Islamic University – Gaza
Deanery of Post Graduate Studies
Faculty of information technology
Master of Information Technology

الجـــامعة الإسلاميـــة – غـــزة
شئـــون البحث العلمـــي والدراســـات العليـــا
كليـــــة تكنـــولـــوجيـــــا المعلـومـــــات
مـــــاجستير تكنـــولـــوجيـــا المعلـومـــــات

Tag Recommendation for Short Arabic Text by Using Latent Semantic Analysis of Wikipedia

اقتراح أوسمة للنصوص العربية القصيرة باستخدام تحليل الدلالات الكامنة على الويكيبيديا العربية

Yousef K. Abu Samra

Supervised By:

Dr. Iyad M. Alagha

Assistant Professor of Computer Science

A thesis submitted in partial fulfilment of the requirements for the degree of Master of Information Technology

April/2017

إقـــــرار

أنا الموقع أدناه مقدم الرسالة التي تحمل العنوان:

Tag Recommendation for Short Arabic Text by Using Latent Semantic Analysis of Wikipedia

اقتراح أوسمة للنصوص العربية القصيرة باستخدام تحليل الدلالات الكامنة على الويكيبيديا العربية

أقر بأن ما اشتملت عليه هذه الرسالة إنما هو نتاج جهدي الخاص، باستثناء ما تمت الإشارة إليه حيثما ورد، وأن هذه الرسالة ككل أو أي جزء منها لم يقدم من قبل الاخرين لنيل درجة أو لقب علمي أو بحثي لدى أي مؤسسة تعليمية أو بحثية أخرى.

Declaration

I understand the nature of plagiarism, and I am aware of the University's policy on this.

The work provided in this thesis, unless otherwise referenced, is the researcher's own work, and has not been submitted by others elsewhere for any other degree or qualification.

Student's name:	يوسف خميس أبو سمرة	اسم الطالب:
Signature:		التوقيع:
Date:		التاريخ:

Abstract

Social media sites enable users to share items, such as texts and images, and annotate them with freely chosen keywords called tags. However, freedom comes at a cost: uncontrolled vocabulary can result in tag redundancy, ambiguity, sparsity, miss-spilling, and idiosyncrasy, thus impeding more effective organization/retrieval of resources in tagging systems.

This work proposes an Arabic Language tag recommender system that exploits the Arabic Wikipedia as background knowledge. Latent semantic analysis was employed to discover hidden semantics between the short text and Wikipedia articles. Apache Spark was used to handle the massive content of Wikipedia and the complex computations of latent semantic analysis which is used to analyze Wikipedia articles into three matrices. Given an Arabic short text as input, the system compares it to the body of the articles and scores them according to their relevance to the short text. Candidate tags are determined from top-scored articles by exploiting articles' titles and categories.

The proposed system was assessed over a dataset of 100 tweets covering three different domains. Generated tags were rated by two human experts in each domain. Our system achieved 84.39% mean average precision and 96.53% mean reciprocal rank, revealing the system adequacy and accuracy for tagging Arabic short texts while still has difficulties regarding Arabic language, and affected by frequencies of rare terms. A thorough analysis and discussion of the evaluation results are also presented to address the limitations and strengths as well as the recommendations for future improvements.

Keywords*:* *Short text, tag recommender, Arabic Language, Wikipedia, Latent Semantic Analysis, Spark*

الملخص

تتيح المواقع الاجتماعية للمستخدمين مشاركة المواد كالنصوص والصور، وتتيح حرية إضافة كلمات رئيسية لها تسمى أوسمة. ولكنَّ الحرية لها مساوئ منها: التكرار الناتج عن عدم ضبط الكلمات، الغموض، التشتت، الأخطاء الإملائية، والتفرّد، مما يعيق عمليات تنظيم واسترجاع البيانات في هذه الأنظمة.

نهدف في هذا العمل إلى عرض نظام اقتراح أوسمة للنصوص العربية القصيرة بالاستفادة من الويكيبيديا العربية كمصدرٍ للمعلومات، بحيث يتم توظيفُ تحليلِ الدلالاتِ الكامنةِ لاكتشافِ التشابه بين النص القصير ومقالات الويكيبيديا. وقد استخدم "أباتشي سبارك" للتعامل مع الحجم الضخم لمحتويات الويكيبيديا والعمليات الحسابية المعقدة لتحليل الدلالات الكامنة المستخدم لتحليل محتوى مقالات الويكيبيديا إلى ثلاث مصفوفات، وعند إدخال نص عربي قصير، يقوم النظام بمقارنته مع محتوى المقالات ويعطي كل مقالةٍ وزناً حسب علاقتها وتشابهها مع النص المدخل، ثم يتم اختيار الأوسمة المرشحة من عناوين وتصنيفات المقالات الأكثر شبهًا بالنص.

تم تقييم النظام المقترح اعتماداً على مجموعة من 100 نص قصير تم جمعها من موقع تويترفي ثلاث مجالات مختلفة و قام خبيران في كل مجال بتقييم الأوسمة التي أنتجها النظام. وقد حقق النظام المقترح 84.39% mean average precision و 96.53% mean reciprocal rank، مما يظهر مناسبة النظام ودقته لتوسيم النصوص العربية في حين أنه يواجه صعوبات تتعلق باللغة العربية وبتكرارات الكلمات النادرة. كما تم عرض تحليلٍ دقيقٍ ومناقشةٍ لنتائج التقييم تتناول نقاط القوة والقصور في النظام إضافةً إلى توصياتٍ لتطوير العمل مستقبلاً.

كلمات مفتاحية: *نصوص قصيرة، اقتراح أوسمة، اللغة العربية، ويكيبيديا، تحليل الدلالات الكامنة، سبارك*

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

قَالُواْ سُبۡحَٰنَكَ لَا عِلۡمَ لَنَآ إِلَّا مَا عَلَّمۡتَنَآۖ إِنَّكَ أَنتَ ٱلۡعَلِيمُ ٱلۡحَكِيمُ ٣٢

سُورَةُ البَقَرَة (32)

Dedication

To my dear mother and father who have given me all their love and support over the years, and for their unwavering commitment through good times and hard times.

To my wonderful, brilliant and supportive wife, **Niveen**, for her patience, forbearance and sustenance through my studying and preparing of this thesis. To my elegant sons Sary and Tameem, and my sweet daughter Yumna, whom I do all of this for them.

To my brothers and sister for their love and care.

To the spirit of martyr, my brother Sary.

To my father-in-law and mother-in-low for their encouragement and believing in me.

To my best friends Ashraf Qahman and Murad abu Jarad for their support and encouragement.

To all my friends and colleagues who supported me.

Acknowledgement

At the very outset, my thankfulness are to Allah the almighty who provided me with the needed strength to successfully accomplish this work, and to be surrounded by great and helpful people.

I would like to express my deepest gratitude to my advisor, **Dr. Iyad Mohammed Al Agha**, for his constant guidance, challenging discussions and advices, enthusiasm, and knowledge. He motivated me to think more deeply about my work. He also made great effort to build the structure and refine every detail of my work. I am grateful to him for working with me. I learned so much, it has been an honor. My Allah reword him on my behalf.

My everlasting gratitude to my parents who encouraged me to be the best I can be and to have high expectations and for their continuous prayer for the sake of my success.

Special thanks to my loving wife and children, who have been a constant source of support and encouragement during the challenges of graduate and life. Words cannot express how grateful I am to have you in my life.

Also, I am thankful for my whole family and friends for encouragement and support.

My sincere thanks also go to the department of information technology for facilitating needed means to accomplish this work.

Table of Contents

DECLARATION II

ABSTRACT III

الملخص IV

EPIGRAPH PAGE V

DEDICATION VI

ACKNOWLEDGEMENT VII

LIST OF TABLES X

LIST OF FIGURES XI

LIST OF ABBREVIATIONS XII

CHAPTER 1 INTRODUCTION 1

1.1 INTRODUCTION 1
1.2 STATEMENT OF THE PROBLEM 4
1.3 OBJECTIVES 4
1.3.1 Main Objective 5
1.3.2 Specific objectives 5
1.4 IMPORTANCE OF RESEARCH 5
1.5 SCOPE AND LIMITATIONS OF THE PROJECT 6
1.5.1 Scope: 6
1.5.2 Limitations: 6
1.6 RESEARCH CONTRIBUTION 7
1.7 STRUCTURE OF THESIS 7

CHAPTER 2 LITERATURE REVIEW 10

2.1 STATE OF THE ART 10
2.2 BACKGROUND 11
2.2.1 Apache Spark 11
2.2.2 Latent Semantic Analysis (LSA) 12
2.2.3 Arabic Wikipedia 17
2.3 RELATED WORKS 18
2.3.1 Short text tagging: 18
2.3.2 Text tagging: 21
2.3.3 Tagging with LSA 21

CHAPTER 3 METHODOLOGY 24

3.1 INTRODUCTION 24
3.2 CONFIGURING ARABIC WIKIPEDIA 24
3.2.1 Parsing and information extraction from Arabic Wikipedia XML Dump 25
3.2.2 Text Preprocessing 26
3.3 TAG RECOMMENDATION SYSTEM 30
3.3.1 Computing the Tf-idfs 30
3.3.2 Vectorization 31
3.3.3 Singular Value Decomposition 31
3.4 TAG SELECTION 33

3.4.1 Text Preprocessing 34
3.4.2 Vectorization 35
3.4.3 Selecting the top N Similar articles 35
3.4.4 Selecting Tags 36
3.5 CASE STUDY.....39
3.6 TOOLS.....42
3.7 SUMMARY.....43

CHAPTER 4 RESULTS AND DISCUSSION 46

4.1 INTRODUCTION.....46
4.2 DATASET.....46
4.3 EXPERIMENT SETTINGS.....47
4.4 EVALUATION PROCESS.....48
4.4.1 Experiment 1: Determining the top N articles 48
4.4.2 Experiment 2: evaluation of the system 52
4.5 EVALUATION METRICS.....53
4.6 RESULTS AND DISCUSSION.....55
4.7 SUMMARY.....61

CHAPTER 5 CONCLUSIONS 63

REFERENCES 67

List of Tables

Table (2.1): Term occurrences in documents 13
Table (2.2): Terms and documents in a concept..... 15
Table (3.1): Information about the downloaded dump and the contained information..... 25
Table (3.2): Deleted texts and terms 26
Table (3.3): Results of Arabic stemmers' comparison..... 27
Table (3.4): Precision and time efficiency for NLP tools 28
Table (3.5): Statistics about the knowledge base 30
Table (3.6): Categories intersections of the similar articles 38
Table (3.7): The top 7 articles for the tweet 41
Table (3.8): Categories selected by the system 41
Table (3.9): Titles selected by the system 42
Table (3.10): Suggested tags for the tweet 42
Table (4.1): A snapshot of the gathered dataset 46
Table (4.2): Master node specifications..... 47
Table (4.3): Worker nodes specifications 48
Table (4.4): Correct and incorrect tags of a tweet 49
Table (4.5): Result of experiment on 10 tweets with different number of top articles 50
Table (4.6): New tags at 18 top articles for example in Table (4.4) 52
Table (4.7): Tags of a tweet evaluated by experts 53
Table (4.8): A short text, resulted tags, expert evaluation and measures calculations..... 55
Table (4.9): Evaluation metrics of the system 55
Table (4.10): Results across different subjects..... 56

List of Figures

Figure (1.1): The system described in simple steps 3
Figure (2.1): The form of the singular-value decomposition 14
Figure (3.1): The tag recommender system 24
Figure (3.2): Result of SVD for a 5 documents 7 terms matrix 32
Figure (3.3): calculating the cosine similarity 33
Figure (3.4): Preprocessing of the short text 39
Figure (3.5): Short text as a vector 40
Figure (4.1): Results for 10 tweets on different number of top articles 51
Figure (4.2): $AP_{(1\text{-}100)}@k_{(1\text{-}10)}$ 57

List of Abbreviations

API	Application Programming Interface
LSA	Latent Semantic Analysis
MAP	Mean Average Precision
MRR	Mean Reciprocal Rank
NLP	Natural Language Processing
P@K	Precision @ position K
PoS	Part-of-Speech
RDD	Resilient Distributed Dataset
SVD	Singular Value Decomposition
Tf-idf	Term Frequency Inverse Document Frequency
URL	Unified Resource Locator
XML	Extensible Markup Language

Chapter 1
Introduction

Chapter 1
Introduction

1.1 Introduction

With the massive daily increase of data on the internet, especially text, automatic tagging recommendation that detects and adds informative, and descriptive tags to documents becomes an important necessity for information aggregation and sharing services(Oliveira et al., 2012).

Tagging is the practice of creating and managing labels called tags that categorize or describe the content using simple keywords. It's not a new concept. Journals, conference proceedings, and even dissertations have required keywords from authors to improve their information retrieval performances for years. (Jeong, 2009). Tagging is considered as the way to organize the stuff you don't have time to organize(Fallows, 2007).

Social activities on Twitter, Facebook, Flicker, personal blogs etc. are becoming very popular among users who want to share local or global news, their knowledge or opinions (Kywe, Hoang, Lim, & Zhu, 2012). Lately, users are also using these services to search for information. Therefore, some services include tag or category information to better facilitate search. However, these tags are typically free-form in nature with users permitted to adopt their own conventions and interests without restriction, which can make the set of tags noisy and sparse. Moreover, many works have addressed tagging documents, whereas short texts are peculiar regarding length, composition and formality(Garcia Esparza, O'Mahony, & Smyth, 2010).

A solution to the above problem is to recommend tags (Garcia Esparza et al., 2010) or categorizations to users to enrich and clarify the content, facilitate retrieval, and perform less cognitive effort. Which, in one hand, if done properly, will improve text retrieval, linking, classification, clustering, recommendation, simplify archiving, and also will give the user or the application insight to the content and facilitate seeing the data (information) from different dimensions and enrich the context of the tagged text. On the other hand manual tags or metadata creation is costly in terms of time and

effort and users are unwilling to provide an adequate number of tags which is called tag sparsity.

Many works have addressed the tag recommendation problem, but the special characteristics of short texts has made the tag recommendation a new and even more challenging dilemma. It is statistically shown that social texts are extremely short, poorly composed, and tend to be more informal (Guo, Li, Ji, & Diab, 2013). So the application of conventional statistical techniques becomes impractical due to these special characteristics.

When we search for a text, what we really want is to look for the meaning behind the words of the text not the exact terms. Latent Semantic Analysis (LSA) has the ability over other techniques to discover these meanings depending on a powerful linear algebra technique called the Singular Value Decomposition (SVD) (Ryza, Laserson, Owen, & Wills, 2015). SVD can describe the intensities of relations between the components of an input matrix, e.g. Documents and terms, which reveals different relations between the components, such as the relation from: term to term, term to document or document to document (Turney, 2001). This property gives LSA the advantage over techniques like Natural language processing (NLP) (Guo et al., 2013; Laclavik, Šeleng, Ciglan, & Hluchý, 2012) or machine learning techniques (Allahyari & Kochut, 2016a; Tang, Hong, Li, & Liang, 2006) that lack semantics, because it goes deeper than comparing terms, to comparing the meanings behind these terms (Ryza et al., 2015).

LSA was used on data sets other than the Arabic Wikipedia, since Arabic language may pose additional problems because few (or less reliable) resources are available to extract the needed data from the text. While the Arabic Wikipedia is recently used in fields other than tagging, this filed remains unexplored especially for short texts.

Our work aims to recommend tags for short Arabic text, e.g. tweets, depending on Arabic Wikipedia Articles and categories, in an effort to select proper tags such as the title and the categories of the articles that are pertinent to the text by utilizing LSA and dimensionality reduction heavy computations. In order to do that, we need to

handle a massive collection of data (Arabic Wikipedia) which contains over a million Articles and a seven million terms, that no single accessible computer we have can deal with, leading to our need to use Apache Spark cluster (Zaharia, Chowdhury, Franklin, Shenker, & Stoica, 2010).

