بسم الله الرحمن الرحيم

الجامعة الإسلامية – غزة
The Islamic University - Gaza

عمادة الدراسات العليا **هاتف داخلي**: 1150

ج س غ/35/
الرقم Ref 2012/09/18م
التاريخ Date

نتيجة الحكم على أطروحة ماجستير

بناءً على موافقة عمادة الدراسات العليا بالجامعة الإسلامية بغزة على تشكيل لجنة الحكم على أطروحة الباحث/ **منار سعيد عبد المحسن فياض** لنيل درجة الماجستير في **كلية *تكنولوجيا المعلومات*** **برنامج تكنولوجيا المعلومات** وموضوعها:

Automatic Arabic Domain-Relevant Term Extraction

وبعد المناقشة التي تمت اليوم الثلاثاء 02 ذو القعدة 1433هـ، الموافق 2012/09/18م الساعة الحادية عشرة صباحًا، اجتمعت لجنة الحكم على الأطروحة والمكونة من:

د. ربحي سليمان بركة	**مشرفاً ورئيساً**	[illegible]
د. علاء مصطفى الهليس	**مناقشاً داخلياً**	[illegible]
د. يوسف نبيل أبو شعبان	**مناقشاً خارجيًا**	[illegible]

وبعد المداولة أوصت اللجنة بمنح الباحث درجة الماجستير في **كلية *تكنولوجيا المعلومات*/ برنامج تكنولوجيا المعلومات.**

واللجنة إذ تمنحه هذه الدرجة فإنها توصيه بتقوى الله ولزوم طاعته وأن يسخر علمه في خدمة دينه ووطنه.

والله ولي التوفيق ،،،

عميد الدراسات العليا

[illegible]

أ.د. فؤاد علي العاجز

عمادة الدراسات العليا
الجامعة الإسلامية غزة
The Islamic University Gaza

بسم الله الرحمن الرحيم

Islamic University of Gaza
Deanery of Post Graduate Studies
Faculty of Information Technology

Automatic Arabic Domain-Relevant Term Extraction

By:

Manar S. Fayyad

Supervised By:

Dr. Rebhi Baraka

A Thesis Submitted as Partial Fulfillment of the Requirements for the Degree of Master in Information Technology

Sep. 2012 - Shawwal 1433 H

Dedication

This work is dedicated to my mother and father

To my family

To my friends

To my professors and teachers

To term extraction researchers

To Muslims

Acknowledgement

First of all thanks to Allah for the gift of Islam and for guiding me in accomplishing this research.

I would like to thank my parents for their support. I extend my thanks to all my family members.

I am very thankful to my supervisor, Dr. Rebhi Baraka, for his enormous support, valuable guide, and assistance throughout the work of this research.

Special thanks to all members of IT faculty at the Islamic university of Gaza for passing their knowledge to me.

Also I would like to thank every natural language processing teacher, researcher or doctor that his knowledge passed to me through his writings which guides me through my work.

Abstract

Term extraction from text corpus is an important step in knowledge acquisition and it is the first step in many Natural Language Processing (NLP) methods and computer lingual systems. In Arabic language there are some works in the field of term extraction and few of them try to extract domain-relevant terms.

In this research a model for automatic Arabic domain-relevant term extraction from text corpus was proposed. The proposed model uses a hybrid approach composed of linguistic and statistical methods to extract terms relevant to specific domains depending on prevalence and tendency term ranking mechanism.

In order to realize the proposed model a multi domain corpus separated into 10 domains (Economic, History, Education and family, Religious and Fatwa's, Sport, Health, Astronomy, Low, Stories, and Cooking recipes) was used. Then this corpus preprocessed by removing non Arabic letters, punctuations, diacritics, and stop words. Then a candidate terms vector was extracted using a sliding window with variant length dropping the windows that contain a stop word.

Candidate terms have been ranked using Termhood method as a statistical method that measures the distributional behavior of candidate terms within the domain and across the rest of the corpus.

Then Candidate terms have been distributed over the domains depending on the higher rank result for the extracted terms constructing a domain term matrix. This matrix has been used in a simple classifier that classifies the testing corpus. The final step gives us a confusion matrix that indicates that the domain term matrix worked as a best classifier achieving an accuracy rate of 100% for some domains and very good in others. The total accuracy of the classifier was 95%. This is a highly accurate classifier.

Keywords: Preprocessing, Stemming, light stemming, Arabic Term Extraction, Terms, Domain-Relevant Term Extraction.

الملخص

إن استخراج المصطلحات (Term extraction) من النصوص يشكل خطوة هامة في عملية استخراج المعرفة (Knowledge acquisition) وهي من الخطوات الأولى في كثير من عمليات معالجة اللغات الطبيعية (Natural language processing) ونظم الكمبيوتر اللغوية (Computer lingual systems) على مستوى اللغة العربية هناك العديد من الأعمال التي تعالج مسألة استخراج المصطلحات ولكن القليل منها عالج المصطلحات المرتبطة بمجال معين.

في هذا البحث تم اقتراح نموذجا آليا لاستخراج المصطلحات العربية المرتبطة بمجال معين من مجموعة نصوص عربية. النموذج المقترح يستخدم نهج هجين يتكون من الأساليب اللغوية والإحصائية لاستخراج المصطلحات ذات الصلة بمجال محدد وإسنادها إلى هذا المجال اعتمادا على انتشار هذا المصطلح داخل المجال وخارجه ومدى ارتباطه بهذا المجال.

