weihong Qian, Li Tan, Guodong Wang, Jimeng Sun, and Ching-Yung Lin. Himap: Adaptive visualization of large-scale online social networks. In *IEEE Pacific Visualization Symposium 2009 (PacificVIS 2009)*, pages 41–48. IEEE Computer Society, 2009.

Yongwook Shin, Chuh Yeop Ryo, and Jonghun Park. Automatic extraction of persistent topics from social text streams. *World Wide Web*, 17(6):1395–1420, 2014.

Sifatullah Siddiqi and Aditi Sharan. Article: Keyword and keyphrase extraction techniques: A literature review. *International Journal of Computer Applications*, 109(2):18–23, January 2015. Full text available.

Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005. ISBN 0321321367.

Ivan Titov and Ryan T. McDonald. Modeling online reviews with multi-grain topic models. *CoRR*, abs/0801.1063, 2008. URL http://dblp.uni-trier.de/db/journals/corr/corr0801.html#abs-0801-1063.

Luís Trigo and Pavel Brazdil. Affinity analysis between researchers using text mining and differential analysis of graphs. In *ECML/PKDD 2014 PhD Session Proceedings*, 2014. URL https://phdsession-ecmlpkdd2014.greyc.fr/sites/phdsession-ecmlpkdd2014.greyc.fr/files/papers/Paper_20702.pdf.

Luis Trigo., Martin Víta., Rui Sarmento., and Pavel Brazdil. Retrieval, visualization and validation of affinities between documents. In *Proceedings of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - KITA, (IC3K 2015)*, pages 452–459. INSTICC, SciTePress, 2015. ISBN 978-989-758-158-8. doi: 10.5220/0005662904520459.

Simon Urbanek. *fastmatch: Fast match() function*, 2017. URL https://CRAN.R-project.org/package=fastmatch. R package version 1.1-0.

Chang-Dong Wang, Jian-Huang Lai, and Philip S. Yu. Dynamic Community Detection in Weighted Graph Streams. *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 151–161, 2013. doi: 10.1137/1.9781611972832.17. URL http://epubs.siam.org/doi/abs/10.1137/1.9781611972832.17.

S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994a.

Stanley Wasserman and Katherine Faust. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994b. URL http://scholar.google.com/scholar.bib?q=info:gET6m8icitMJ:scholar.google.com/&output=citation&hl=en&as_sdt=0,5&as_vis=1&ct=citation&cd=0.

Andrew P. Wheeler. Laplacian centrality in networkx (python). https://andrewpwheeler.wordpress.com/2015/07/29/laplacian-centrality-in-networkx-python/, 2015. [Online; accessed April-2017].

Se-Young Yun and Alexandre Proutière. Community detection via random and adaptive sampling. *CoRR*, abs/1402.3072, 2014. URL http://arxiv.org/abs/1402.3072.

Se-Young Yun, Marc Lelarge, and Alexandre Proutière. Streaming, memory limited algorithms for community detection. *CoRR*, abs/1411.1279, 2014. URL http://arxiv.org/abs/1411.1279.

Ke Zhai and Jordan L. Boyd-Graber. Online latent dirichlet allocation with infinite vocabulary. In *ICML (1)*, volume 28 of *JMLR Proceedings*, pages 561–569. JMLR.org, 2013. URL http://dblp.uni-trier.de/db/conf/icml/icml2013.html#ZhaiB13.

X. Zhang, X. Zhou, and X. Hu. Semantic smoothing for model-based document clustering. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 1193–1198, Dec 2006. doi: 10.1109/ICDM.2006.142.

Xiaohua Zhou, Xiaohua Hu, Xiaodan Zhang, Xia Lin, and Il-Yeol Song. Context-sensitive semantic smoothing for the language modeling approach to genomic ir. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 170–177, New York, NY, USA, 2006. ACM. ISBN 1-59593-369-7. doi: 10.1145/1148170.1148203. URL http://doi.acm.org/10.1145/1148170.1148203.