The choice of Arabic Wikipedia as a source of tags is motivated by its large coverage of different knowledge areas, a thing that makes it adequate for recommending tags in any domain of knowledge. Given an Arabic short text, the system suggests ranked tags to that text. These tags are selected from the titles and categories of the Arabic Wikipedia. (Figure 1.1) presents the system as simple steps, details will be discussed later in Chapter 3.

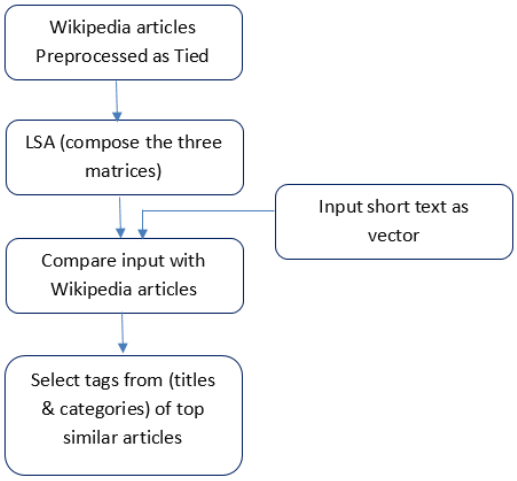

Figure (1.1): The system described in simple steps

First the system constructs the term document matrix by employing the term frequency-inverse document frequency (Tf-idf) weighting schema on the body of the articles after segmentation and lemmatization. Then the latent semantic analysis LSA is applied on that matrix by performing the singular value decomposition. This step allows the system to discover hidden semantics between the input short text and the Wikipedia articles by calculating cosine similarity. Tags are selected from the titles and categories of the articles that are most similar to the short text. Furthermore, the selected tags are ranked in order to present the best tags first.

As far as we aware of, this is the first effort that aims to offer tag suggestion of Arabic text using Wikipedia. While the English version of Wikipedia has been widely utilized in several research areas related to information retrieval and Natural Language Processing. Not all researchers and developers have the computational resources to process such a volume of information and there has been little efforts to utilize the Arabic Wikipedia for similar research. The proposed system is expected to act as a baseline for the research tackling Wikipedia-based tagging of Arabic text.

The tag recommender was assessed over a dataset of 100 short texts gathered randomly from Twitter in three domains: Sports, Technology, and News. The tags generated by the system where examined and judged by two human experts in each field. Our recommender achieved (**84.38%**) mean average precision and (**96.53%**) mean reciprocal rank.

1.2 Statement of the problem

The main problem addressed by this research is how to recommend semantically related tags to Arabic short text by exploiting Arabic Wikipedia. No effort, to our knowledge, has explored the use of Arabic version of Wikipedia for tagging Arabic texts.

Besides, tags generated by existing techniques mostly relied on statistical approaches while they lacked semantics. They were also restricted to English Language or were applicable on long documents only. In addition, many of existing approaches were domain specific, had limited coverage of knowledge areas, and did not often suit extremely short, poorly composed, and informal short texts.

1.3 Objectives

In this section, we present both main and specific objectives of the research work.

1.3.1 Main Objective

The main objective of this research is to design and implement an automatic semantic tag recommender for short Arabic texts that is accurate and reliable, by exploiting the Arabic Wikipedia.

1.3.2 Specific objectives

The specific objectives of the proposal are:

1. Explore how the massive content of the Wikipedia can be processed effectively.
2. Explore the best processing and NLP techniques for Arabic language Lemmatizing and segmenting, compare them, and select the most suitable to our work in order to access, preprocess, clean and filter the content of Arabic Wikipedia.
3. Investigate the implementation of LSA and how to identify most relevant and similar documents.
4. Provide a novel technique for tagging Arabic short texts from the titles and categories of the relevant Wikipedia articles.
5. Assess the performance of our system by annotating short texts obtained from social networks (Twitter). The performance will be evaluated by a number of experts in different fields and evaluation metrics.

1.4 Importance of Research

1. Recommend semantically related tags for Arabic short texts which give insight and enrich the text. Since tags are becoming more significant to improve search and text retrieval, simplify archiving, linking, classification, clustering, recommendation, and provide consistency among users.
2. Due to the scarcity of works that are oriented towards Arabic language in the field of automatic tag recommendation, this work could advance first step in the field of Arabic tag recommendation. While our technique still general but the test is limited to Arabic short text.
3. Extend the coverage of our tagger by exploiting Arabic Wikipedia with its massive content as a background knowledge. This will provide a system that is more general than domain specific taggers.

1.5 Scope and limitations of the project

1.5.1 Scope:

- This work utilizes only the Arabic Wikipedia.
- Our work is limited to short Arabic texts. But the process is easily applicable for any language.
- Our technique considered standard Arabic language as well as non-standard Arabic language texts published by common people.
- The evaluation of the system was done using a specific dataset gathered from posts on twitter in the fields of Sports, Technology, and News, similar to the fields of our experts. It was not possible to conduct a comparative study due to the lack of similar tagging approaches of Arabic text
- Apache Spark was used as parallel framework to process the content of Wikipedia and build the LSA based system.

1.5.2 Limitations:

1. Low efficiency of the existing Arabic segmenters and stemmers affects the quality of results.
2. Some of the Arabic Wikipedia pages have misspellings and incomplete content.
3. Tweets used for testing contain words of daily dialect (slang), and misspellings, which have a negative influence on the results.
4. Non-Arabic names sometimes are written differently in Arabic (e.g. people, places, scientific experiments, compounds) which affect the quality and accuracy of the results. Also, the system excludes terms written in Latin characters.
5. The terms of input short text that are not found in Wikipedia was excluded from the short text.
6. Comparing a short text with a long one could increase the computation on the system.

1.6 Research contribution

The work in this thesis has the following research contributions:

1. A comparison was conducted between some NLP for Arabic language to select the best one based on the suitability of outcome for our work and regardless of the execution time.
2. Implement (LSA) on the whole Arabic Wikipedia. Because, as we recall, LSA is used mostly to tackle the English version not the Arabic version.
3. Present a novel system that we can consider it as a guideline for the future efforts in utilizing Arabic Wikipedia structure in real life applications.
4. It proposes an in-depth evaluation of our tagging system and explored the potential shortcomings and strengths. This detailed evaluation can inform Arab research community with the various design options, challenges and recommendations when designing similar approaches.
5. This is the first work, as far as we know, that explores the tagging of short Arabic text by exploiting Arabic Wikipedia content and LSA. Arabic Wikipedia has been exploited recently by the Arab researchers and few efforts have tried to interface to the Arabic version of Wikipedia for different purposes distant from tagging.
6. Generate a standard dataset for Arabic short-texts and tags.

1.7 Structure of Thesis

The thesis consists of five chapters. The chapters are organized in general as follows:

Chapter 1: Introduction: this chapter is an overview of the problem, work done in the field, and focuses on the proposed solution. It also discusses the challenges and difficulties of using Arabic text and Arabic Wikipedia.

Chapter 2: Literature Review: this chapter focuses on related works that employed Wikipedia or LSA as well as the works on the tagging field.

Chapter 3: Methodology: This chapter explains the detailed steps of the tagging system. And present a scenario of the system and the results of each phase.

Chapter 4: Results and Discussion: this chapter explains the assessing process of our system, test dataset, evaluation metrics, and discusses the results focusing on the sources of strengths and weaknesses.

Chapter 5: Conclusions: this chapter presents a conclusion of the thesis and possible future works.

Chapter 2
Literature Review

Chapter 2
Literature Review

2.1 State of the Art

The world-wide-web has become the largest ever free-access information repository with billions of web pages (Abdeen & Tolba, 2010). With the massive daily increase of data, especially text, novel approaches are needed to mine such data efficiently and effectively. One way to improve efficiency is to provide proper tags. Some recent works employ tags in retrieval (Ionescu et al., 2015),clustering (Bernotas, Karklius, Laurutis, & Slotkienė, 2015), classification (Dafney & Mary, 2014) etc.

Plenty of state-of-the-art have addressed the issues of *tagging*(Allahyari & Kochut, 2016a; Garcia Esparza et al., 2010; Hassan, Karray, & Kamel, 2012; Otsuka, Wallace, & Chiu, 2014), *keywording* (HaCohen-Kerner, 2003; Hulth, 2003; Laclavik et al., 2012; O'Neil & Sangiovanni-Vincentelli, 2014; Tang et al., 2006; Tonella, Ricca, Pianta, & Girardi, 2003; Turney, 2000; Yih, Goodman, & Carvalho, 2006), and *summarizing text* (Gong & Liu, 2001; Yeh, Ke, Yang, & Meng, 2005), which all, one way or another, are aiming to acquire important and meaningful tags (words, phrases, or sentences) that describe the content and the soul of the text.

Our work aims to recommend tags for short Arabic text, e.g. tweets, depending on Arabic Wikipedia Articles and categories, in an effort to select proper tags such as the title and the categories of the articles that are pertinent to the text by utilizing LSA and dimensionality reduction heavy computations. In order to do that, we need to handle a massive collection of data (Arabic Wikipedia) which contains over a million Articles and a seven million terms, that no single accessible computer we have can deal with, leading to our need to use Apache Spark cluster (Zaharia et al., 2010).

The following section presents a brief background about Apache Spark, Latent semantic analysis, Singular Value Decomposition and Arabic Wikipedia.

2.2 Background

2.2.1 Apache Spark

Apache Spark is an open source big data processing framework built around speed, ease of use, and sophisticated analytics. It was originally developed in 2009 in UC Berkeley's AMPLab, and open sourced in 2010 as an Apache project (Zaharia et al., 2010).

We restrict our attention to Spark, because it provides a highly-optimized machine learning library called MLlib (Meng et al., 2016) which has several features that are particularly attractive for matrix computations (Bosagh Zadeh et al., 2016; Zadeh et al., 2015):

1. Resilient Distributed Datasets (RDDs) is essentially a distributed fault-tolerant vector that can perform operation as in local mode(Gittens et al., 2016).
2. RDDs allow user-defined data partitioning, and the execution engine can exploit this to co-partition RDDs.
3. And co-schedule tasks to avoid data movement.
4. Spark logs the history of operations used to build an RDD, enabling reconstruction of lost partitions upon failures.
5. Spark provides a high-level API in Java that can be easily extended. Which lead to creating a coherent API for matrix computations.

Hadoop (Zikopoulos, 2011) is another big data processing framework that is a software library and a framework which allows for distributed processing of large data sets (big data) across computer clusters using simple programming models. But Spark is favorable to us because (Spark, 2014) first: its ease of use compared to Hadoop and allows writing applications in Java and other languages. Second: Spark runs programs up to 100 times faster than Hadoop. Third: Spark powers a stack of libraries including MLib for machine learning which is essential to our work and also provid near real time analysis that is suitable for machine learning.

Many works have used Spark and MLib for data analysis purposes (Agnihotri, Mojarad, Lewkow, & Essa, 2016; Moss, Shaw, Piper, Hawthorne, & Kinsella, 2016),

stating the adequacy for processing terabytes/petabytes of data, which are commonplace in modern day society where both machines and humans generate petabytes of data every day.

2.2.2 Latent Semantic Analysis (LSA)

Latent Semantic Analysis, as the name indicates is the analysis of hidden semantics in a corpora of text. Any collection of documents can be represented as a huge term-document matrix and other things like how close two documents are, how close a document is to a query etc. can be deduced by cosine similarity. However, such models have two drawbacks that are common in many languages: polysemy and synonymy (Deerwester, Dumais, Furnas, Landauer, & Harshman, 1990) where polysemy is a word that have different meanings in different contexts and synonymy is a concept having multiple forms of representation i.e. two or more words denoting the same concept.

LSA transforms the original data into a different space so that two (or more) documents/words about the same concept are grouped together (so that they are most similar to each other). LSA achieves this by Singular Value Decomposition (SVD) of term-document matrix.

2.2.2.1 How Latent Semantic Analysis Works

When we try to find relevant document to search words, the problem arose because what we really want is to compare the meanings or concepts behind the words. LSA attempts to solve this problem by mapping both words and document into a concept space and doing comparisons in that space (Deerwester et al., 1990).

In order to make this problem solvable, LSA introduces some dramatic simplifications.

1. Documents are represented as "bags of words", where the order of the words in a document, sentence structure, and negation are not important, only the number of the word occurrences in the document matters.

2. Concepts are represented as patterns of words that usually appear together in documents. For example "حريق", "اندلع", and "إطفاء" (fire, flare, and firefighting, respectively) might usually appear in documents about "حريق" (Conflagration).
3. Words are assumed to have only one meaning. This is clearly not the case ("جدول" could be a table "صفوف وأعمدة", schedule "الفعل: جدول" or a spring "ينبوع") but it makes the problem tractable.

To build the term-document matrix words are usually pre-processed by means of tokenization, stop-words removal and stemming (Sarwar, Karypis, Konstan, & Riedl, 2001). Then each token is assigned a weight which is proportional to its frequency normalized using various schemes, the most known is the Term frequency-Inverse Document Frequency Tf-idf scheme (Han, Pei, & Kamber, 2011) ,where

$$w_{t,d} = \log_{10}(1+tf_{t,d}) \times (1+\log_{10}(N/df_t)) \qquad \textbf{(2.1)}$$

Tf-idf is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. In this matrix each column represents a document and each row in the column represents a term frequency in that document. We apply Tf-idf weighting because it negates the effect of high frequency words in determining the importance of a document. And we use log to the base 10, to diminish the values of the results, since we are dealing with huge number of documents and terms. As a simple example we present (Table 2.1) below, which shows each term occurrences in every document that we depend on in calculating the Tf-Idf for each term-document.

Table (2.1): Term occurrences in documents

	D1	D2	D3	D4	D5
t1	1	0	3	0	0
t2	1	1	0	0	0
t3	0	1	0	3	1
t4	1	1	0	1	0
t5	0	0	0	0	2

For example to calculate the Tf-Idf for the term t1 in the document D3:

First: $Tf_{t1,d3} = \log_{10}(1+tf_{t1,d3})$ = log(1+ occurrences of t1 in D3) = log (1+3) = 0.6

Second: Idf $_{t1,d3}$ = $1+\log_{10}(N/\mathrm{df}_t)$ = log(No of all documents/No of documents that contain t1)
.

= 1+log(5/2) = 1.398

Finally: Tf-Idf $_{t1,d3}$ = 0.6 * 1.398= 0.8388

And this is performed for every term in each document.

In LSA, matrix approximation performed by singular value decomposition that can relate documents and terms into concepts. Documents and terms in each concept are all semantically related which make it superior to frequency based approaches. SVD effectively "splits" a term-document matrix M(m × n) into three new matrices, U, S, and V(Ryza et al., 2015).(Figure 2.1) shows the SVD form. Where m is the number of documents and n is the number of terms.

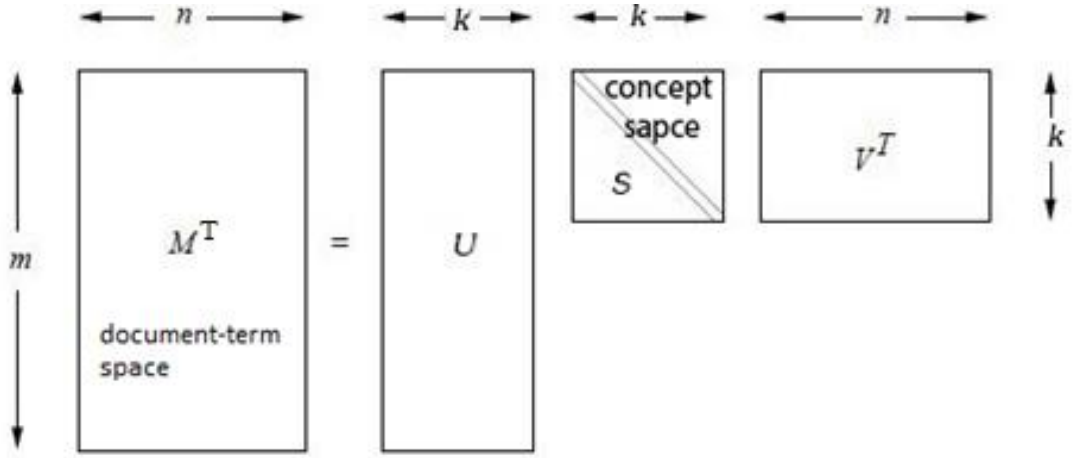

Figure (2.1): The form of the singular-value decomposition

A numerical example of SVD and dimensionality reduction is introduced below.