من أجل تحقيق النموذج المقترح استخدم مكنزاً (Corpus) (مجموعة مستندات أو نصوص) عربيا مقسما إلى عشرة مجالات هي (اقتصاد، تاريخ، تربية وأسرة وإمرة، دين وفتاوى شرعية، رياضة، صحة، فلك، قانون، قصص، وصفات واكلات). ثم تم معالجة هذه المستندات معالجة لغوية سطحية (Light stemming) بإزالة الحروف غير العربية وعلامات الترقيم وحركات التشكيل والكلمات الموقوفة (Stop words) (الشائعة الاستخدام) مع الاحتفاظ بمكان الأخيرة فارغا لأنها تؤثر على استخراج المصطلحات المرشحة. من ثم قمنا باستخراج المصطلحات المرشحة باستخدام أسلوب الشباك المنزلق (Sliding window) وبأطوال مختلفة حيث تم إسقاط الشبابيك التي تحتوي على كلمات موقوفة.

بعد ذلك تم تقييم كل مصطلح من المصطلحات المرشحة (Candidate terms) بناءً على معيار وزن محدد يقيس مدى انتشار المصطلح داخل المجال المحدد وخارجه ومدى ارتباطه بهذا المجال وتتكرر هذه العملية لكل مصطلح مرشح على جميع المجالات المتوفرة لهذه التجربة.

ومن ثم تم مقارنة الأوزان المحسوبة لكل مصطلح وتخصيص المصطلح المرشح للمجال ذو الوزن الأكبر مع إهمال المصطلحات ذات الأوزان مساوية أو اقل من الصفر وكررت هذه العملية لجميع المصطلحات المستخرجة من المكنز وهكذا فقد تم الحصول على قائمة من المصطلحات لكل مجال تختلف عن قائمة المجالات الأخرى وأطلق على مجموعة القوائم هذه بمصفوفة مصطلحات المجالات(Domain term matrix) .

ولاختبار مدى فاعلية هذا النموذج تم استخدام هذه المصفوفة في عملية تصنيف بعض المستندات أو النصوص وتحديد مجالاتها مع العلم بان مجالاتها كانت محددة مسبقا وقد تم تصميم مصنف يعتمد على مصفوفة مصطلحات المجال وقد تم استخراج مصفوفة التشويش (Confusion matrix) لهذا المصنف وكانت النتائج ممتازة في أغلب المجالات بحيث حققت نسبة دقة بلغت ١٠٠% في بعض المجالات وجيدة جدا في بعضها الآخر. وقد بلغت نسبة الدقة الكلية ٩٥ %.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ACC	: Adjusted Contextual Contribution.
ACDW	: Average Contextual Discriminative Weight.
ANLP	: Arabic Natural Language Processing.
ATE	: Automatic Term Extraction.
BMA	: BuckWalter Morphological Analyzer.
DP	: Domain Prevalence.
DT	: Domain Tendency.
DW	: Discriminative Weight.
FA	: Field Association.
IDF	: Inverse Document Frequency.
LLR	: Log-Likelihood Ratio.
MF	: Modifier Factor.
MWE	: Multi Word Expression.
NGD	: Normalized Google Distance.
NLP	: Natural Language Processing.
OSAC	: Open Source Arabic Corpora.
P	: Precision.
PMI	: Point wise Mutual Information.
POS	: Part of speech.
POST	: Part of Speech Tagging.
R	: Recall.
TF	: Term Frequency.
TF-IDF	: Term Frequency Inverse Document Frequency.
TH	: Termhood.

Transliteration of the Arabic terms within this thesis

Arabic	English phoneme	Meaning
كتب	ktba	wrote
كاتب	katib	Writer
كتاب	ktab	Book
خط	khat	Line
أنابيب	anabayb	Pipes
غاز	ghaz	Gas
يمر	ymr	Passes
عبر	abar	Across
تركيا	torkya	Turkey
إلى	eela	To
اتحاد	athad	Union
أوروبي	awrwbai	European
ومن	wmn	It is
منتظر	muntader	Expected
أن	an	That
يكتمل	yktml	Completed
مشروع	mshrwa	Project
نابوكو	nabawkw	Nabucco
بالغ	baalgh	about
طوله	toloh	Length
كيلومترا	kylwmtra	Km
في	fia	In
عام	am	Year
بتكلفة	batklfh	Cost
تقدر	tqdr	Estimated
مليارات	mlyarat	Billion
يورو	ywrw	Euro
يطرح	yatrah	Raises
أسهم	ashm	Stocks
شركة	shrkh	Company
مدمجة	mdmjh	Merged

Chapter 1: Introduction

This chapter talks about automatic Arabic domain relevant term extraction from text corpus which is very important for natural language processing studies and applications. Firstly, we define the problem of the study and the main objective to solve this problem; and to recognize the specific objectives related to this main objective. Secondly we mention the scope and limitation of doing this research. Thirdly, the proposed methodology to achieve our objectives is clarified. Finally, we summarized the content of this research in the final paragraph.

The term is albeit provisionally definition by Sager as a constructs of human cognition processes which assist in the classification of objects by way of systematic or arbitrary abstraction [1]. He acknowledges that there exists considerable divergence of opinion in this matter and chooses to leave it more or less undefined and considered as an “axiomatic primitive, like word or sentence” [1].

Term extraction is a method that scans text to extract terminological units. It contains in order to enrich lexicographic resources. Software solutions can automate the process by scanning texts for terminological units, extracting word combinations to fulfill preset criteria and generating reports for filtering are extremely helpful because they automated a task that can otherwise be a time consuming, and costly undertaking [2].

The resulting terms maybe used in many NLP tasks such as information retrieval, text mining, document summarization etc… [3]. Any corpus participate in the term extraction process need to be preprocessed like removing no letters, removing stop word, etc… [4]. The term extraction has two main stages: Firstly, extraction of candidate terms. Secondly, validating and ranking of these terms [5].