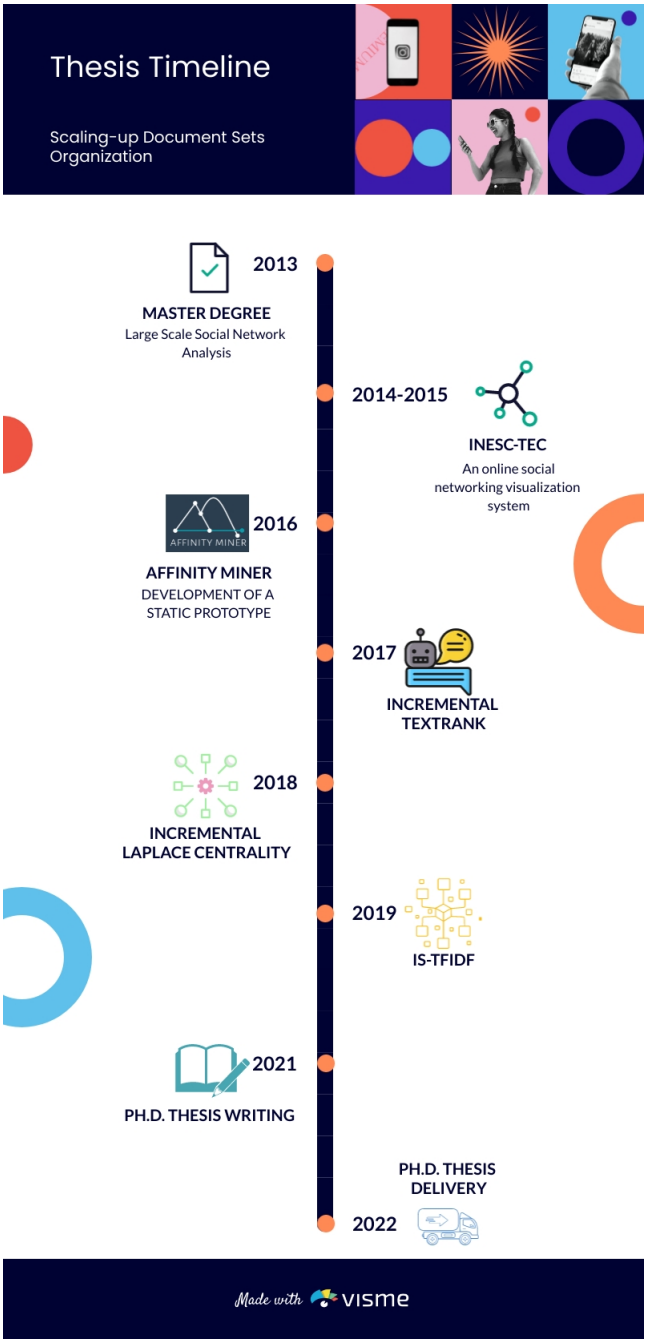
Appendix A

Patent Pending

The Incremental TextRank algorithm developed in the Thesis gave origin to a Patent submission. In the link [1], we transcribe the submission document.

Temporal Line of Developments

[1] https://worldwide.espacenet.com/publicationDetails/originalDocument?CC=EP&NR=3528144A1&KC=A1&FT=D&ND=&date=20190821&DB=EPODOC&locale=en_EP


Thesis Timeline
Scaling-up Document Sets Organization
2013
MASTER DEGREE
Large Scale Social Network Analysis
2014-2015
INESC-TEC
An online social networking visualization system
2016
AFFINITY MINER
AFFINITY MINER
DEVELOPMENT OF A STATIC PROTOTYPE
2017
INCREMENTAL TEXTRANK
2018
INCREMENTAL LAPLACE CENTRALITY
2019
IS-TFIDF
2021
PH.D. THESIS WRITING
PH.D. THESIS DELIVERY
2022
Made with visme

Index

centrality, 7, 8, 11, 12, 20–23, 30, 31, 50, 54, 60, 70–74, 125
centroids, 60, 89
clustering, 11, 13–19, 24, 37, 38, 41, 43–47, 54, 59, 78, 80, 103, 111–113, 115, 116, 118
community detection, 11, 12, 36, 50, 54, 56–66, 68, 118, 126

density, 19, 26, 45, 57, 59, 61–63, 65
document, 3, 8, 9, 11, 12, 16, 24, 25, 29–31, 35–38, 40, 42, 43, 47, 56, 57, 70, 77–80, 82, 84, 85, 92, 101, 102, 104–107, 111–115, 117, 118, 126, 139

Ego-Networks, 89, 91–93, 97, 99, 125

Incremental Laplace, 71–74, 125
Incremental PageRank, 82, 85, 118
Incremental Similarity, 101, 112
Incremental TextRank, 78, 80, 81, 83, 84, 126, 139
IS-TFIDF, 104–109, 112, 113, 115–118

keywords, 2, 12, 24, 25, 29, 31–33, 37–41, 43–47, 60, 77, 78, 81–88, 107, 111–113, 118, 126

Louvain, 12, 58–61

matrix, 12, 15, 19, 26, 27, 30, 31, 36–38, 41, 44, 89, 101–104, 107

network, 2, 7, 12, 18–21, 23, 26, 29–33, 35, 36, 47, 50, 51, 53–61, 63, 66, 70–75, 81, 82, 89–97, 99, 103, 104, 125, 126

prototype, 8, 25, 29, 30, 32, 34–36, 47, 60, 80, 89, 103, 111, 118

TextRank, 24, 25, 31, 77, 78, 80–88, 126, 139
Top-K, 50–52, 54, 59, 77, 81–85, 88, 90, 113, 118, 125

visualization, 8, 11, 12, 19, 35, 47, 50, 54, 60, 89–91, 93, 97, 99, 125, 126

Word2Vec, 111–113, 115, 116, 118