Example: Let M be a (5 documents) × matrix (7 terms), which has the shown values. The number reflects term counts in documents for simplicity. We need to perform the SVD on the matrix, then perform the dimensionality reduction setting k=2, where 2 in the number of concepts to map the documents into.

M=

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 0 & 0 & 2 & 2 \end{bmatrix}$$

The Result after performing SVD and dimensionality reduction with k=2 will be as below:

$$\underbrace{\begin{bmatrix} 1 & 3 & 4 & 5 & 0 & 0 & 0 \\ 1 & 3 & 4 & 5 & 0 & 0 & 0 \\ 1 & 3 & 4 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 5 & 2 \\ 0 & 0 & 0 & 0 & 4 & 5 & 2 \end{bmatrix}}_{M^{\mathrm{T}}} = \underbrace{\begin{bmatrix} .58 & 0 \\ .58 & 0 \\ .58 & 0 \\ 0 & .71 \\ 0 & .71 \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} 12.4 & 0 \\ 0 & 9.5 \end{bmatrix}}_{S} \underbrace{\begin{bmatrix} .14 & .42 & .56 & .7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & .6 & .75 & .3 \end{bmatrix}}_{V^{\mathrm{T}}}$$

Where the shaded values in U represent the documents related to the shaded concept in S, and the terms related to the same concept are the ones shaded in V^T.

An example to the terms and documents that can be found in a concept are shown in (Table 2.2).

Table (2.2): Terms and documents in a concept

Documents	Terms
Lepidoptera حرشفيات الأجنحة	family عائلة
Orchid زهرة الأوركيد	Orchidaceae سحلبية
Crustaceans قشريات	beetle خنفساء
biology علم الأحياء	zone منطقة
Insects مملكة الحشرات	moth عث
species-typica نوع نمطي	hawkmoth فراشة
	species أجناس
	genus نوع

We notice that the presented documents in the concept have a thematic coherence with each other and with the terms related to the same concept. And also the terms are semantically related to each other.

2.2.2.2 Querying and scoring with the low dimensional representation

The Tf-idf composed matrix presents a shallow knowledge about the relationship between entries, depending on the simple frequency count. LSA has the ability to base scores (similarities) on a deeper understanding of the corpus. For example: if the term Samsung (سامسونج) appears in the article of smartphone (هاتف ذكي), which frequently mentions Apple (أبل), the LSA representation may be able to recover the relation between Samsung and Apple based on the co-occurrence of them in other documents.

Now, consider the task of finding the most relevant document to a particular document. The naïve approach requires computing the dot product between the row vector of the document, and every other row vector in the term-document matrix. Where the number of multiplications is proportional to the number of documents times the number of terms. LSA can achieve this by a number of multiplication proportional to the number of document times the number of concepts. So rather than calculating the similarities on the low rank matrix (Tf-idf matrix), some linear algebra manipulations show that the cosine similarity between two rows in the reconstructed matrix is exactly equal to the cosine similarity between the corresponding rows in US matrix. Finding the cosine similarity between the document and all other documents is equivalent to multiplying US to the corresponding row resulting in $(US)u_d$.

What about new documents? Simply, the same. But, instead of finding the row of the document in the matrix, we need to create it. It can be done by setting the value of each term in the query (the new short text) to its inverse document frequency to maintain the weighting scheme used in the original term-document matrix(Ryza et al., 2015). Before the comparison and after forming the short text vector, it is multiplied by the matrix V^T to compute the concept space vector of the short text.

2.2.3 Arabic Wikipedia

Wikipedia in general has been adopted in many works, specially, text processing. Works in the field of this thesis (Allahyari & Kochut, 2016a; Hassan et al., 2012; Mei & Zhang, 2008; Schönhofen, 2009; Singhal & Srivastava, 2013) and other fields (Gabrilovich & Markovitch, 2006; Shapira, Ofek, & Makarenkov, 2015; Wang, Hu, Zeng, & Chen, 2009) have used Wikipedia as a training data or test data.

Wikipedia is currently the most popular free-content, online encyclopedia, which surpasses in scope many conventional encyclopedias and provides a cornucopia of world knowledge(Gabrilovich & Markovitch, 2006). Arabic Wikipedia is one of the popular Wikipedia projects, to date it is ranked 19th. It contains 1,238,570 pages with 435672 actual articles and 267580 categories with average 10 edits each. Also, it has a base of about 1,288,144 registered users and written collaboratively by largely anonymous internet volunteers. There are about 4,438 active contributors working on the articles(Wikipedia, 2016). Thus the knowledge presented in the articles over Wikipedia in general are convinced upon by editors of similar interest. It covers most of the technical and non-technical topics, events that have happened, topics related to most of the domain areas (Ramudu & Murty, 2012).

It is essential to note that we are not only using Arabic Wikipedia to simply increase the amount of the data. Rather, we use the knowledge distilled from the encyclopedia to enrich the representation of tags, by better matching the short text to the articles. Since we believe that Arabic Wikipedia has several advantages over other Arabic corpora:

First: its articles are much cleaner, mostly qualify as standard written Arabic, heavily revised and edited. *Second*: the categories assigned to an article cover the perspectives and interests of large number of editors. *Third*: categories and articles (content and title) are continually updated and checked. *Forth*: High coverage for many domains, including medicine, News, Sports, Technology, etc. *Finally*: Arabic Wikipedia represents massive amounts of world knowledge (Milicevic, Nanopoulos, & Ivanovic, 2010).

Although Arabic Wikipedia structure is fairly shallow, and we propose to treat Arabic Wikipedia categories as having essentially no hierarchy. This way, mapping documents to relevant Wikipedia concepts yields truly better tag selection.

2.3 Related Works

Recently, automatic semantic tagging and annotation of documents have attracted a great deal of attention, since it can add significant benefits to many text mining tasks(Allahyari & Kochut, 2016a) , as information retrieval(Shapira et al., 2015), and text classification(Wang et al., 2009), text clustering and cluster labeling (Tonella et al., 2003) although, many attempts have been conducted to address this issue. In the field of our work, several efforts employed different techniques and knowledge bases, some of them targeted documents, and others targeted short texts. In the following sections we review short and long text tagging in association with the works that applied LSA in their approaches.

2.3.1 Short text tagging:

Several previous studies have addressed the problem of tagging of short text such as social snippets(Li, Zhou, Juan, & Han, 2010; Singhal & Srivastava, 2013) and abstracts of research papers (Bhowmik, 2008; HaCohen-Kerner, 2003; Hulth, 2003), topics (Bhowmik, 2008; HaCohen-Kerner, 2003), and micro-blog posts (Garcia Esparza et al., 2010; Kywe et al., 2012; Otsuka et al., 2014).

Depending on the title and the abstract of scientific papers, Bhowmik (Bhowmik, 2008) utilized a set of keywords that are pre-weighted, to weight and extract keywords and sentences according to their importance and position. His work is domain specific, and depends on a set of keywords that needs to be updated. Also it cannot enrich very short texts. likewise, Hulth (Hulth, 2003) built a supervised rule induction classifier that uses the abstract of the paper to generate tags, before she added linguistics knowledge to the representation, therefore each word has Part-of-Speech as a new feature that improved the results. In addition, HaCohen-Kerner (HaCohen-Kerner, 2003) used the frequency of words and phrases to create a weight matrix from abstracts then sorted these weights and chose the highest as tags. All previous works

consider only the occurrences of the words, and the resulting tags are included in the original text and may lack semantics, but in our work we consider semantic relations and the generated tags mostly are not contained in the original text.

Singhal and Srivastava (Singhal & Srivastava, 2013) proposed a technique for automatically tagging documents by concepts and named entities using only "short text" information from the documents, such as a document title, or a news article head line. In their work they employ the knowledge bases of Wikipedia, DBpedia, Freebase and Yago to generate semantically relevant tags for the document. They used a search engine to enrich the text with author name, snippets and/or URL. Then find word frequencies in the snippets. After that all short texts are clustered, pruned, and finally the remaining concepts and named entities are returned as tags. This work has a number of drawbacks. One is that it needs a collection of short texts to perform clustering which have to be pre-prepared so the model may not handle the variety of the new entries. The other is its need to use a search engine which may provide shallow or wrong information. Otherwise, the search engine results may depend on the whole document which converts the assumption of short text tagging.

Li and others (Li et al., 2010) worked on social snippets. First they calculated a set of features for each word such as Tf-Idf, PoS, position in the text, text length, etc. They trained a classification model based on the labeled keywords of social snippets. And finally the keyword candidates with highest scores through the classification model are returned. But the training data in this model is manually prepared to meet the experiment, indication insufficiency for new snippets, and may generate redundant tags.

Based on the output of a topic model that was run on a collection of short documents, a framework for topical keyphrase generation and ranking was proposed by O'Neil and Sangiovanni (O'Neil & Sangiovanni-Vincentelli, 2014). By means of clustering the words of the short texts into topics using Latent Dirichlet Allocation, the authors were able to generate and rank candidate keyphrases according to word topic assignment. The system has high performance. However, they need multiple short

texts as input, and topics have to be informative for good clustering results indicating the insufficiency to handle very short texts, and in ability to handle new entries.

Other works attempted to model users' interests based on their historical tagging behaviors, and recommend tags to the user from other similar users (Bogers & Van den Bosch, 2008; Golder & Huberman, 2006). In Bogers and Van work (Bogers & Van den Bosch, 2008) the social reference management website CiteULike was used for recommending scientific articles to users, based on their reference library. Their work depends mainly on collaborative filtering algorithm, and uses a relatively small collection of documents. Golder and Huberman (Golder & Huberman, 2006) presented a dynamical model of collaborative tagging that predicts stable patterns in user activity and tag frequencies then relates them to recommendations and shared knowledge, both of the above works are user-centered while we focus on documents, and they are affected by the user's perspective and interests.

Several attempts have addressed micro-blogs posts tagging. for example Otsuka and others (Otsuka et al., 2014) rely on compiling a large number of tweets to construct Tf-Idf matrix, that allows to measure the similarity between tweets, and recommend tags that are associated with the most similar tweets. also Esparza and his colleagues (Garcia Esparza et al., 2010) aim to categorize and recommend tags for tweets and other short messages in order to meet the different tagging conventions of users and to facilitate search. They used Tf-idf term weighting and a kNN classifier with k =1. Tfidf is considered naïve compared to LSA in a way that results in an undesirable matches and lack semantics. While Kywe and authors (Kywe et al., 2012) consider both user preferences and tweet content in selecting hashtags to be recommended. The system depends mainly on collaborative filtering and their method recommends hashtags found in the previous month's data which is biased by the user concerns and is inefficient for suggesting new tags. Also, Mei and Zhang (Mei & Zhang, 2008) recommends tags for short text utilizing highly weighted words and titles of Wikipedia articles, performing all the work using a probabilistic model. Despite being similar to our work, the last mentioned techniques use old tweets to tag new tweets, while we use revised, rectified, and widely sparse Arabic Wikipedia documents.

2.3.2 Text tagging:

In recent time, several attempts have been made to annotate documents and web pages, for example; Tang et al.(Tang et al., 2006) were concerned of semantic annotation on hierarchically dependent data, where targeted instances can have hierarchical dependencies with each other. Ontea (Laclavik et al., 2012) is a platform for automated semantic annotation or semantic tagging, its implementation based on regular expression patterns was presented while the test was carried out on job offers as documents with evaluation of results. Both of the above works use linguistic techniques to address annotation of the documents, and differ from our work in a way that they are primarily focused on specific entities mentioned in the documents, whereas we take all the words in consideration.

Other works similar to ours include Schönhofen's (Schönhofen, 2009) where he used Wikipedia articles titles and categories to tag documents. In his method, he first finds all the Wikipedia articles related to a document by matching their titles with the words of the document. Then, they select categories assigned to these articles and rank them, and finally choose the categories with the highest weights as the topics of the document. Our work is not restricted to titles and categories, but exploits the whole content Wikipedia articles in LSA to determine articles related to short Arabic text. Also Hassan and others (Hassan et al., 2012) used Wikipedia text and hierarchical ontology to tag documents by constructing a category term matrix C, and then term-document matrix D for the document. They eventually, find document-category similarity $S=DC^T$. Allahyari and Kochut (Allahyari & Kochut, 2016b) as well, used a probabilistic model. The authors incorporate DBpedia knowledge into the topic model for tagging web pages and online documents. Our work is similar to both Allahyari's and Hassan's (Allahyari & Kochut, 2016b; Hassan et al., 2012) in terms of using Wikipedia, but ours explores the use of Arabic Wikipedia instead, while our technique remains general.

2.3.3 Tagging with LSA

All the mentioned works are similar to ours in terms of the objective of text tagging, other works are similar in technique, where we employ LSA to generate

features before matching new documents. LSA was used by Symeonidis et al. (Symeonidis, Nanopoulos, & Manolopoulos, 2010) and utilized to select tags for biomedical abstracts by finding similar documents in the MEDLINE database. The system then uses a ranking schema to select candidate tags drawn from the most similar documents. While this work is domain specific as it is restricted to 2000 abstracts from the MEDLINE database, our work is generic as it builds the LSA based system from the whole content of Wikipedia, and employs parallelization to handle the huge size of data.

Gong and Liu(Gong & Liu, 2001) performed SVD on m×n term-sentence matrix (m: number of terms ≥ n: number of sentences where each column represents a document and each row in the column represents a sentence frequency in that document). They used a couple hundreds of CNN news in order to obtain the singular value matrix S, and the right singular vector matrix V^T, then select the k^{th} right singular vector from matrix V^T. And finally, select the sentence which has the largest index value with the k^{th} right singular vector, and include it in the summary. Likewise, the term-sentence matrix was used by Yeh and others (Yeh et al., 2005) accompanied with modified corpus-based approach to select the best sentences that summarize one hundred political articles from New Taiwan Weekly. Both works are analogous to ours, except we construct a term-document matrix instead of term-sentence matrix. Also we tag with titles and categories, and deal with enormous number of documents whereas they use hundreds. Finally, we use short texts instead of long documents.

The algorithm proposed by Symeonidis et al. (Symeonidis, Nanopoulos, & Manolopoulos, 2008) performed latent semantic analysis and dimensionality reduction using the higher order singular value decomposition technique. This algorithm was tested on two data sets from Last.fm and BibSonomy. They stated the results showed substantial improvements in terms of effectiveness measured through recall.

All the works that exploited LSA have been used to tag a document using other documents in the same corpus, while in our work we use the Wikipedia as a corpus to tag new short texts that are not in the corpus.

Chapter 3
Methodology

Chapter 3
Methodology

3.1 Introduction

This chapter presents the system of a tag recommender system that utilizes Latent Semantic Analysis on the Arabic Wikipedia. It clarifies the detailed steps of the tagging System which include: configuring Arabic Wikipedia and preprocessing of the text. Second, computing Tf-idf and SVD dimensionality reduction. Third, preprocess the short text to be tagged. Forth, the tag selection procedure exploiting titles and categories of the articles. And finally, a case study is presented to view the functional steps of the tagging process.

3.2 Configuring Arabic Wikipedia

This section briefly explains the configuration needed for our tagging system. The description of the system is depicted in (Figure 3.1).