There are several approaches for extracting candidate terms like linguistic filtering that uses linguistic patterns like (N ADJ, N N, and N PREP N) for filtering the tagged corpus [6]. Also the noun phrase which take any sequence of words following a noun can be used [5]. Other researchers uses a local grammar approach that uses a role for extracting a term like the telling role in [7]. The n-gram sliding window method could be used for extracting candidate term with n words length [8][9].

There are several ranking ways for validating the extracted term. They are classified into two categories unithood and termhood [10]. First, the unithood is the degree of strength or stability of syntagmatic combinations and collocations [11]. It is calculated only for complex terms. Some of the unithood measures are T-Score, NGD (Normalized Google Distance), mutual information, and log-likelihood. They simply relies on the occurrence and co-occurrence frequencies from domain corpora as the source of evidence [12]. Second, the termhood measures the degree to which these stable lexical units are related to domain-specific concepts like C-value, NC-value, TF/IDF, etc… [13]. Some ranking methods use both of them like Termhood (TH).

This study aims to build a model for automatic Arabic domain-relevant term extraction from multiple domains corpus. The model depends on the prevalence and tendency measures for ranking the extracted candidate term on the target domain and across the rest of the corpus. We expect to have pure domain-relevant terms matrix as an output of the model. This matrix could be helpful in classifying document, automatic library indexing, and other lingual application. Depending on the type of the corpus this model could be used in generating spam mail matrix for spam mail detection.

1.1 Problem statement

The Existing Arabic domain-relevant term extraction methods and models depend on a single domain to measure the term relevancy for specific domain. Therefore Arabic Domain-relevant Term Ranking needs to be enhanced depending on prevalence and tendency of the selected domain-relevant terms within the domain and across the irrelevant corpus. Consequently the problem in this research is how to extract domain relevant terms from Arabic text corpus to construct a domain relevant term matrix.

1.2 Objectives

Main objective

The vital purpose of this study is to develop a model for automatic Arabic domain-relevant term extraction from text corpus using several domains. The model would use linguistic methods for the term extraction, prevalence and tendency statistical technique to rank the selected terms within the domain and across the irrelevant domains. Hence forth to distribute these terms over the domains depending on their rank value to construct a domain term matrix.

Specific objectives

- To select a corpus from several domain specific corpuses, preprocess it, and construct a word vector containing tokens extracted from this corpus.
- To extract candidate terms the word vector using sliding window.
- To rank the extracted terms depending on distributional behavior (prevalence and tendency) for each term within the domain and across other domains using the Termhood method.
- To assign the extracted terms to the strongest domain and remove it from the other domains.

- To realize the model through a term extraction system and evaluate its accuracy using the precision and recall measures.

1.3 Importance of the research

To our knowledge, there exists no similar research in Arabic term extraction that combines both the linguistic as well as the statistical techniques to extract terms.

This research will assist other natural language possessing applications such as automatic translation, question answering, document classification, ontology building, etc… By introducing a domain term matrix; and a method for domain relevant term extraction.

The research will help to improve the precision and recall for domain-relevant term extraction which affects the automatic ontology learning process for Arabic language.

Arabic natural language text processing domain will benefit from this model to support Arabic knowledge management.

Extracting knowledge from text is a very challenging problem and we hope this work will help to enhance this process.

1.4 Scope and limitations of the research

Within term extraction, the research focuses on automatic term extraction with emphasize on natural language processing such as: Part-of-speech tagging and phrase chunking. We deal with Arabic language; therefore we use Arabic natural language processing to deal with Arabic corpuses.

The Arabic text corpus will be divided into certain specific domains as we are going to measure the term relevancy depending on the prevalence and tendency of the term across the domain and the rest of the corpus.

Prevalence and tendency as statistical techniques for term ranking will be used here as they are widely used and proven to be efficient especially for domain-relevant term extraction.

1.5 Methodology

We present the following methodology for carrying out the objectives of the research:

1. Build a model for automatic Arabic domain-relevant term extraction.
2. Select several domain specific corpuses.

3. Perform the suitable preprocessing like removing punctuations, Arabic diacritics, non-letters, definite articles, and stop words.
4. Construct a domain word vectors from the corpus
5. Combine the domain word vectors into one vector.
6. Extract terms from word vector using sliding window.
7. Calculate the occurrences of each term within the word vectors of the corpus and number of documents the term appears in.
8. Ranking terms depending on distributional behaviors (prevalence and tendency) of term within the target domain and also across different domains.
9. After constructing the term ranking vectors for all domains within the corpus, find intersected terms and put them in the strongest domain and remove it from other domains constructing the domain term matrix.
10. Evaluation of the accuracy and comparison of results:
    a. Evaluate the results of the examples that use the model output.
    b. Comparing the model with other models based on selected criteria.

1.6 Thesis structure

The rest of the thesis is organized as follows: Chapter 2 discusses the background of the study and the related works that have studied term extraction issues. Chapter 3 presents the detailed development of the model. Chapter 4 describes the stages of implementing the model. Chapter 5 evaluates the model depending on the implementation examples in classifying documents. Chapter 6 concludes the study and suggested future work that would be done to promote and develop the model.

Chapter 2: Background and related work

In this chapter we present the background of term extraction by defining the word term and talk about the characteristics and properties of terms. Also we define term extraction and talk about the special characteristics of Arabic language. After that we review the related work in term extraction domain and discuss methods, results, and methodologies that are applied to evaluate the necessity of our work.

2.1 Background

2.1.1 Term definitions

There are different definitions of the word term. One such a linguistic definition is; “Term is a noun or a compound word used in a specific context to give a dedicated meaning” [14]. But here we should define the term depending on the purposes of the corpus-based computational terminology extraction process which may serve like document classification, construction of ontology’s, document indices, validation of translation memories, and even classical terminology works.

Thus, the definition of term must clarify the purpose it serves. What is common to the different applications however is the need to distinguish domain-specific terms from general vocabulary [15]. Domain-specific terms are terms that have significant meaning(s) in a specific domain [16].

Terms are habitual recurrent word combinations of everyday language [17]. Terms is albeit provisional defined as “…constructs of human cognition processes which assist in the classification of objects by way of systematic or arbitrary abstraction”. He acknowledges that there exists considerable divergence of opinion in this matter and chooses to leave it more or less undefined and considered as an “axiomatic primitive, like word or sentence” [1]. In our work we define term as a sequence of word or verbs that do not contain a stop word.

2.1.2 Term characteristics

There are several characteristics for Term that should be available in terms to apply a term extraction algorithm. Those characteristics are included into two categories, Unithood characteristics which deals with terms as linguistic unit of some sort that enters into syntactic relations with other units, and Termhood characteristics which measures the degree to which a linguistic unit is related to domain-specific context [18][19]. Term characteristics are:

▪ **Linguistic properties of terms**

Some Terms are defined using a linguistic patterns that could only be applied to a corpus that has been tagged using a part-of-speech tagger in pre-processing phase [20][21].

$$((Adj|Noun) + | ((Adj|Noun) * NounPrep)?) \qquad (2.1)$$

Equation (2.1) is an example for a linguistic pattern where the pattern contains an adjective or noun which could be followed be any sequence of noun preposition sequence. Therefore it could be applied on a tagged text to extract candidate terms.

▪ **Statistical properties of terms**

The frequency of Term is the basic statistical property for Term in a corpus and generally they called Unithood of Term. The basic frequency counts are combined to compute co-occurrence measures for words. Common co-occurrence measures are the Dice similarity coefficient [22] which means the greater the frequency of term AB the bigger dice value will be. Point-wise Mutual Information (PMI) and Log-Likelihood Ratio (LLR), as they listed below in [23] and [24]. As result all these masseurs approve the relation between compound term and its components:

$$Dice = \frac{2 \times f_{AB}}{f_A + f_B} \qquad (2.2)$$

$$PMI = \log f_{AB} - (\log f_A + \log f_B) \qquad (2.3)$$

$$LLR = \log L(f_{AB}, f_A, f_B/N) + \log L(f_B - f_{AB}, \mathrm{N} - f_A, f_B/N) - \log L(f_{AB}, f_A, f_{AB}/f_A) - \log L(f_B - f_{AB}, \mathrm{N} - f_A, (f_B - f_{AB})/N - f_A) \qquad (2.4)$$

Equations (2.2, 2.3, and 2.4) are Examples of statistical proprieties of terms where f represents the frequencies of A, B terms and AB as a compound term of A and B. N is the text. L is the likelihood of choices between brackets like (f_{AB}, f_A, f_B).

Other statistical measures for overlapped terms are [25] :

- The frequency of a term candidate as a substring of another candidate.
- The frequency of a term candidate as a modifier substring or a head.
- The number of longer candidate terms of which a candidate is a part.
- The length of term |a| is the number of words in the term.

▪ **Distributional properties of terms**

There are several distributional properties of terms. First, their distribution within documents. Second, their distribution across documents in a corpus.

Third, their distribution in a domain-specific corpus as compared to their distribution in a contrastive corpus. Samples of these properties are [15]:

- *tf-idf* where *tf* stands for term frequency (in a given document) and *idf* stands for inverse document frequency measuring the spread of a term through the entire document collection.

$$idf = \log\left(\frac{N}{n_i}\right) \tag{2.5}$$

In equation (2.5) N is the number of documents for the corpus. And n_i the number of the document the term appears in.

The *tf-idf* is primarily used to rank documents, but it can also be used to rank words and word sequences of a document as term candidates.

- A simple metric that directly compares the distribution of a term in a domain specific corpus with its distribution in a general corpus is weirdness.

$$Weirdness = \left(\frac{f_D \times N_G}{f_G \times N_D}\right) \tag{2.6}$$

In equation (2.6) D is for the domain-specific corpus, G is for the general corpus, N is for corpus size, and f is for absolute frequency of terms over the domain corpus or the general corpus.

2.1.3 Term Extraction

Term extraction (which also called terminology mining, term recognition, or glossary extraction) is a subtask of information retrieval that extracts relevant terms from a given corpus using statistical like prevalence and tendency and natural language processing (NLP) methods [26][27].

As stated in Table 2.1, term-based NLP is partitioned into four sub-domains of research [14].

Table 2.1: Term based NLP domains.

	Prior terminological data	No prior terminological data
Term discovery	Term enrichment	Term acquisition
Term recognition	Controlled indexing	Free indexing

Based on this division, this thesis is concerned with term acquisition (Extraction). We should distinguish them from term checking and term spotting, which use a validated terms to search for in a set of documents.

Term extraction consists of both mono-lingual and multi-lingual term extraction, and single-word as well as multi-word terms. It is a major component in many language processing models and applications.

There are four approaches for term extraction: (a) Statistical methods which use association measures to rank MWE (Multi Word Expression) candidates. (b) Symbolic method which use morpho-syntactic patterns. (c) Hybrid methods which use both statistical measures and linguistic filters. And (d) Word alignment [28].