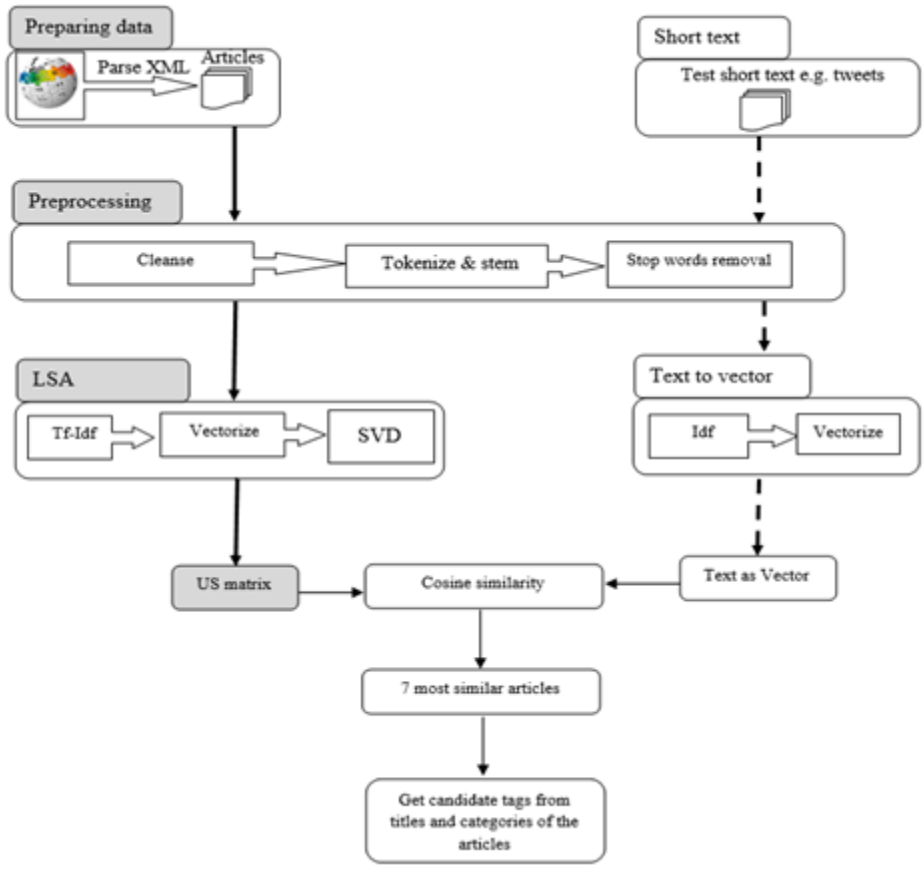

Figure (3.1): The tag recommender system

This configuration includes parsing and preprocessing of Arabic Wikipedia to enable fast information access and retrieval. Note that all the configuration settings are performed only once. (Figure 3.1) shows the complete system processes from preparation until tag selection. The solid arrows are for the system preparation, the dashed arrows are for the tagging process. Detailed description is provided below. Code, data set and results can be found at **https://github.com/YousefSamra/ShortTextTagging**

3.2.1 Parsing and information extraction from Arabic Wikipedia XML Dump

In this section we briefly explain the steps we have taken to gather the content that is essential to our work. We selected the most recent XML Dump file of the Arabic Wikipedia, 1st January 2017(Wikipedia, 2017), which contains a large number of revised, reviewed and verified articles. The Arabic Wikipedia contains 1,238,570 pages including 435,672 actual articles, 267,580 categories and has a hierarchical depth of 217. All this data is available in the XML dump file. After downloading the dump file. It was parsed to extract only the main content of Wikipedia articles. This content includes the article content, title, and associated categories, that are valuable information to our work, because we need to match the input short text to the articles body and need the categories and titles of Wikipedia articles to select tags. Other pages e.g. disambiguation, redirect, template, etc. are not needed in our work, so we neglected them. (Table 3.1) presents some information about the file, and the information it contains.

Table (3.1): Information about the downloaded dump and the contained information

XML Dump File Size	3.42 GB
Number of Categories	267580
Number of All Pages	1238570
Number of Redirect Pages	437726
Number of Disambiguation Pages	10473
Number of Template Pages	345759
Number of Discussion Pages	181
Number of Empty Body Pages	8756
No Category Pages	3
Articles needed for our work	435672

After removing all pages listed in the previous table, the relevant remaining 435672 articles that we used in our system were stored in a text file after being preprocessed, in order to be distributed among the working nodes of Spark cluster lately. All other pages were swiftly investigated for any miss enumerated ones, and there wasn't any.

3.2.2 Text Preprocessing

In order to better match the terms of the input text with the Arabic Wikipedia terms, it is important to perform some text preprocessing on both of them. The steps we undertook includes cleansing, tokenizing, stemming, and stop-word removal, performed only on the body of the articles, titles and categories remains untouched. As these steps are significant to our work, they are also tricky because it requires a lot of investigation and comparisons between some of the available tools along with our precious time.

Cleansing:

This step is meant to remove all texts that increases the size of the corpus, and not affecting the performance of the system, but the contrary. These include all the Latin alphabets, special characters, numbers and punctuations on one hand. On the other hand we found some terms that are repeated in most of the articles and are not adding any information related to the context but in some cases may cause performance deviations. These terms mostly found at the end of many articles and used for redirections or external links. (Table 3.2) presents texts that require deletion.

Table (3.2): Deleted texts and terms

Latin alphabets	e.g. A-Z, a-z
Special characters	e.g. !@#$% ε ā é
Punctuations	e.g. ; : , .
Numbers	0-9
Repeated terms	باللغة العربية، باللغة الإنكليزية، شاهد أيضا، ملاحظات، المراجع، المصادر، اقرأ أيضاً، طالع أيضا، أنظر أيضا، وصلات خارجية

After this step the corpus has a pure Arabic language content. Latin characters are mostly refer to names of persons, locations, etc. that are also written in Arabic such

as "Twitter" is written "تويتر". While punctuations are common in most languages, they can make words differ for example "عربي", "Arabic" is not equal to "عربي." with a period, "Arabic.". This step is vital because it is not performed in the subsequent steps.

Tokenization and stemming:

Tokenization is the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens, while stemming is the process of reducing inflected or sometimes derived words to their word stem, base or root form. Tokenization and stemming (also called lemmatization) are crucial to our system, because the generated terms are the input to Latent Semantic Analysis. Different term formation, may influence the system ability to match terms. To optimize this step we carried out a comparison between four commonly known Arabic Language processors, two stemmers Al-khoja(Khoja, 2001) and SnowBall(snowballstem, 2016), and two segmenters Stanford(CoreNLP, 2016) and Farasa (QCRI, 2016). To perform the experiment we have randomly selected 5 articles and applied each tool to their terms after removing all stop words and repetitions. The final set consists of 751 unique terms. (Table 3.3) shows out a snippet of the results. The complete set of results can be found on **https://github.com/YousefSamra/ShortTextTagging**

Table (3.3): Results of Arabic stemmers' comparison

Original	Al-khoja	SnowBall	Sanford	Farasa
الأردن	الاردن	الارد	الاردن	اردن
لبنان	لبن	بنان	لبن	لبنان
لبناني	لبن	لبنا	لبن	لبناني
ليبي	لوب	ليب	لوب	ليبي
بنسلين	نسل	نسل	نسل	نسل
الإيثيلين	ثول	ايثيل	ثول	ايثيلين
مياه	موه	ميا	موه	مياه
تمويه	موه	تمو	موه	تمويه
مارتن	مور	مار	مور	مارتن
ماراثون	أرث	ماراث	أرث	ماراثون
جار	جور	جار	جور	جار
سماء	سمي	سماء	سمي	سماء
سوائل	سول	سوايل	سول	سوائل
تسول	سول	تسول	سول	تسول
المدرسة	درس	مدرس	درس	مدرس

Original	Al-khoja	SnowBall	Sanford	Farasa
المسطحات	سطح	مسطح	سطح	مسطح
زوجين	زوج	زوجين	زوج	زوج
نوعان	نوع	نوعان	نوع	نوع
ضيف	ضيف	ضيف	ضيف	ضيف
ضفة	ضيف	ضف	ضيف	ضف
يضيف	ضيف	يضيف	ضيف	يضيف
مصب	صبأ	مصب	صبأ	مصب
قياس	قوس	قياس	قوس	قياس
أساس	سوس	اساس	سوس	اساس
كلفن	كلف	كلف	كلف	كلفن
كيميائي	كيميائي	يمياء	كيميائي	كيميائي
فيزيائي	فيزيائي	يزياء	فيزيائي	فيزيائي
برغوث	رغث	رغوث	رغث	برغوث

Also, we have calculated the precision and time efficiency for each tool. Results were judged according to correctness and suitability for our work but execution time is out of scope. Besides this step is one time execution, meaning that it will be done only once before the system runs. The only need for preprocessing after that is for constructing the input vector. Results in (Table 3.4) shows that Farasa has the best measures, and all tools out performed Stanford segmenter in both precision and execution time. This is because Stanford did not remove "ال" from the beginning of most terms that contain it, such as the first term in (Table 3.3). This is why we consider it inappropriate.

Table (3.4): Precision and time efficiency for NLP tools

Tool	Precision	Time
Farasa	89.88%	4.4 sec
Snowball	87.47%	0.6 sec
Stanford	73.90%	3.3 sec
Alkhoja	84.63%	13.14 sec

While investigating the results, we noticed that both Al-Khoja and Stanford are unifying Arabic terms that have different meanings in the context or generate wrong roots. For example, "ضفة""bank/shore","ضيف""guest" and "يضيف""add" all became "ضيف", while "مصب""estuary" was wrongly rooted to "صبأ""renounce" where the correct root is "صبب". Both also result in errors in the course of dealing with terms

containing "Hamza" "ؤ", "ئ".In addition, they work badly on both Arabic and non-Arabic names such as "تمويه" "camouflage", "مياه""waters", "ماراثون" "Marathon" and "مارتن"" Martin". This last fault was produced by SnowBall stemmer too. On the contrary Farasa segmenter works well on Arabic terms as well as on non-Arabic names, besides it does not completely root the Arabic terms which helps our system to distinguish between them. While still has the ability to take out the Arabic Additive letters and pronouns, which make it the algorithm of choice. We have chosen Farasa over snowball despite the difference in execution time because we are concerned in correctness of the results more than efficiency. Besides Wikipedia will be processed once by Farasa only when the system is built. We will call Farasa a stemmer because it help partially stem terms by removing the attached letters and additive pronouns.

As an example of Farasa **"اليوم موقعة قويّة بين تشيلسي ومانشستر سيتي وليفربول يترصّد",** **"A Strong match between Chelsea and Manchester City and Liverpool awaits"** the output of the algorithm was **"ال يوم موقع قوي بين تشيلسي ومانشستر سيتي و ليفربول يترصد"** while Al-khoja resulted in "**يوم وقع قوا بين تشيلسي مون سيأ ربل رصد** ", Snowball resulted in "**يوم موقع قو بين تشيلس مان سيت يفربول يترصد**". Other examples provided in (Table 3.3).

Stop-words removal:

Stop-words are commonly used words that are frequently appear in a corpus. Such words increase the size of the text and removing them doesn't affect the retrieving efficiency(Al-Shalabi, Kanaan, Jaam, Hasnah, & Hilat, 2004). We applied a stop-word removal algorithm to reduce the size of the corpus and improve the retrieving efficiency. Since our text is already cleansed and stemmed, the algorithm just iterates over the text and remove all the listed 266 words if found. For example, the previous text **"ال يوم موقع قوي بين تشيلسي ومانشستر سيتي و ليفربول يترصد", "A Strong match between Chelsea and Manchester City, and Liverpool awaits"** the output of the algorithm will be **"موقع قوي تشيلسي مانشستر سيتي ليفربول يترصد"** after removing **"ال" "the", "يوم" "today", "بين" "between"** and **"و" "and"**.

After this step each Arabic Wikipedia article will be presented as a title, a List of tokens (cleansed, stemmed, and stop-words removed), and a List of categories the

article associated with. These articles are now ready in a file to be distributed among the working nodes of a standalone Spark cluster.

At this point, our knowledge source contains only Arabic Wikipedia articles and each article body is presented as a list of tokenized and partially stemmed tokens. (Table 3.5) shows some information about our base knowledge.

Table (3.5): Statistics about the knowledge base

Our Work Needed Articles	435672
Number Unique Terms	662205
Number of Categories	267580

3.3 Tag Recommendation system

After preparing the data, it is now ready to go through the system. In the following steps we generate the singular value decomposition matrices to be searched for the most similar articles to the input short text, but first we need to calculate term frequencies Tf-idf, then convert document representation into vectors.

3.3.1 Computing the Tf-idfs

At this point all the articles are presented as Arrays of terms, each corresponding to a document. The next step is to compute the frequencies of each term in the document Tf, and for each term within the entire corpus DF. We apply Tf-idf weighting because it negates the effect of high frequency terms in determining the importance of a document. And we use log to the base 10, to diminish the values of the results, since we are dealing with huge number of documents and terms.

Tf-idf is a well-known numerical statistic that is intended to reflect how important a term is to a document in a collection or corpus (Han et al., 2011). And we employ it to gain statistics about our corpus as follows:

$$w_{t,d} = \log_{10}(1 + tf_{t,d}) \times (1 + \log_{10}(N / df_t)) \qquad \textbf{(3.1)}$$

Where $tf_{t,d}$ is the number of the term appearances in the document, N the total number of documents in the corpus, and df_t is the number of documents in the corpus that contain the term.

3.3.2 Vectorization

With the Tf-idf matrix in hand, we can perform the singular value decomposition, but first we need to convert the Tf-idf into sparse vectors for two reasons. The first reason is that it is essential to perform the singular value decomposition. The second reason depends on the nature of our data which contains mostly zeros for each document. A sparse vector implementation would be more space efficient since it only stores the indices of the terms and its non-zero values neglecting all terms with zero values which makes it a space efficient technique and help speed up calculations.

3.3.3 Singular Value Decomposition

Finally, we can proceed to the dimensionality reduction. MLib the machine learning library in Apache Spark contains an implementation of the singular value decomposition (SVD) that can handle enormous matrices. The singular value decomposition takes an m x n matrix and returns three matrices that approximately equal it when multiplied together

$$M_{(m \times n)} = U_{(m \times k)} S_{(k \times k)} V^{T}_{(k \times n)} \quad \textbf{(3.2)}$$

Where m, n, k are the number of document, number to terms and the number of concepts respectively. It is important to know that S is a k x k diagonal matrix that holds singular values. Each diagonal element in S correspond to a single concept or topic, which relates to a column in U and column in V and its magnitude correspond to the importance of this concept for the corpus. A key insight of LSA is that only small number of concepts are important to representing the data(Ryza et al., 2015). On the ground of that we chose k to be 1000 concepts, which is more than enough to represent the Arabic Wikipedia.

To make this as simple as possible, consider the example presented in chapter 2. After performing the SVD on Tf-idf matrix of 5 articles that contain 7 unique terms, the resulted 3 matrices will be theoretically as shown in (Figure 3.2) taking the number of concepts k=2.

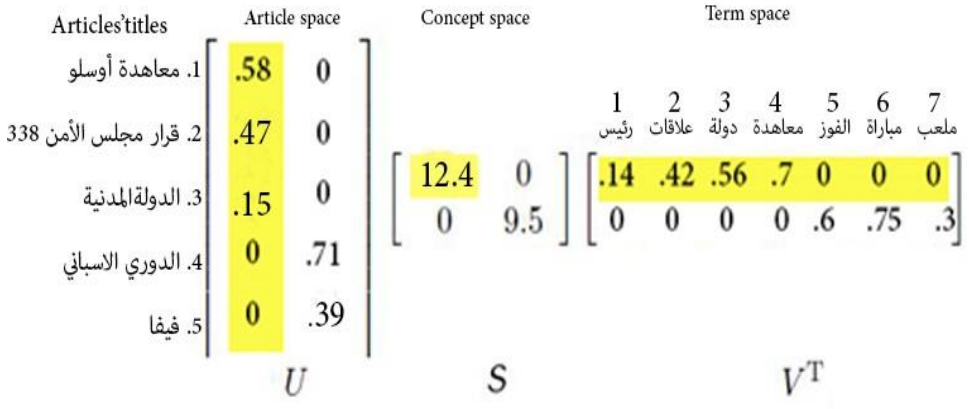

Figure (3.2): Result of SVD for a 5 documents 7 terms matrix

U is an m*k matrix whose columns form a basis for the article space. S is a k*k diagonal matrix, each of its entries correspond to the strength of a concept. V is a k*n matrix whose columns are basis of the term space.