- **Domain relevant term extraction**

  An issue of term extraction is domain relevant term extraction which is concerned with extracting the terms relevant to specific domain. Determining the domain of terms helps to increase the performance of the classifiers that in turn increase the efficiency of knowledge retrieval. Many automatic term extraction (ATE) methods used with domain-specific document were discussed, such as TERMHOOD, UNITHOOD, C-VALUE, NC-VALUE etc... These methods are used with machine translation, summarization, question answering, and many important applications. These methods help in increasing the efficiency and accuracy of these systems.

  An overview of the general model for term extraction process is given in Figure 2.1 [26].The first, process in this figure is preprocessing and the second is term extraction and ranking the extracted terms. Then, presentation and sorting the terms. Finally, validate of terms [4]. In each stage there are several tools and approaches which could be used.

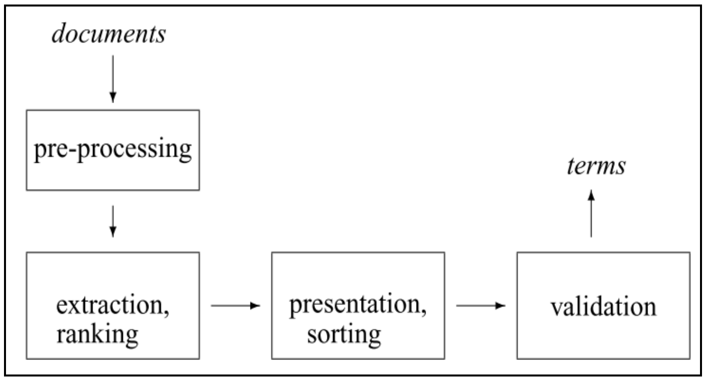


Figure 2.1: The four modules of term extraction process [26]**.**

▪ **Pre-processing**

In general term extraction model preprocessing step consist sub tasks: removing no letters, syntactic tagger tagged every input sentence from input document, and produces a list of syntactic information (Noun Phrase-NP). Removing stop words from each of the list of NP. Finally, the list of NP should be stemmed to produce list of clean NP, as the term candidate [4].

▪ **Candidate term extraction**

Detecting of term candidates is generally depends on morpho-syntactic criterion [29]. Generally, linguistic-oriented techniques rely on linguistic theories, morphological and syntactical dependency information obtained from natural language processing. Together with templates and patterns in the form of regular expressions, these techniques attempt to extract and identify term candidates. There are two common approaches for extracting term candidates. The first, requires the corpus to be tagged or parsed, and a filter is then employed to extract words or phrases satisfying some linguistic patterns. There are two types of filters for extracting from tagged corpus, namely, open or closed. Closed filters, which rely on a small set of allowable part-of-speech, will produce high precision but poor recall; On the other hand, open filters allow part-of-speech such as prepositions and adjectives will have the opposite effect. Most of the existing approaches rely on regular expressions and part-of-speech tags to accept or reject sequences of n-grams as term candidates. The second, type of extraction approach works on raw corpus using a set of heuristics. This type of approach, does not rely on part-of-speech tags, is quite rare. Such approach has to make use of the textual surface constraints to approximate the boundaries of term candidates. One of the constraints includes the use of a stop word list to obtain the boundaries of stop words for inferring the boundaries of candidates. A selection list of allowable prepositions can also be employed to enforce constraints on the tokens between units [30].

Most of these criterion are made for English language and could be applied to Arabic language but the precision of Arabic taggers are very low [31]. So I used a sliding window with length from 1 to 4 for candidate term extraction [32].

- **Ranking candidate terms**

There are several ranking methods these measures divided into two categories unithood and termhood. Unithood is defined as “the degree of strength or stability of syntagmatic combinations and collocations” [11]. and calculated only for complex terms like T-Score, NGD (Normalized Google Distance) , mutual information and log-likelihood, and rely simply on the occurrence and co-occurrence frequencies from domain corpora as the source of evidence [12]. On the other hand termhood measures the degree to which these stable lexical units are related to domain-specific concepts like C-value, NC-value, TF/IDF, etc. [13]. Some ranking methods use both of them like Termhood (TH).

- **Term ranking metrics**

There are several metrics for evaluating term extraction methods. Metric summaries and abbreviations are listed in Table 2.2. They are based on [33] and construct the metric tree in Figure 2.2. These metrics evaluates the extracted terms according to the domain or corpus and do not evaluate the terms according to the distributional behavior over the domain and across the rest of the corpus in a separated domain corpus.

Table 2.2: Metric Summary and Abbreviations [33]

Abbreviation	Metric	Rational
TF	Corpus Term Frequency	Rewards high term count, large document have advantage.
LTF	Logged Corpus Term Frequency	Minimize the effect of highly frequent terms, similar to normalization.
USN	Document Term Frequency	Reward word that appears lots In one document.
ED	Evenly Distributed	All documents contribute same number of terms.
BD	Favor Big Documents	Reward for large document.
NTF	Normalized Term Frequency	Rewards high term count but negates large document skewing.
DR	Document Relativized	Less reward for large documents penalizes verbose documents.
CR	Corpus Relativized	Less reward for large documents.
DRDA	Document Relativized-Document Average Frequency	Less reward for large documents.
CRDA	Corpus Relativized-Document Average Frequency	Less reward for large documents.

TFIDF	Term Frequency and Inverse Document Frequency	Reward terms that are in few documents, but that appear frequently.
LTFIDF	Term Frequency and logged Inverse Document Frequency	Flattens distribution of document frequency, making outlier less powerful.
DC	Distribution Consensus	Reward terms that occur in the same frequency in multiple documents.
BC	Binary Consensus	Reward Consensus, reward minimum frequency of one.

A few multi domain metrics found in the literature one of them is a Termhood (TH) that measures distributional behaviors within the target domain and also across different domains as statistical evidence to quantify the linguistic evidences in the form of candidate, modifier, and context for the term membership to a domain [10].

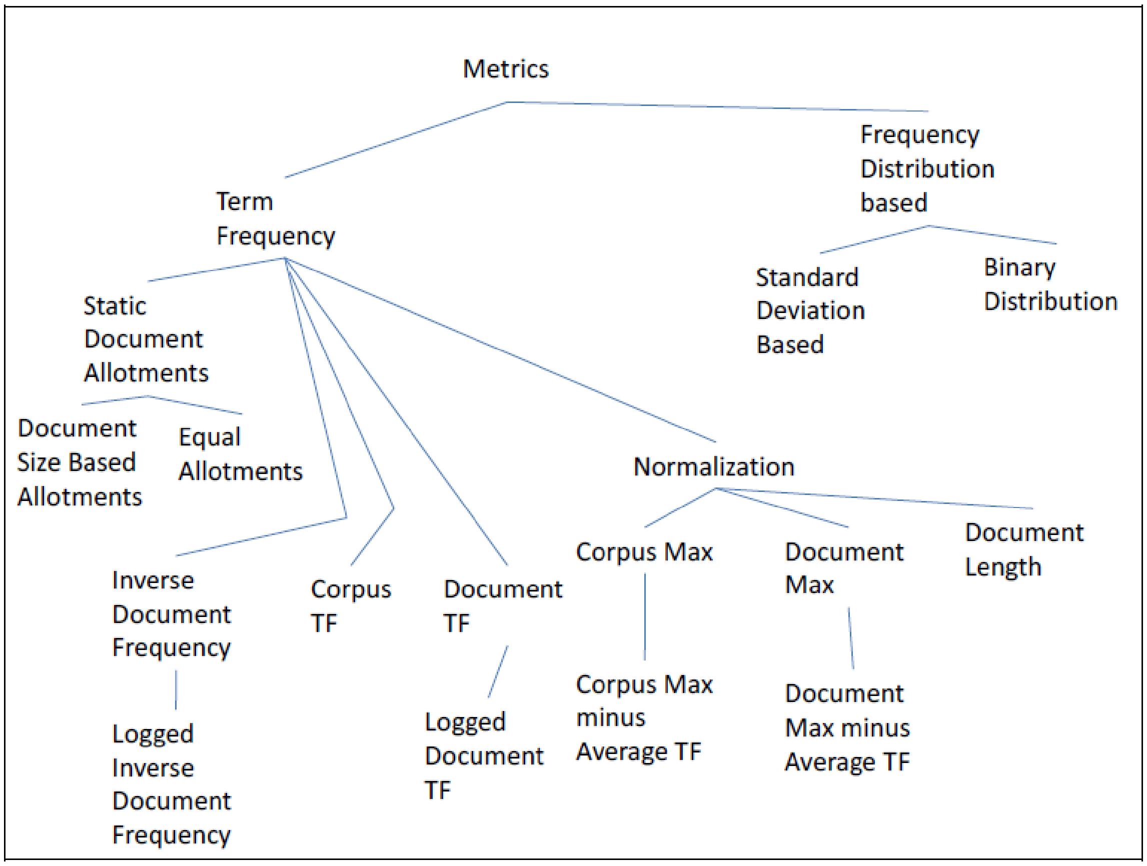

Metrics
Term Frequency
Frequency Distribution based
Static Document Allotments
Standard Deviation Based
Binary Distribution
Document Size Based Allotments
Equal Allotments
Normalization
Inverse Document Frequency
Corpus TF
Document TF
Corpus Max
Document Max
Document Length
Logged Inverse Document Frequency
Logged Document TF
Corpus Max minus Average TF
Document Max minus Average TF

Figure 2.2: Metric Hierarchical Ordering [33]

Although there are a lot of advantages for ATE such as machine translation which helps the Arabic reader to benefit from the English content on the web, there are few works for Arabic language and there is a need to increase this work to support the Arabic users and the Arabic content in the Internet.

2.2 Arabic language

- **Importance of Arabic language**

Arabic language is the first language for majority of the Arabic countries and the second language for Islamic countries. The language distinct them from countries in other regions, and it is also a language manifest in their faith, and is the religious language of all Muslims of various ethnicities around the world. It is a Semitic language with 28 alphabet letters. Its writing orientation is from right-to-left. Arabic is also considered one of the six official languages of the United Nations and the mother tongue of more than 330 million people. The Arabic Quran which means 'the recital' or the proclamation was revealed to Muhammad, the Prophet of Islam making the use of Arabic wider among the Muslims, those who profess Islam [34].

- **Difficulties of Arabic language**

A lot of difficulties and special issues face the automation of domain-relevant term extraction from Arabic corpuses; for instance, at the level of language processing there are issues that need to be addressed such as: short vowels, absence of capital letters, affixations (for example infixes, suffixes, prefixes, etc…). The Arabic has two genders, feminine and masculine, three cardinality, singular, dual, and plural. At the level of Part of Speech Tagging (POST) there are issues that need to be addressed such as complex morphology related to nouns, verbs and particles. Arabic is also highly inflectional and derivational, which makes morphological analysis a very complex task. Also Arabic has three grammatical cases, nominative, genitive, and accusative. Arabic noun is determined by its gender, cardinality, and grammatical case [34][35][36].

Arabic is a challenging language for a number of reasons [37]:

- Orthographic (الإملاء) with diacritics is less ambiguous and more phonetic in Arabic, certain combinations of characters can be written in different ways.
- Arabic language has short and long vowels which give different pronunciation. Grammatically they are required but omitted in written Arabic texts.
- Arabic has a very complex morphology as compare to English language.
- Synonyms are widespread. Arabic is a highly inflectional and derivational language.
- Lack of publically freely accessible Arabic Corpora.
- Lack of Arabic digital contents.