It obvious from the values of S that the first concept is the most important in representing the corpus (5 documents) because it holds the largest value 12.4. This concept is related to the first column in U which holds 3 articles and also related to the first row in V which holds 4 terms. Let's be clear that the article "معاهدة أوسلو" in U is the most important to the first concept with value (0.58) while the article "الدولة المدنية" is the least important to the concept with value (0.15). Furthermore, the term "الدولة" in V^T is the most important to the same concept with value (0.56). As well, the first three documents in U and the first 4 terms in V contribute in the first concept but not the second since there values that correspond to the second concept are zeros. In other words, the first column in U and the first row in V^T are mapped to the first concept. At this stage, we can refer to a concept as the main topic that describes the articles it contains. But concepts are not names, they are just concepts. However, we can simplify things by naming them. For example, we can name the first concept "Policy""سياسة" or

"International affairs" "شؤون دولية" and we can name the second concept "Sports" "رياضة" or "Football" "كرة قدم".

A key insight of LSA is that only a small number of concepts are important to representing the data, e.g. two are sufficient in the example. So the corpus of the example basically talks about policy and football.

The system now is ready to receive the input short text and select the appropriate tags.

3.4 Tag Selection

After performing the SVD on the Arabic Wikipedia we can select tags for the input short text. The input text has to pass through the preprocessing steps. Then select the top similar articles. The preprocessing of the short text is vital because it allows us to map the terms of the short text to the terms of the Wikipedia. Note that the terms of the Wikipedia had gone through preprocessing in earlier steps. This allows two terms in both the short text and the Wikipedia article to be identified as equal and consequently the short text and the article are identified as similar.

It's clear now that the first two matrices U and S are the article space and the concept space respectively. Having a new preprocessed input of short text, we can compute the cosine similarity between itself and every other article simply by multiplying vectors and divide the result by their lengths (Sidorov, Gelbukh, Gómez-Adorno, & Pinto, 2014). (Figure 3.3) shows this part of the system, and Equation 3.3 represents the cosine similarity between vectors.

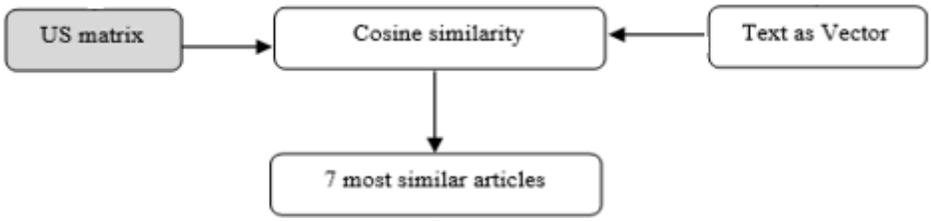

Figure (3.3): calculating the cosine similarity

$$\cos(q) = \frac{\vec{a} \cdot \vec{b}}{\left\|\vec{a}\right\| \left\|\vec{b}\right\|} \qquad \textbf{(3.3)}$$

The first vector in equation 3.3 is the short text, and the second is the rows of the US matrix each at a time. The result is a list of numbers each number is the similarity (score) between the input vector and an article vector of Wikipedia. These scores are sorted and the articles with the top scores are returned.

The cosine similarity is employed because: it is simple, very efficient to evaluate especially for sparse vectors and gives the value in between [0, 1]. But also we state two points (Baxla, 2014):

1. We need to match vectors of document in both magnitude and direction. Two document vectors compared to the input vector could have the same magnitude, but not equal. The direction can decide which vector is most similar to the input. This is a benefit of cosine similarity over Euclidian distance, Murkowski distance and Manhattan distance.
2. Compared to Jaccard similarity, adjusted based similarity and correlation based similarity these metrics used to calculate how much similar all the items are to each other in the matrix. Cosine and Jaccard similarities take less execution time and the cosine similarity performs excellent on huge matrices.

It is also worth mentioning that comparing two long vectors with small number of term is time inefficient, but the representation of the document and the tweet is done using sparse vector. A sparse vector keeps only the indices of the terms that has value other than zero. This help speed up computations and also increase space efficiency. But it may increase the creation time of the vector of the input tweet.

3.4.1 Text Preprocessing

The short input test goes through all text processing procedure as Wikipedia articles did.

Cleansing:

As discussed before all Latin characters, special characters, and punctuations, which presented in (Table 3.2), are removed.

Tokenization and stemming:

Separating all Arabic Language additive pronouns from terms, then partially stem these terms with the help of Farasa stemmer.

Stop-Word Removal

Removing all Arabic Language stop-words, including the generated additive letters and pronouns that was separated in the previous step.

3.4.2 Vectorization

The previous preprocessing will produce clean terms of the short text. These terms have to be formed as a vector to be compared to the Wikipedia articles in the concept space resulted from the SVD. As a matter of fact, these terms may contain some terms that are not in the Wikipedia, because of a miss-spilling for example. These terms has to be remove before creating the short text vector. The remaining terms are used to create the query vector by setting the value of the term to its inverse document frequency to maintain the weighting scheme used in the original term-document matrix (the input of the SVD) and compare it to the articles in the next step. Before the comparison and after forming the short text vector, it is multiplied by the matrix V^T to compute the concept space vector of the short text.

3.4.3 Selecting the top N Similar articles

Selecting the similar articles depends mainly of computing the cosine similarity between the vector of the short text and the rows of the US matrix. As explained previously, it is exactly as comparing document in the concept space, the only difference is that we compare a new document (short text) presented as a vector, then return the documents with the highest scores. This enables LSA to discover hidden semantics between the short text and documents.

The number 7 that we have chosen for our top articles to be retrieved has been determined through an experiment. We have processed 10 short texts and recorded the results of the experiment. We have carried out the test for 13 different number of top articles ranging from 2 to 20. After investigating the results we have decided 7 to be the number of the selected top articles. More details on the results are available on the next chapter. This test has to be carried out early in order to lighten the burden on expert while examining the results.

To give more insight into the importance of this step we report that experiment based on only the 10 short texts which resulted in around 2000 different tags. Imagine the number of tags that a 100 short texts would produce.

3.4.4 Selecting Tags

In Wikipedia, each article is assigned to a number of categories. Each category groups a number of Wikipedia articles together. The articles of a category are similar to each other. If we look closely to these articles we will find that they describe the name of the category they belong to or vice versa. Meaning if we consider the category name is a title of a book, each article is considered a chapter in that book. Any chapter in an English grammar book can be tagged "English grammar". Also an article can belong to a number of categories, consider the chapter "Introduction" that is found in many books.

In our system, tags are meant to be categories and titles of some of the 7 top articles similar to the short text. Because the tweet is similar to these top articles, the categories that contains some of them also can include the input tweet. In other words, this category- the one contains some of the 7 articles- describes the content of the tweet in a general way and can be used as a tag for it. Accordingly, because the tweet is similar to the content of these articles, their titles may be suitable as tags for the tweet. We consider a title to be appropriate if it contains some terms of the tweet. Titles that satisfy this condition are more specific than Wikipedia categories. Speculating in the example of (Figure 3.2) the short text "**معاهدة السلام الفلسطينية الاسرائيلية**" "**Palestinian Israeli peace treaty**" may result in similarities with the first two articles that share the *category* "**الصراع العربي الإسرائيلي**" "**Arab Israeli conflict**" which considered an

appropriate tag in a broad manner. This presents selecting 'categories as tags' discussed below. Furthermore, the title of the first article contains the term "**معاهدة**""**treaty**" which exists in the short text. This allows it to be elected as a tag. So it is given a higher score letting the title "**معاهدة أوسلو**" "**Oslo treaty**" appear in the top tag suggestions. This tag describes the tweet in particular. This stage has two steps; obtaining the categories of the 7 articles with the highest scores, note that we treat categories as if they have no hierarchy. Then adding analogous titles of the 7 articles as follows:

Categories as tags

It is obvious that if two articles are similar to each other, there is a chance to be partners in a category. We can refer to it as category, subject, topic, division, class, tag, etc. but let us call it category as it is in the Wikipedia. This means that it can be suggested as a tag. But our articles, which has been compared to the short text in the concept space, are assigned to variant types of categories, and we are concerned with the categories that involve some or all of them preferably. One simple way to identify these categories -or tags- is to pick out intersection between the categories of the articles. These tags are assigned a weight or a score equals the number of intersections. The highest the score of the tag, the most appropriate it would be. It is worth mentioning that categories cover the general aspects of the short text. We can describe the procedure as follows:

Procesure1: selecting tags from categories of top articles

Let D={d1, d2,.., d7} be the set of documents similar to a short test based on SVD.

Let Cdi = {Cd1, Cd2,..., Cdj} be the set of categories for document di

We compute the importance of each category by using the following equation:

Importance of C = $\sum_1^i \sum_i^{j,i\neq j} |d_i \cap d_j|$

For example, **"موقعة قويّة بين تشيلسي ومان سيتي وليفربول يترصّد" "A Strong match between Chelsea and Manchester City while Liverpool awaits"** the articles with the highest scores to this short text are shown in (Table 3.6).

Table (3.6): Categories intersections of the similar articles

Categories	Titles of top articles
• أندية الدوري الإنجليزي الممتاز • أندية رابطة الأندية الأوروبية	مانشستر سيتي
• أندية الدوري الإنجليزي الممتاز	نادي ليفربول
• أندية الدوري الإنجليزي الممتاز • أندية رابطة الأندية الأوروبية	تشيلسي

The categories **"أندية الدوري الإنجليزي الممتاز" "English Premier League clubs"** has 3 intersections, indicating that it is a category for three of the similar articles, and this make it appear first in the suggestions. While **"أندية رابطة الأندية الأوروبية" "European Club Association"** appears last as less relevant because it has only 2 intersections.

Titles as Tags

This is the second part of the tag selection procedure, after selecting the categories, the system moves on to check out the titles of the most similar articles. It is simply selects the title that contains a term of the short text. This is very efficient when the terms refer to names of persons, locations, etc. The title that suffice this criteria is likely to be a most relevant tag. Consequently, we set its score as the maximum category intersection +1. If the title contains more than one term, its score is incremented by the number of terms it contains. We can describe the procedure as follows:

```
        Procedure2: selecting tags from titles of top articles
Let T={t1, t2, …, tn} be the terms of the tweet
Let MaxCatScore be the maximum score of categories
Let Li={l1,l2,…,l7} be the set of the titles of the 7 top articles

      We compute the importance of the title as follows:
      For i=7 to 1
      IF li contains terms in T THEN
            Set score of li = MaxCatScore + number of terms it contains
```

For example, referring to the example in (Table 3.6)**"موقعة قويّة بين تشيلسي ومان سيتي وليفربول يترصّد"** the titles of the selected articles that contains a term of the short text are **"تشيلسي" " Chelsea", "مانشيستر سيتي" "Manchester City", and "ليفربول" "Liverpool",** and they are more relevant and appropriate as tags than categories. So,

they are assigned a higher weight. Each title has a score 4 which equals 3+1. Checking the titles is carried in reverse order as the procedure suggests. This means that we examine the titles with the least scores before the ones with high scores. It keeps the order of the selected titles unless one contains more than one term. In the example above the order of titles will be as presented in (Table 3.6) even they has the same score. Titles cover the specific aspects of the short text unlike categories that are broader. The criteria we adopted let title tags appear at the top in the suggestions, while categories appear last.

3.5 Case study

In the following case study, we illustrate a full scenario of the short text tag suggestion, showing how the short text is processed, until the suggestion of tags. At this point our system is started, Wikipedia formed into three matrices, and these matrices are stored in the memory in a distributed fashion, ready for any input.

Suppose a user posting **"الفرق بين المبرمج ومصمم الجرافيك☺" "Programmer vs graphic designer☺"** on a social media website, for example. Our system grabs the text of the post and suggest tags for it as follows:

1. Preprocessing

The input text is first processed by cleansing all non-Arabic letters, punctuations, and special characters. Afterwards, the text is tokenized and segmented. Finally, stop-word removal is applied, as (Figure 3.4) shows.

Figure (3.4): Preprocessing of the short text

2. Vectorization

The terms of the short text is ready to be formed as a vector. This is done by setting the value of the term to its inverse document frequency to maintain the weighting schema of the original matrix. (Figure 3.5) shows the text as vector. Then the vector is multiplied by the matrix V^T to compute the concept space vector of the short text.

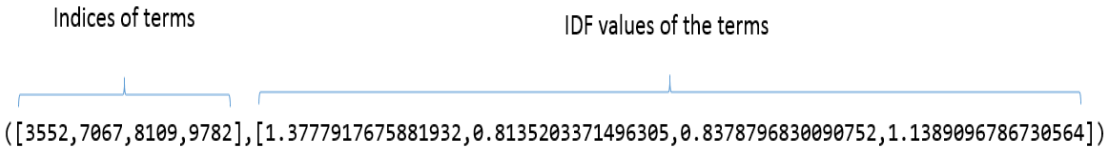

Figure (3.5): Short text as a vector

In the Tf-idf matrix of the Wikipedia articles, each column represents an article where each row in that column in the importance of a term in that article. We can refer to this column as the vector of the article

We treat the tweet as an ordinary article in Wikipedia, and that its Tf-idf score is calculated with reference to Wikipedia as a corpus. Tf-idf is calculated by first calculating the frequency of terms in the tweet but we consider it as if appears once in the short text in our case while if it appears twice or more it will has a negligible effect compared to Wikipedia. Then, the inverse document frequency is calculated by dividing the total number of Wikipedia articles by the number of articles containing the term, and then taking the logarithm of that quotient. This formula is presented in equation 3.2.

The aim of vectorizing the tweet with reference to Wikipedia as a corpus is make its Tf-idf representation comparable to the Tf-idf representation of other Wiki articles, and thus the application of the similarity measure (cosine measure) becomes possible using Equation 3.3.

3. Select 7 most similar articles:

The vector generated in the previous step is now compared to the rows of the US matrix, which denotes the Wikipedia articles. The dot product between the tweet vector and each row of the US matrix results in the cosine similarities between them. Then articles are sorted according to that similarity and the 7 top articles are retrieved. (Table 3.7) shows the 7 articles with the highest scores that are similar to the short text in this case study.

Table (3.7): The top 7 articles for the tweet

تصميم الجرافيك
مبرمج
فريق العمل لإنتاج برمجيات الوسائط المتعددة
علم الحاسوب
مصمم جرافيك
رسوميات
تصميم المعلومات

4. Tag selection

With the articles in hand, the system looks for intersections between the categories of the top articles, setting the number of the intersections as score of the category (tag). It is performed according procedure 1. (Table 3.8) shows the categories and their scores. In this case the first category has score=3 indicating better suitability than the other two. But the list will be updated in the next step.

Table (3.8): Categories selected by the system

Category	Weight
علم الحاسوب (Computer science)	3
تصميم الجرافيك (Graphic design)	2
مهن الحاسوب (Computer occupations)	2

After finding all the category intersections, the system looks for titles that contain terms of the input short text. If found, the system sets the weight of the title to the maximum category score incremented by the number of terms in the short text it contains according to procedure 2. (Table 3.9) show the titles selected by the system.

Table (3.9): Titles selected by the system

Title	Weight
مصمم جرافيك (Graphic designer)	5
تصميم الجرافيك (Graphic design)	4
مبرمج (Programmer)	4
فريق العمل لإنتاج برمجيات الوسائط المتعددة (Multimedia development team)	4
تصميم المعلومات (Information design)	4

The first title contain two term of the short text "مصمم" and "جرافيك". The score is set to 3+2 terms =5, while the other contain only one term each, so the weight is set to 3+1=4.