▪ **Issues to be solved in this thesis related to Arabic language**

Removing the definite article (ال وال بال كال فال الـ ال لل) from the word.

Removing the diacritics (- ◌ّ ◌ْ ٍ ◌ِ ٌ ُ ً ◌َ).

Removing stop words.

Remove punctuations.

2.3 Related Work

A lot of work in the field of domain-relevant term extraction is done in non-Arabic languages. For example ExATOLLP [38] is a software that extracts domain-relevant terms of syntactic annotated corpus which is a software tool that uses both linguistic and statistical approaches to extract and select significant terms from a domain represented by the annotated corpus. The system starts by extracting the noun phrases form xml documents and count the iteration of each phrase and save a list of them [38].

Also a high-performing technique for automatic extraction of shared terminology from available documents in a given domain is designed in [39] named as TermExtractor. It identifies relevant terms based on two steps: First, a linguistic processor is used to parse text and extract typical terminological structures, like compounds, adjective-noun and noun preposition noun sequences. Then, the list of terminological candidates is purged according to, domain pertinence, domain consensus, lexical cohesion, structural relevance, and miscellaneous filters to give a list of terms.

The aim of this study is to construct a model for automatic Arabic domain-relevant term extraction from corpus. For the Arabic language several works is available for term extraction, but little work is done in the domain-relevant term extraction. A few approaches for single domain as well as for multiple domains automatic term extraction is done. These works mostly use what is called Field Association (FA) to classify terms related to a specific domain [40]. The pre-processing step is very important in the Arabic language; because it is highly inflectional. Moreover special stemmer is designed depending on the topic of the research and the methods that are used. In information retrieval light stemming is widely used to keep the information value within the terms and words [41][42][34].

In building a word vector, [43] designs and implements a system for building an Arabic lexicon with 96% accuracy. The stemming process they use is likely more accurate. Other light stemmer approaches like the tested in [44] have low results, and the tool proposed by [37] could be merged with Al-Shalabis tool to enhance the preprocessing stage we will try to test several preprocessing methods to choose the best for our work.

The local grammar approach is used in [45] for the extraction of persons names from the Arabic financial news. It is a way of describing syntactic restrictions of certain subsets of sentences, which are closed under some or all of the operations in the language. They define some rules (see Figure 2.3) which are based on that the subject argument of the class of verbs known as reporting verbs (RVs) it must refer to a person [46][47]. This approach is not efficient for the term extraction as there are no rules for all the terms in the Arabic language. But it could be used as a part of the system for the future developers.

On the other hand, [48] for extract multi-word terms they use the N ADJ, N1 N2 and N1 PREP N2 patterns; and the ranks of the extracted term-like units according to their domain representatives.

A multi-word term extraction program for Arabic language is designed in [48]. They take into consideration the linguistic specifications of Arabic word like, graphical, Inflectional, morpho-syntactic and syntactic variants. They rank the multi word term like (MWT-like) units by means of statistical techniques, log-likelihood ratio (LLR), FLR, Mutual Information (MI3) and t-scores.

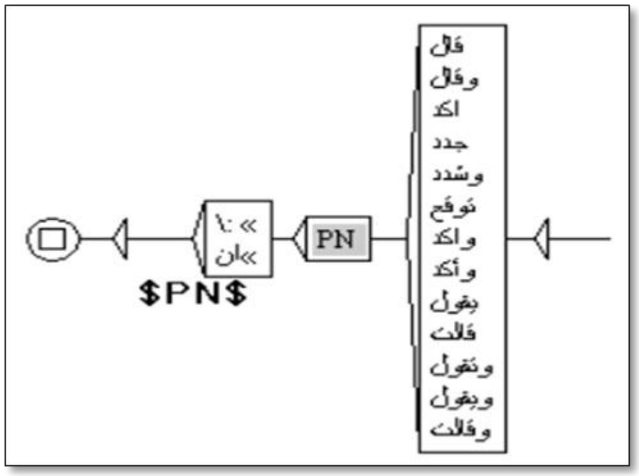


Figure 2.3: Local grammar rule for reporting verbs [45]

So in the term candidate extraction process they select patterns in Table 2.3 and we think this selection limits the probability of covering all the term forms in the corpus although this reduces the computational time. They work with one domain corpus and use a single domain ranking methods which could affect the prevalence and tendency of the extracted terms to the domain [49].

Table 2.3: Patterns and Part Of Speech mapping [48]**.**

MWT Pattern	Part of speech pattern
N1 N2	NN [P]? | NNs [P]?
N1 ADJ	NN [P]? | NNs [P]? | JJ
N! PREP N2	NN [P]? | NNs [P]? | IN | NN [P]? | NNs [P]?

In Table 2.3 N stands for noun, ADJ stands for adjective and PREP stands for preposition.

Also Khalid AI Khatib and Amer Badarneh [42] propose a two steps approach for extracting candidate MWEs: First, using a POS (Part of Speech) linguistic filter to extract candidate MWTs then using a bigram compound noun patterns(see Figure 2.4). Second, they assign each candidate MWT a score depending on the combination of both the C-value ranking method and the log likelihood ration (LLR) ranking method [50][51][52].