All titles in the (Table 3.9) are more appropriate than categories in (Table 3.8) as tags. Tags in both tables are presented to the user in a descending order showing the tag with the highest score at the top of the list. The categories are replaced by the titles that equals them such as the category "Graphic design" " تصميم الجرافيك". The full list of the tags are presented in (Table 3.10). Luckily, all the tags in both tables considered suitable except for "تصميم المعلومات" "Information design". Also, one can notice that titles with the same scores are presented in their same order of relevance (refer to (Table 3.7)

Table (3.10): Suggested tags for the tweet

Suggested tags
مصمم جرافيك (Graphic designer)
تصميم الجرافيك (Graphic design)
مبرمج (Programmer)
فريق العمل لإنتاج برمجيات الوسائط المتعددة
تصميم المعلومات (Information design)
علم الحاسوب (Computer science)
مهن الحاسوب (Jobs)

3.6 Tools

- **Wikixmlj**

Wikixmlj is a Java API for parsing Wikipedia XML dumps (wikixmlj, 2016). It is part of the larger WikiSense project aimed at understanding Wikipedia for semantic annotation of texts. It provides easy access to Wikipedia XML dumps, and have been

used in different works(Santoso, Nugraha, Yuniarno, & Hariadi, 2015). Wikixmlj is available on (Github, Wikixmlj 2016).

- **Farasa Segmenter**

Farasa is a fast and accurate text processing toolkit for Arabic text. Farasa consists of the segmentation/tokenization module, POS tagger, Arabic text Diacritizer, and Dependency Parser. It have been used in recent works(Abdelali, Darwish, Durrani, & Mubarak, 2016). Farasa is available on(QCRI, 2016).

- **Apache Spark**

Apache Spark (Sprak, 2016) is an open source big data processing framework built around speed, ease of use, and sophisticated analytics. It was originally developed in 2009 in UC Berkeley's AMPLab, and open sourced in 2010 as an Apache project(Zaharia et al., 2010).

It provides a highly-optimized machine learning library called MLlib (Meng et al., 2016) which has several features that are particularly attractive for matrix computations (Bosagh Zadeh et al., 2016; Zadeh et al., 2015). Spark enables us to maintain the huge data in memory in a distributed manner.

3.7 Summary

This chapter presents the methodology we followed to construct our tag suggestion system. First, the XML dumb was parsed for complete articles; body, titles and categories and stored in a text file to be distributed among working nodes. Then the tagging process begins by preparing the system. Text preprocessing is applied to the bodies of the articles, cleansing, segmenting and stop-word removal in order. The third step is constructing the Tf-idf matrix then the Singular Value Decomposition. The system is now ready to receive any input which is the fourth step. The input is preprocessed, vectored, and compared the articles in the concept space to find the most similar ones. The final step is to generate tags from the titles and the categories of these articles. The category selection is based on the intersection, while the title selection depends on containing a term of the input text.

A detailed example was discussed as a case study. The proposed system thoroughly capable of suggesting probable and suitable tags for any short text.

Chapter 4
Results and Discussion

Chapter 4
Results and Discussion

4.1 Introduction

This chapter presents the system we utilized to assess and evaluate our tag recommender system. The main objective of the evaluation is to assess the reliability of the tag recommendation system: we aim to explore the extent to which the proposed system can accurately suggest suitable and correct tags to the input tweet from relevant Arabic Wikipedia articles.

Similar approaches from the state of the art have been evaluated by being compared to other approaches (Hassan et al., 2012; Otsuka et al., 2014). However, we are not aware of any similar approach that utilizes the Arabic version of Wikipedia for the tagging of short texts to compare with. Therefore, we opted to assess our system by experts' evaluation of the results.

4.2 Dataset

The dataset is a set of 100 tweets selected randomly from three different domains: Sports, Technology, and News mainly Palestinian news. The tweets were divided according to the subjects as follows: Sports; 36 tweet, Technology; 41 tweets, and News; 23 tweets. The aim is to assess how the generated recommendations are affected by changing the domain of knowledge. In addition, we emphasize that the selected 100 tweets were used only for the evaluation step, and were not used beforehand to tune or test the system during the design and implementation. (Table 4.1) shows a snapshot of the dataset. The complete dataset can be downloaded from **https://github.com/YousefSamra/ShortTextTagging.**

Table (4.1): A snapshot of the gathered dataset

Subject	Tweet
Sports	موقعة قويّة بين تشيلسي ومان سيتي وليفربول يترصّد
Technology	سيانوجين مود رائدة تطوير رومات الأندرويد
News	فلسطين المحتلة: الصحفي محمد القيق يواصل إضرابه عن الطعام بسجون الصهاينة

4.3 Experiment settings

Some sizes of data cannot be processed on a single machine. Operations on data may require memory spaces that could not be located in one machine. Performing the singular value decomposition on the Arabic Wikipedia requires tens of gigabytes of memory to make it doable. Besides, this heavy computations needs an efficient environment to handle it in a reasonable time, regardless that we are not concerned of time efficiency in our experiment. Those reasons lead us to utilize Apache Spark in the experiment. We restrict our attention to Spark, because it provides a highly-optimized machine learning library called MLlib (Meng et al., 2016) which has several features that are particularly attractive for matrix computations. Spark cluster parallel environment provides us with a sufficient memory space that is distributed among the nodes of the of a standalone cluster.

The experiment were carried out in a computer lab. It consists of 20 identical laptops which we used as a Spark cluster. The settings of the experiment was as follows:

1. **Master node**: it is the computer that executes the code of the system and organize the communications with other worker nodes, collects and saves the results. The specifications of this machine is depicted in (Table 4.2).

Table (4.2): Master node specifications

Machine	HP laptop
CPU	Core i5 2.6 GHz
RAM	6 GB
OS	Windows 10, 64bit

2. **Worker nodes**: are 20 computers that are connected to the master node. Each node provide CPU and memory space for the tasks assigned by the master node. Nodes sends results to the master node when needed. (Table 4.3) depicts the specifications for worker nodes.

Table (4.3): Worker nodes specifications

Machine	Dell laptop
CPU	Intel Core i3 2.53GHz
RAM	4 GB
OS	Windows 10, 64bit
Worker assigned CPU cores	4 cores
Worker Assigned Memory	2.8 GB

The experiment settings provided us with 80 CPU cores and around 56GB of memory that were sufficient to complete our tests.

Our data which is the Arabic Wikipedia articles that are cleaned, tokenized, and segmented were transferred manually to every worker node. Data is needed on worker nodes to lighten the load of communications and data transfer among the cluster. Also, we had to deploy Apache spark on worker nodes. Worker nodes had to be started manually because there is no way to start them automatically.

After starting the master and the worker nodes, we can run our code on the cluster and record the results to be evaluated.

4.4 Evaluation Process

The evaluation process had two experiments. First experiment aimed to determine the number of the top articles the system has to utilize in order to result in a qualified and considerable number of tags. The second experiment was for the assessment of our system. We ran the tag recommender on the dataset and recorded the results which are an ordered set of titles and categories of top articles (as tags) for each tweet. In the next sections we discuss in details the two experiments and their results.

4.4.1 Experiment 1: Determining the top N articles

As explained in Section 3.4.3 in Chapter 3, the tweet, will be compared with Wikipedia articles by using the cosine similarity measure. Then, top similar articles will be used to identify the recommended tags by exploiting their titles and categories (refer to Section 3.4.4) Therefore, we aim at this stage to explore how the accuracy, in

terms of the correctness of generated tags, is affected by changing the number of top articles used for tag recommendation. We also aim to optimize our system by identifying the best number of articles that should be used to give the best possible recommendations.

We tested our recommendation system with only 10 tweets while varying the number of top similar articles from 2 to 20. For example, the first trial used only the top 2 Wikipedia articles to recommend tags, while the last trial used 20 articles. Tags generated from each trial were validated by six human experts, two in each field, who marked each tag as "Correct" or "Incorrect". A tag was considered correct if it highlighted the meaning of the tag, or it can be used to categorize the tweet. (Table 4.4) shows a tweet and a sample of the resulted tags. The first column presents the tweet. The second and the third columns present correct tags. Tags in the first column highlights the meaning of the tweet while tags in the third column are considered categorization of the tweet which describes the topic (subject) the tweet belongs to. The last column presents incorrect results that experts considered inappropriate as tags.

Table (4.4): Correct and incorrect tags of a tweet

Tweet	Correct tag (Highlighting tag)	Correct tag (Categorizing tag)	Incorrect tag
قوات الاحتلال تقتحم شمال مدينة الخليل	محافظة الخليل	فلسطين	نزاعات في 2015
	الخليل	القضية الفلسطينية	مدن محافظة رام الله والبيرة
		إرهاب صهيوني	هجمات إسرائيلية ضد قطاع غزة

It is important to notice that the total number of generated tags from all trials was 2007. This large number of tags that needed to be validated by the experts explains why we limited the number of tweets for this experiment to 10 tweets only rather than 100 tweets.

(Table 4.5) illustrates the results of changing the number of similar documents. The first column shows the changing number of articles for the ten tweets. The second column shows the average number of recommended tags that are correct for each number of articles for all tweets. The third column shows the average number of incorrect tags for all tweets. The final column shows the accuracy for each trial. Notice

that the total number of recommended tweets increases as the number of articles increases.

Table (4.5): Result of experiment on 10 tweets with different number of top articles

Number of top articles	Correct tags for 10 tweets	Incorrect tags for 10 tweets	**Accuracy**
2	15	6	71.43%
4	29	16	64.44%
6	51	29	63.75%
7	66	32	67.35%
8	72	39	64.86%
9	82	45	64.57%
10	96	50	65.75%
11	102	55	64.97%
12	110	63	63.58%
14	119	88	57.49%
16	128	112	53.33%
18	140	136	50.72%
20	155	171	47.55%

During early investigation of the results applying a step of 2 for the number of top articles, we noticed a small peak of accuracy (65.8%) at 10 top articles. Testing other values for top articles before and after this peak was required. Therefore, we recorded results for 7, 9, and 11 top articles as presented in (Table 4.5).

(Figure 4.1) shows how the accuracy, number of correct tags and number of incorrect tags change for different number of top articles judged by human experts. The x-axis presents the number of selected top articles (N), and the y-axis presents the number of generated tags.

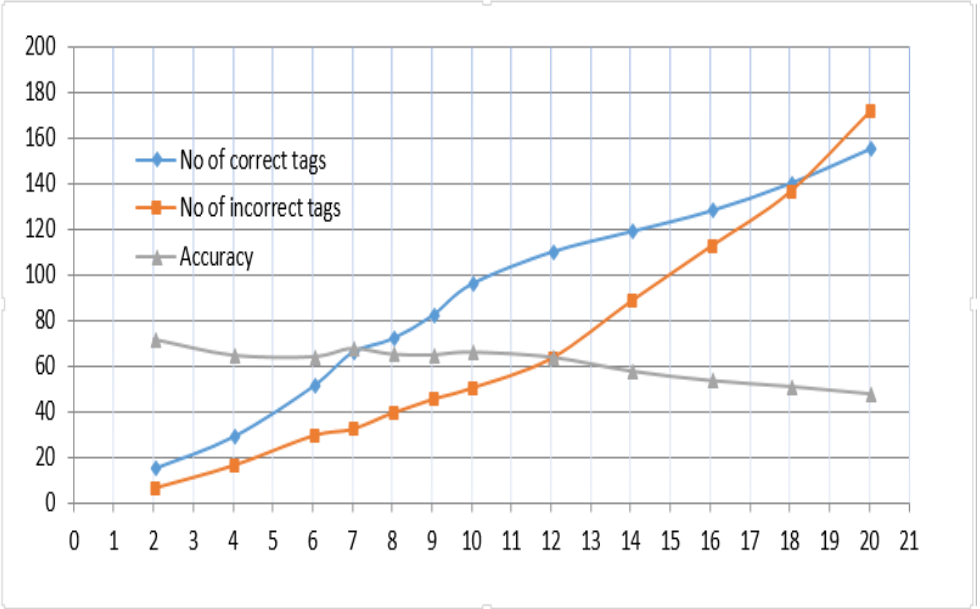

Figure (4.1): Results for 10 tweets on different number of top articles

(Figure 4.1) shows that at low number of articles the accuracy looks high, but the number of tags are very small. For example at 2 articles the average accuracy was 71.4% but the number of tags was (1-2) tags on average for each tweet which is very small and sometimes not related to the short text. Some texts had 100% accuracy while some had 0%. While at 18 articles, the incorrect tags began to exceed the correct ones causing accuracy to drop to 50.7%. The system resulted in (8 to 24) correct tags, and around the same number of incorrect ones for each tweet.

Using bigger number (N) for top articles to have more tags will also increase the number of wrong ones. comparing the results when N=20 and N=18 the system added 6 more correct tags but also introduced 19 more incorrect at N=20, which decreases accuracy and increases tag ambiguity. Besides, using bigger number of top articles increases the number of correct tags, but the majority of these new tags are general or broad which categorize the tweet rather than highlighting the meaning of it. For example, (Table 4.6) shows results at N=18 for the tweet in the previous example in (Table 4.4), all of the new tags are general and similar to the tags in the third column. But no specific or highlighting tags were added.

Table (4.6): New tags at 18 top articles for example in Table (4.4)

Tweet	Correct tags (categorizing tags)
قوات الاحتلال تقتحم شمال مدينة الخليل	قرى فلسطين
	بلديات فلسطين
	مدن مقدسة
	مدن الكتاب المقدس
	مدن كنعانية

The best number of selected top articles (N) that the experiment suggests is tend to be 7 articles, which preserve balance between the number of correct tags (5-10) for each tweet and an acceptable accuracy of 67.4% at this experiment. On the other hand 7 articles will generate a reasonable number of tags for experts to conceive. There will be 10 resulted tags for tweets. Accordingly, 7 top articles is the N that we chose for our system, avoiding our experts the burden of fruitlessly investigating an immense number of tags, leaving other choices for future work. However, restricting the experiment on only 10 tweets is a limitations, since repeating the experiment on a different 10 tweets may result in selecting different number of top articles. Investigating results for a second experiment costs the experts time and effort. But we believe that the selected number of top articles will remain around 7 based on the structure of Wikipedia.

4.4.2 Experiment 2: evaluation of the system

For the assessment of our system we ran the tag recommender on the dataset and recorded the results which are an ordered set of tags for each tweet. Tweets with their corresponding generated tags were divided into 3 groups according to subject domains. Then each group was handed to two human experts in each domain to examine the tags and mark the suitable ones. Since two human experts validated the tags, we considered only the tags that both experts agreed upon to be correct. (Table 4.7) shows how each tweets and its recommended tags are presented to the expert for validation. The expert was asked to mark each tag as "1" if it is correct or "0" if it is incorrect.

Counting only the 10 top ranked tags that all experts agreed upon, there were 658 appropriate tags from 933 resulted tags yield in 70.35 average accuracy .The correct tags were divided as follows; Sports; 227, Technology; 276, and News; 155

correct tags. These results were used to evaluate our system. The full results collected from the experts can be downloaded from **https://github.com/YousefSamra/ShortTextTagging.**

One should also notice that the order of recommended tags was preserved and considered in the evaluation. A good recommender approach should order recommendations so that most relevant ones come first.

Table (4.7): Tags of a tweet evaluated by experts

واتساب 2017: توقف خدمة واتس اب عن العمل على بعض الهواتف اكتشف ان كان هاتفك من القائمة	
1	واتسآب
0	سناب شات
1	تراسل فوري
1	برمجيات آي أو إس
1	برمجيات أندرويد
1	برمجيات متعددة المنصات
1	برمجيات اتصال
1	مراسلة فورية
0	برمجيات بلاك بيري
0	برمجيات سيمبيان

4.5 Evaluation Metrics

Most state of the art works have adopted precision(Gong & Liu, 2001), recall(Otsuka et al., 2014) and f-measure (Hassan et al., 2012) to evaluate the performance of their approaches. While being simple and descriptive, recall and consequently F-measure, requires a pre-knowledge of all possible correct tags for each short text, which is infeasible in our case.

Therefore, what is appropriate for our tag recommender is to take into account the rank of the items. In recommender systems, the most important result for a final user is to receive an ordered list of recommendations, from best to worst. So, we adopted Precision at position K (P@K) where k from 1 to 10, Mean Average Precision (MAP), and Mean Reciprocal Rank (MRR). Works, such as (Allahyari & Kochut, 2016a; Bogers & Van den Bosch, 2008) had applied these metrics.

The first two metrics emphasize on the quality of the top K tags, while the MRR focuses on a practical goal, "how deep the user has to go down a ranked list to find one useful tag?" (Sun, Chen, & Rudnicky, 2017).

The metrics are defined as follows(Liu, 2009):
To define MAP, one needs to define Precision at position k (P@k) first,

$$\mathrm{P@k(q)} = \frac{\#\{\text{relevant documents in the top k positions}\}}{k}$$

K in our system denotes the number of recommended tags for each tweet. For example, P@5 corresponds to the number of relevant tags for a tweet from the first 5 results. We aim to explore how the precision is affect when changing the number of tags to be examined.

Then, the Average Precision (AP) is defined below:

$$\mathrm{AP(q)} = \frac{\sum_{k=1}^{m} P@k(q)}{\#\{\text{relevant Documents}\}}$$

Where m is the total number of documents associated with query q. The mean value of AP over all the test queries is named MAP.

$$\mathrm{MAP} = Avg(\sum_{q=1}^{n} AP(q))$$

Where n is the number of queries.

Mean reciprocal rank (MRR): For query q, the rank position of its first relevant document is denoted as r(q). Then 1/r(q) is defined as MRR for query q. It is clear that documents ranked below r(q) are not considered in MRR.

$$\mathrm{MRR} = \frac{1}{\mathrm{r(q)}}$$

Based on the above definitions, the metrics in our experiment are calculated as follows:

$$P@k(q) = \frac{\#\{\text{relevant tags in the top k positions}\}}{k} \quad \textbf{(4.1)}$$

$$MAP = Avg(\sum_{q=1}^{100} AP(q)\ ) \quad \textbf{(4.2)}$$

$$MRR = Avg(\sum_{q=1}^{100} \frac{1}{r(q)}) \quad \textbf{(4.3)}$$

Also we have calculated precision for more evaluation using the following equation

$$precision = \frac{number\ of\ all\ correct\ ta\ gs}{number\ of\ all\ resulted\ tags} \quad \textbf{(4.4)}$$

Recommended tags for each tweet were first assessed by human experts. The above evaluation metrics were calculated based on the expert's evaluation of tags. (Table 4.8) depicts a sample short text, ordered tag results, expert evaluation, and the calculations of P@k, AP@k, and reciprocal rank where maximum k=10.

Table (4.8): A short text, resulted tags, expert evaluation and measures calculations

RR	AP@K	P@K	Experts judgement	واتساب 2017: توقف خدمة واتس اب عن العمل على بعض الهواتف اكتشف ان كان هاتفك من القائمة	
1	0.82602	1	1	واتسآب	1
		0.5	0	سناب شات	2
		0.666667	1	تراسل فوري	3
		0.75	1	برمجيات آي أو إس	4
		0.8	1	برمجيات أندرويد	5
		0.833333	1	برمجيات متعددة المنصات	6
		0.857143	1	برمجيات اتصال	7
		0.875	1	مراسلة فورية	8
		0.777778	0	برمجيات بلاك بيري	9
		0.7	0	برمجيات سيمبيان	10

4.6 Results and Discussion

(Table 4.9) presents the evaluation metrics of the tag recommender.

Table (4.9): Evaluation metrics of the system

Number of generated tags @ k=10	933
Number of correct tags	658
Mean Average Precision	84.39%
Mean reciprocal Rank	96.53%
Precision	70.35%

The results depicted in (Table 4.9) have been calculated for 100 tweets in three different domain subjects processed through the system and then judged by experts - in each subject- for tag suitability and relevance. We have expected around a thousand tags, ten for each tweet on average according to experiment 1. We had 933 tags because some of the tweets had less than 10 tags. Their top articles belong to different categories. It is possible to have such number of tags based on the top articles. Top articles that are related to each other share categories more than weakly related top articles. Shared categories are suggested as tags.

Inspection of the results revealed that the system achieved a good performance by 84.39% mean average precision, which as we think and the results suggest are adequate for a tag recommendation system. Also the system achieved a considerable mean reciprocal rank of 96.53% which means that the user will find a suitable tag as the first or mostly the second result that proofs the effectiveness of our simple rank algorithm. But this was not the case with all input tweets, we have recorded a few where a suitable tag did not appear neither first nor second. As an example, the tweet "**أبردين يعزز موقعه في المركز الثاني بالدوري الأسكتلندي**" had only one proper tag at k=6 resulting in AP@k =16.67% and reciprocal rank = 16.67% too. Detailed discussion is provided in the next section.

We were also interested in examining the differences across different subject domains. Results for each subject domain is depicted in (Table 4.10). Results from the table below shows that MAP and MRR are close for the three subject domains, suggesting the adequacy for other different subjects.

Table (4.10): Results across different subjects

Subject	NO tweets	MAP	MRR
Sports	36	80.81%	95.46%
Technology	41	85.85%	96.83%
News	23	87.12%	97.83%

While applying more investigation into the results we noticed that the precision is higher at the top of the list. Meaning, as we encounter new results from the list of recommendations, the precision drops down indicating weak relatedness of the tags at the rear of the list. (Figure 4.2) shows average precision for the results of the 100 tweets at k=1 to 10. This result is consistent to a large extent with most web search and information retrieval systems since it introduces more relevant tags at the top of the list than on the bottom of the list.

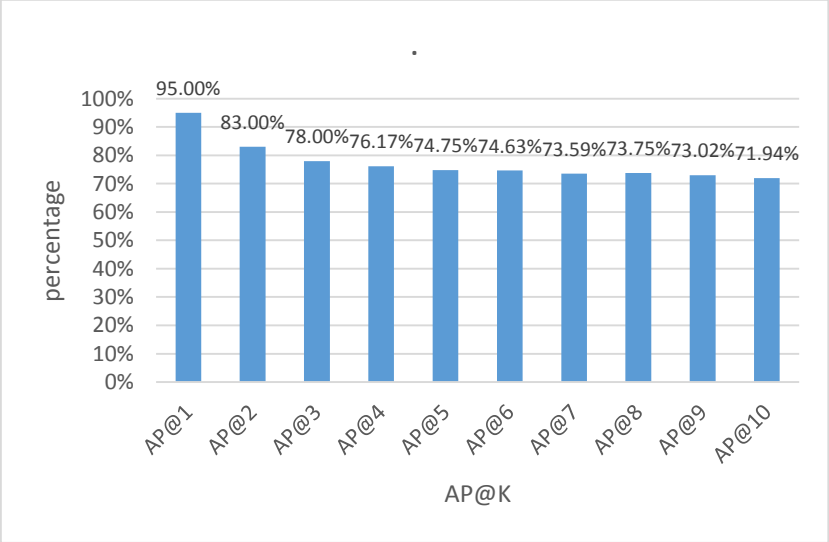

Figure (4.2): $AP_{(1-100)}@k_{(1-10)}$

To further explain our results, we inspected the results thoroughly to identify the main sources of strengths and weaknesses. Strengths can be stated in the following points:

1. Comparison in the concept space: this is mainly the job of singular value decomposition. Classifying articles into concepts before comparing them with the input tweet gives higher scores to the articles in the concept that the tweet belongs to. Leading to better matches to the input. For example the term "**زيدان**" could be the philosopher "**يوسف زيدان**", the actor "**أيمن زيدان**", or the media figure "**بدر آل زيدان**", but comparison in the concept of the tweet "**زيدان: أخشى أن يسقط ريال مدريد مجددا**" resulted in "**زين الدين زيدان**" and "**محمد زيدان**" as the second and the third similar articles, where both of them are football players. This technique allows the tags to be semantically related to the concept the tweet belongs to.

2. Tag selection procedure: as discussed in Chapter 3, the tag selection covers two parts; the categories then the titles of the top articles, giving the titles higher priority. This allows the tags suggested by the system to cover specific followed by general aspects of the tweet. For example the tweet " **فلسطين المحتلة: الصحفي محمد القيق يواصل إضرابه عن الطعام بسجون الصهاينة**" has a specific tag "**محمد القيق**" and general tags such as "**أسرى ومعتقلون فلسطينيون**" and "**القضية الفلسطينية**". Also, " **غسان كنفاني: روائي وقاص وصحفي فلسطيني تم اغتياله على يد الموساد الصهيوني**" has a specific tag "**غسان كنفاني**" and general ones "**أدباء وكتاب فلسطين**" and "**الصراع العربي الإسرائيلي**". Specific tags are suggested from titles while general tags are suggested from categories.

No human work is perfect. This work has some weaknesses that can be classified into the following categories based on the source of weakness:

1. Polysemy: is a word that have different meanings in different contexts. In Arabic most words have different vowelization. Although, vowelization mostly not applied which contributes into polysemy. Moreover, in order to compute the Tf-idf matrix, our system needs to remove all form-adjustment of terms if existed. Polysemy causes the system to miss-interpret the term with another. For example, the term "**بيت، شعر، تراث**" "residence, hair, heritage" in " **ملعب كرة_قدم في قطر على شكل بيت شعر بمساحة تتسع ل٦٠ الف ينتهي عام ٢٠١٨ تصميمات مستوحاة من ثراثنا**" are related to a concept not related to the context of the whole sentence. Our system suggested tags not related to the context such as:"**شعر (أدب)**","**إيقاع شعري**", and "**قافية**". On the other hand, more related suggested tags such as"**ملاعب كرة قدم في قطر**" got low ranks.

2. Synonymy: is a concept having multiple forms of representation. Arabic language is full of synonymy. It is a common drawback in models like LSA. In addition, Arabaization introduces synonyms that is written in Arabic alphabets but their pronunciation sounds foreign such as **computer** "**كمبيوتر**", "**حاسوب**", **mobile** "**جوال**", "**موبايل**", "**هاتف**", "**محمول**", and **software** "**برنامج**", "**تطبيق**", "**سوفت وير**". These new synonyms introduce more complications to LSA. Wikipedia tries to manage the synonymy and polysemy problems by introducing page redirections. Redirections allow users to search for terms while Wikipedia takes care of the synonyms. But this is not the case in LSA. For example, the term "**حاسوب**" is found much more than the

term "**كمبيوتر**" in Wikipedia articles. Because articles such as "**حاسوب**" "**حاسوب شخصي**" have redirect pages from "**كمبيوتر**" and "**الكمبيوتر الشخصي**" respectively. While the Articles contains only the term "**حاسوب**".

Consider, the tweet **" اذا انت حاب تتعلم برمجة مواقع وبرامج ايفون واندرويد وبرامج كمبيوتر كلها متوفره بلغة برمجة وحده اسمها سي شارب"** contains the term "**كمبيوتر**" while the suggested tags was not influenced by the term "**كمبيوتر**" allowing other tags like "**آي**","**هواتف ذكية**" and "**أندرويد**" that are more related to the rest of the terms to appear. We refer that to existence of the term "**حاسوب**" rather than "**كمبيوتر**" in the relevant articles. Since most terms in Arabic Wikipedia that means computer is written "**حاسوب**".

On the contrary, the term computer "**حاسوب**" in the tweet"**أول مواجهة بين حاسوبين كميين**" resulted in more adequate tags such as "**حاسوب**", "**معمارية الحاسوب**" and "**حاسوب كمومي**". The term "**حاسوب**" was found in number of articles that are three times the articles contain the term "**كمبيوتر**". The term distribution over many articles offers a better chance to be combined with other terms of the tweet in the same concept, leading to better tag selection.

3. Different ways for writing a foreign term: foreign terms that have been Arabized could be written in different ways. It could be introduced as polysemy or miss-spilling. However, the change either includes one letter of the term such as English "**إنجليزي**", "**إنكليزي**", "**إنغليزي**", and Instagram "**إنستجرام**", "**إنستغرام**", "**إنستقرام**" or adding a space to split the terms such as iPhone "**آيفون**", "**آي فون**" and Hard disk "**هارديسك**", "**هارد ديسك**". Such writing if does not match with a similar term, affects the tags to be biased to the other terms of the input text. This writing differences was the major contributor in failure of extracting the expected tags. For example, the tweet " **طريقة تثبيت ويندوز من خلال الهاردسك نفسه**" contains the term "**هاردسك**" where there is no such term in the whole Arabic Wikipedia and the system could not find relevant articles to it. Therefore, only three suitable tags were introduced "**ويندوز**", "**مايكروسوفت ويندوز**", and "**أنظمة تشغيل**". While the bitter truth is that the term in different writing "**هارد ديسك**" is found in 9 articles 4 of them combined with the term "**ويندوز**" which if written properly, may lead to electing other suitable tags.

Also, the tweet "**هدف جيرو ينتزع التعادل لأرسنال أمام يونايتد**" contains a name of a football player "**جيرو**". But the system could not find resemblance with the article "**أوليفيه جيرو**" because the name is written "**غيرو**" in the body of the article. Unfortunately, this title of the article was not suggested as a tag.

Another tweet "**جيمس جوسلينج مخترع لغة الجافا**" contains "**جوسلينج**", Java inventor's name, but no relevance to the article "**جيمس غوسلينغ**" was found because it is written "**جوسلنج**" in the body of the article, accordingly was not selected as a tag.

4. Terms written in English: some of tweets has terms that are written using Latin characters, these term are deleted in the prepressing step of the tweet. Therefore, they have no effect on the results. For example, in the tweet" **XML file لاستيرده في ووردبريس طريقة تقسيم**" the terms "**XML file**" was deleted. So the tags focus on other terms resulting in suggesting "**ووردبريس**", and "**برمجيات التدوين**" as tags, while not mentioning XML files.

5. Distinguishing names: we refer to distinguishing terms as the terms that are found in a few articles such as names of persons, places, etc. in a concept. Which cause the articles containing these terms to be highly relevant to the input tweet and gain high scores while being weakly related to the context of the tweet. And since relevant articles are used to select tags, the resulted tags tend to be not descriptive or irrelevant.

For example, the term "**موغيريني**" is a name of one person that no one else shares the same name in the whole Wikipedia, and it is found in only 37 articles. This distinguishing term in "**صربيا وكوسوفو تبحثان في نتائج زيارة موغيريني**" lead to a scattered set of top articles with high scores. The top articles gave 11 irrelevant tags out of 14 such as "**أشخاص قتلوا بإطلاق النار في تونس**", "**هجمات على سياح**", and "**إرهاب في تونس**". Fortunately, the first result was "**صربيا**" which is relevant to the tweet.

6. Title contains a tweet term: to select a title of a top article as a tag, the title must contain a term of the tweet. These titles (as tags) are given higher scores. This technique guarantees two things. First; the article is one of the top 7 candidate articles that are most similar to the input tweet. Second; the title itself is also similar to the

input tweet by containing a term of it. Unluckily, some terms in the tweet lead to the selection of unsuitable tags.

For example, the title "**الدوري العراقي الممتاز**" appeared as the forth suggested tag for "**مباراة الهلال_الرائد تنهي بفوز الهلال بهدفين مقابل هدف، ضمن الأسبوع التاسع من دوري_المحترفين**" because it contains the term "**دوري**". Also, "**معرض إيفا برلين**" was suggested as the second tag for "**معرض جايتكس احد اكبر الاحداث في دبي في مجال تكنولوجيا_المعلومات**" since it contains the term "**معرض**". This type of weakness also drops the mean average precision because these tags appear at the top of the suggested list of tags.

7. Title does not contain a term: this is similar to the previous point, but instead of containing a term, a title that is suitable as a tag has not been selected because it did not contain any term of the input tweet. For example, the title "**دارك نت**" which is a descriptive tag for "**ديب ويب يعني الشبكة العميقة أوالإنترنت المظلم! هي مجموعة مواقع صعب توصلها و عشان كذا يسمونها عميقة** " was not selected since it does not contain any term of the tweet.

8. Other causes: may include miss-spilling, and we can define miss-spilling as wrongly written terms or missed spaces between terms that combines them. Luckily, our dataset does not contain any.

4.7 Summary

This chapter presented the evaluation of the system. And also discussed the results besides the strengths and weaknesses of the system.

We claim that there is no previous effort in short text tagging using Wikipedia in Arabic Language domain. We have formulated a dataset of 100 short texts to assess the system. Results were judged by human subjects' opinion. The results indicated that our system achieved a high relevant measures with 84.39 mean average precision and 96.53 Mean reciprocal rank.

Chapter 5
Conclusions

Chapter 5
Conclusions

In this work, we have developed a tag recommender system for short Arabic texts by exploiting Arabic Wikipedia as a base knowledge. Given a short Arabic text, the system compares it to the Wikipedia articles in the concept space to find the most relevant and articles then uses these articles to suggest ranked tags from their titles and categories.

The system process consists of the following steps: First, configuring Arabic Wikipedia: in this step the XML dumb is parsed for complete articles; body, titles and categories. Then text preprocessing is applied including, cleansing, segmenting and stop-word removal. Second, preparing the system: this step constructs the Tf-idf matrix then the Singular Value Decomposition. Third, in this step the system compares the input to the articles in the concept space to find the most similar ones. Forth, electing tags: this step is to select tags from the titles and the categories of relevant articles. The category selection is based on the intersection, while the title selection depends on containing a term of the input text, these tags are ranked using a simple ranking procedure.

The tag recommender system is evaluated over 100 short texts from online Arabic tweets in three different subject. The results of the system were evaluated by experts' subject opinion. Then the system is assessed based on the evaluation metrics of mean average precision, mean reciprocal rank. Results indicated that the system achieved high relevance measures with 83.39 mean average precision and 96.53 mean reciprocal rank.

This work has the following research contributions:

To our knowledge, this is the first work to explore the Arabic short text tagging using Arabic Wikipedia. Arabic Wikipedia has only been exploited recently by the Arab computer researchers and few efforts from the literature have tried to extend to the Arabic version of Wikipedia for different purposes such as determining relations

between topics (Kanan et al., 2015) and named entity recognition(Althobaiti, Kruschwitz, & Poesio, 2014) but not the tag recommendation.

Our work proposed a simple ranking procedure that is especially designed for ranking results in our case. This is different from other ranking algorithms, but in our humble opinion the system can be used in other application such as suggesting links in "Read More" section that offers documents similar to the current document in the same website. Also, the system, as it is, can be employed for auto categorization of Wikipedia articles.

Our system, is one of few works that utilize latent semantic analysis to non-Latin languages compared to Latin languages. These works, including ours, proof the possibility of employing LSA to achieve high performances.

As far as we know, most works utilize LSA to summarize documents or to find similarities between existing documents. This work is one of a scarce to confirm the applicability of introducing new document to the system.

The results show that the system help mapping poorly composed short texts into real life concepts that can help improve other information retrieval processes. Also it helps unifying tags among users which can improve classification and linking by providing more insight to the content and the meaning (purpose) of the short text.

We proposed an in-depth evaluation of our tag recommender and explored the potential shortcomings and strengths of each involved process. This detailed evaluation can inform Arab researchers with the various options and recommendations for designing similar approaches.

For the uniqueness of this work, we have some aims for the future:

1. Evaluate the system in the field of question answering. Dealing with Arabic Wikipedia as the source knowledge and the question as a short text, the system must provide one article, at best, that contains the answer of the question.

2. Exploit the latent semantic analysis of Arabic Wikipedia for other applications such as finding similarity between Arabic documents or recommender systems.
3. Explore solutions for the weakness points discussed in Section 4.6. For example, results can be improved by unifying the way of writing foreign words in Arabic.
4. Proof the generality of the tag recommender by Appling it to the English Wikipedia.

References

References

Abdeen, M., & Tolba, M. F. (2010). Challenges and design issues of an Arabic web crawler. *Computer Engineering and Systems (ICCES), 2010 International Conference on, 4*(1), 203-206.

Abdelali, A., Darwish, K., Durrani, N., & Mubarak, H. (2016). Farasa: A fast and furious segmenter for arabic. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, 11-16.

Agnihotri, L., Mojarad, S., Lewkow, N., & Essa, A. (2016). Educational data mining with Python and Apache spark: a hands-on tutorial. *Proceedings of the Sixth International Conference on Learning Analytics & Knowledge*, 507-508.

Al-Shalabi, R., Kanaan, G., Jaam, J. M., Hasnah, A., & Hilat, E. (2004). Stop-word removal algorithm for Arabic language. *Proceedings of 1st International Conference on Information and Communication Technologies: From Theory to Applications, CTTA, 4*, 19-23.

Allahyari, M., & Kochut, K. (2016a). Semantic Tagging Using Topic Models Exploiting Wikipedia Category Network. *2016 IEEE Tenth International Conference on Semantic Computing (ICSC)*, 63-70.

Allahyari, M., & Kochut, K. (2016b). *Semantic Tagging Using Topic Models Exploiting Wikipedia Category Network.* Paper presented at the 2016 IEEE Tenth International Conference on Semantic Computing (ICSC).

Althobaiti, M., Kruschwitz, U., & Poesio, M. (2014). Automatic Creation of Arabic Named Entity Annotated Corpus Using Wikipedia. 106-115.

Baxla, M. A. (2014). *Comparative study of similarity measures for item based top n recommendation.* National Institute of Technology Rourkela.

Bernotas, M., Karklius, K., Laurutis, R., & Slotkienė, A. (2015). The peculiarities of the text document representation, using ontology and tagging-based clustering technique. *Information Technology And Control, 36*(2).

Bhowmik, R. (2008). Keyword extraction from abstracts and titles. *IEEE SoutheastCon 2008*, 610-617.

Bogers, T., & Van den Bosch, A. (2008). Recommending scientific articles using citeulike. *Proceedings of the 2008 ACM conference on Recommender systems*, 287-290.

Bosagh Zadeh, R., Meng, X., Ulanov, A., Yavuz, B., Pu, L., Venkataraman, S., . . . Zaharia, M. (2016). Matrix Computations and Optimization in Apache Spark. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 31-38.

CoreNLP, S. (2016, Dec 23, 2016). Stanford CoreNLP. Retrieved February 4th, 2017, 2017, from http://nlp.stanford.edu/software/stanford-arabic-corenlp-2016-10-31-models.jar

Dafney, J. A. J., & Mary, A. L. (2014). Semantic Analysis of tags for the Social Classification of Resources in Tagging Systems. *International Journal, 4*(3), 105-111.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science, 41*(6), 391-407.

Fallows, J. (2007). Tag teams. *The Atlantic*, Jan/Feb: 163-165.

Gabrilovich, E., & Markovitch, S. (2006). Overcoming the brittleness bottleneck using Wikipedia: Enhancing text categorization with encyclopedic knowledge. *AAAI, 6*, 1301-1306.

Garcia Esparza, S., O'Mahony, M. P., & Smyth, B. (2010). Towards tagging and categorization for micro-blogs. *Paper presented at the 21st National Conference on Artificial Intelligence and Cognitive Science (AICS 2010), Galway, Ireland, 30 August-1 September, 2010*.

Github. (Wikixmlj 2016, Seb 5, 2016). wikixmlj wikiParser. Retrieved Feb 9, 2017, 2017, from https://github.com/delip/wikixmlj

Gittens, A., Devarakonda, A., Racah, E., Ringenburg, M., Gerhardt, L., Kottaalam, J., . . . Chhugani, J. (2016). Matrix Factorization at Scale: a Comparison of Scientific Data Analytics in Spark and C+ MPI Using Three Case Studies. *arXiv preprint arXiv:1607.01335*, 1-26.

Golder, S. A., & Huberman, B. A. (2006). The structure of collaborative tagging system. *Journal of information science, 32*(2), 198-208.

Gong, Y., & Liu, X. (2001). Generic text summarization using relevance measure and latent semantic analysis. *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, 19-25.

Guo, W., Li, H., Ji, H., & Diab, M. T. (2013). Linking Tweets to News: A Framework to Enrich Short Text Data in Social Media. *ACL (1)*, 239-249.

HaCohen-Kerner, Y. (2003). Automatic extraction of keywords from abstracts. *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, 843-849.

Han, J., Pei, J., & Kamber, M. (2011). *Data mining: concepts and techniques*: Elsevier.

Hassan, M. M., Karray, F., & Kamel, M. S. (2012). Automatic document topic identification using wikipedia hierarchical ontology. *Information Science, Signal Processing and their Applications (ISSPA), 2012 11th International Conference on*, 237-242.

Hulth, A. (2003). Improved automatic keyword extraction given more linguistic knowledge. *Proceedings of the 2003 conference on Empirical methods in natural language processing*, 216-223.

Ionescu, B., Popescu, A., Lupu, M., Gînscă, A. L., Boteanu, B., & Müller, H. (2015). Div150cred: A social image retrieval result diversification with user tagging credibility dataset. *Proceedings of the 6th ACM Multimedia Systems Conference*, 207-212.

Jeong, W. (2009). Is tagging effective?–overlapping ratios with other metadata fields. *International Conference on Dublin Core and Metadata Applications*, 31-39.

Kanan, T., Ayoub, S., Saif, E., Kanaan, G., Chandrasekarar, P., & Fox, E. A. (2015). Extracting Named Entities Using Named Entity Recognizer and Generating Topics Using Latent Dirichlet Allocation Algorithm for Arabic News Articles (pp. 1-22): Department of Computer Science, Virginia Polytechnic Institute & State University.

Khoja, S. (2001). Arabic Stemmer. Retrieved january. 4th, 2017, from http://zeus.cs.pacificu.edu/shereen/ArabicStemmerCode.zip

Kywe, S. M., Hoang, T.-A., Lim, E.-P., & Zhu, F. (2012). On recommending hashtags in twitter networks. *International Conference on Social Informatics*, 337–350.

Laclavik, M., Šeleng, M., Ciglan, M., & Hluchý, L. (2012). Ontea: Platform for pattern based automated semantic annotation. *Computing and Informatics*, *28*(4), 555–579.

Li, Z., Zhou, D., Juan, Y.-F., & Han, J. (2010). Keyword extraction for social snippets. *Proceedings of the 19th international conference on World wide web*, 1143-1144.

Liu, T.-Y. (2009). Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, *3*(3), 225-331.

Mei, Q., & Zhang, Y. (2008). Automatic web tagging and person tagging using language models. *International Conference on Advanced Data Mining and Applications*, 741-748.

Meng, X., Bradley, J., Yuvaz, B., Sparks, E., Venkataraman, S., Liu, D., . . . Owen, S. (2016). Mllib: Machine learning in apache spark. *JMLR, 17*(34), 1-7.

Milicevic, A. K., Nanopoulos, A., & Ivanovic, M. (2010). Social tagging in recommender systems: a survey of the state-of-the-art and possible extensions. *Artificial Intelligence Review, 33*(3), 187-209.

Moss, L., Shaw, M., Piper, I., Hawthorne, C., & Kinsella, J. (2016). Apache Spark for the Analysis of High Frequency Neurointensive Care Unit Data: Preliminary Comparison of Scala vs. R.

O'Neil, T., & Sangiovanni-Vincentelli, A. L. (2014). Automatic construction and ranking of topical keyphrases on collections of short documents. 398-406.

Oliveira, V., Gomes, G., Belém, F., Brandao, W., Almeida, J., Ziviani, N., & Gonçalves, M. (2012). Automatic query expansion based on tag recommendation. *Proceedings of the 21st ACM international conference on Information and knowledge management*, 1985-1989.

Otsuka, E., Wallace, S. A., & Chiu, D. (2014). Design and evaluation of a twitter hashtag recommendation system. *Proceedings of the 18th International Database Engineering & Applications Symposium*, 330-333.

QCRI. (2016, 2016). Farase Arabic Segmenter. Retrieved Feb 9, 2017, from http://alt.qcri.org/farasa/segmenter.html

Ramudu, B., & Murty, M. N. (2012). Topic based semantic clustering using Wikipedia knowledge. *Data Science & Engineering (ICDSE), 2012 International Conference on*, 1-7.

Ryza, S., Laserson, U., Owen, S., & Wills, J. (2015). *Advanced Analytics with Spark: Patterns for Learning from Data at Scale*: " O'Reilly Media, Inc.".

Santoso, J., Nugraha, J. N., Yuniarno, E. M., & Hariadi, M. (2015). Noun ontology generation from Wikipedia article using Map Reduce with pattern based approach. *Intelligent Technology and Its Applications (ISITIA), 2015 International Seminar on*, 373-378.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. *Proceedings of the 10th international conference on World Wide Web*, 285-295.

Schönhofen, P. (2009). Identifying document topics using the Wikipedia category network. *Web Intelligence and Agent Systems: An International Journal, 7*(2), 195-207.

Shapira, B., Ofek, N., & Makarenkov, V. (2015). Exploiting wikipedia for information retrieval tasks. *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1137-1140.

Sidorov, G., Gelbukh, A., Gómez-Adorno, H., & Pinto, D. (2014). Soft similarity and soft cosine measure: Similarity of features in vector space model. *Computación y Sistemas, 18*(3), 491-504.

Singhal, A., & Srivastava, J. (2013). SEMANTIC TAGGING FOR DOCUMENTS USING 'SHORT TEXT'INFORMATION. 337-350.

snowballstem. (2016, Jan 27, 2016). SnowBall Stemmer. Retrieved February 4th, 2017, 2017, from https://github.com/snowballstem/snowball/tree/master/algorithms

Sprak, A. (2016). Apache Spark (Version 2.0.0): Apache. Retrieved from http://spark.apache.org/

Sun, M., Chen, Y.-N., & Rudnicky, A. I. (2017). HELPR: A framework to break the barrier across domains in spoken dialog systems *Dialogues with Social Robots* (pp. 257-269): Springer.

Symeonidis, P., Nanopoulos, A., & Manolopoulos, Y. (2008). Tag recommendations based on tensor dimensionality reduction. *Proceedings of the 2008 ACM conference on Recommender systems*, 43-50.

Symeonidis, P., Nanopoulos, A., & Manolopoulos, Y. (2010). A unified framework for providing recommendations in social tagging systems based on ternary semantic analysis. *IEEE Transactions on Knowledge and Data Engineering, 22*(2), 179-192.

Tang, J., Hong, M., Li, J., & Liang, B. (2006). Tree-structured conditional random fields for semantic annotation. *International Semantic Web Conference*, 640-653.

Tonella, P., Ricca, F., Pianta, E., & Girardi, C. (2003). Using keyword extraction for web site clustering. *Web Site Evolution, 2003. Theme: Architecture. Proceedings. Fifth IEEE International Workshop on*, 41-48.

Turney, P. D. (2000). Learning algorithms for keyphrase extraction. *Information retrieval, 2*(4), 303-336.

Turney, P. D. (2001). Mining the web for synonyms: PMI-IR versus LSA on TOEFL. *European Conference on Machine Learning*, 491-502.

Wang, P., Hu, J., Zeng, H.-J., & Chen, Z. (2009). Using Wikipedia knowledge to improve text classification. *Knowledge and Information Systems, 19*(3), 265-281.

Wikipedia. (2016, 2016 August 2nd). Statistics. Retrieved September 30th, 2016, from https://meta.wikimedia.org/w/index.php?title=List_of_Wikipedias/ar&uselang=ar

Wikipedia. (2017, 2017-01-01). arwiki dump progress on 20170101. Retrieved january. 1st, 2017, from https://dumps.wikimedia.org/arwiki/20170101/

wikixmlj. (2016, sep 5, 2016). wikixmlj wikiParser WikiSense project. Retrieved Feb 9,2017, 2017

Yeh, J.-Y., Ke, H.-R., Yang, W.-P., & Meng, I.-H. (2005). Text summarization using a trainable summarizer and latent semantic analysis. *Information processing & management, 41*(1), 75-95.

Yih, W.-t., Goodman, J., & Carvalho, V. R. (2006). Finding advertising keywords on web pages. *Proceedings of the 15th international conference on World Wide Web*, 213-222.

Zadeh, R. B., Meng, X., Yavuz, B., Staple, A., Pu, L., Venkataraman, S., . . . Zaharia, M. (2015). linalg: Matrix computations in apache spark. *arXiv preprint arXiv:1509.02256*, 1-14.

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). Spark: cluster computing with working sets. *HotCloud, 10*, 10-10.