In their work they use Khoja stemmer which is a root extraction stemmer that removes the informational value of the token or word within the text. Also, he used a

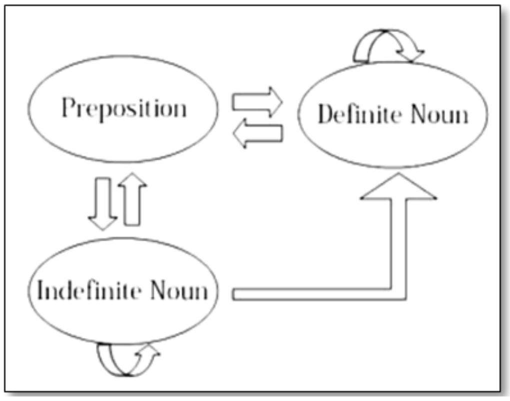


Figure 2.4: Graphical model of bigram syntactic pattern [42]

Rule Based Approach for Tagging Non Vocalized Arabic Words which has its own stemmer and concentrate on specified text; beside, he works with on domain that could affect the resulting domain terms. They also use a bi-gram term length, and one domain ranking methods.

A new methodology in [40] is used for building extensive Arabic dictionary using linguistic methods to extract relevant compound as well as single Field Association (FA) Terms from domain-specific corpora using Arabic POS as shown in Figure 2.5.

The system in Figure 2.5 consists of a part-of-speech (POS) tagger, a FA Terms candidate extraction module, a weighting module for candidate terms, selecting the relevant FA Terms, and appending them to the FA Terms dictionary. In their work they use a sliding window with 10 tokens, but they extract only terms matches the selected patterns.

Furthermore they depend on the referral corpus to rank the term to a specific domain and the results were obtained separately for the nine domains. They do not take into their consideration the distributional behavior of the term over the other domains. Their methodology is tested using their method on 14 domains using 251 MB of domain specific corpora from Wikipedia and Alhyah news giving recall and precision results around 84 percent and 79 percent respectively [40].

A new waiting function is presented in [53] for increasing the first ranked field association terms using declinable words and concurrent words which relate to narrow association categories and eliminate FA word ambiguity by weighting according to the degree of importance of concurrent words.

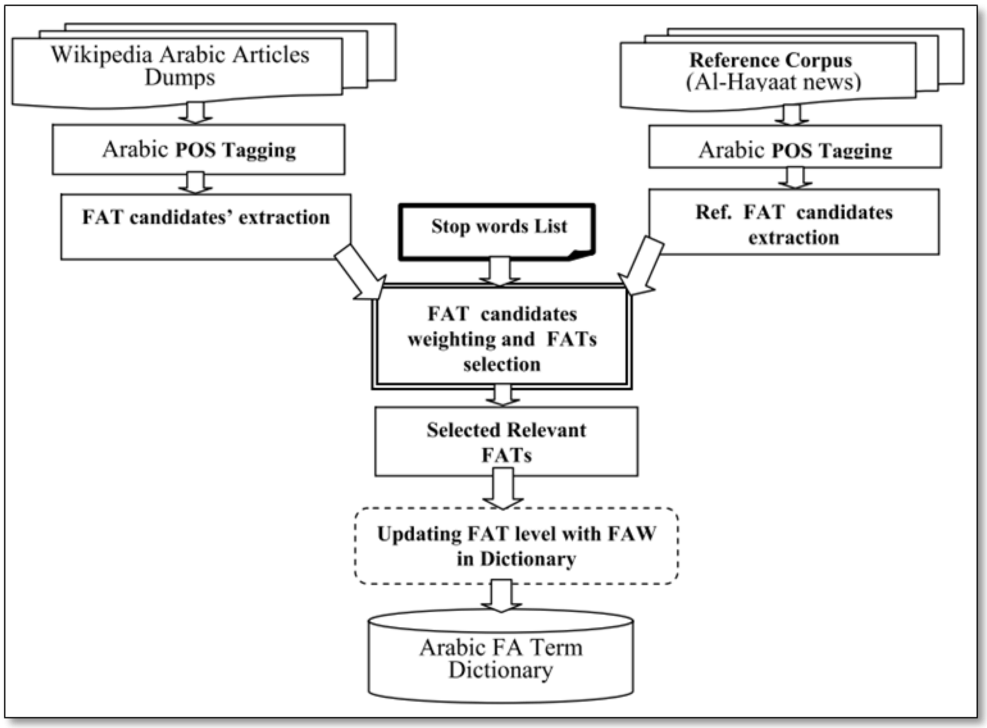

Wikipedia Arabic Articles Dumps
Reference Corpus (Al-Hayaat news)
Arabic POS Tagging
Arabic POS Tagging
FAT candidates' extraction
Stop words List
Ref. FAT candidates extraction
FAT candidates weighting and FATs selection
Selected Relevant FATs
Updating FAT level with FAW in Dictionary
Arabic FA Term Dictionary

Figure 2.5: System outline of the FA Terms selection methodology [40]

Three proposed complementary approaches to extract MWEs in [28] is implemented:

a) A cross lingual correspondence asymmetries which relied on the correspondence asymmetries between Arabic Wikipedia titles and titles in 21 different languages.
b) Translation-based extraction which collects English MWEs from Princeton Word Net 3.0, translates the collection into Arabic using Google Translate, and utilizes different search engines to validate the output.

c) Lexical association measures to extract MWEs from a large unannotated corpus.

They mention that the identification of MWEs is too complex to be dealt with by one simple solution; but also here the researchers concentrate on general term extraction and not the domain of the terms. Using the heeders of wiki articles limits the number of terms that could be extracted to evaluate; and the direct translation from other language is not suitable for the Arabic language as it is highly inflectional and has a lot of synonyms.

A model for automatic Collocation Extraction is proposed by [6]. Collocation is "A word combination whose semantic and/or syntactic properties cannot be fully predicted from those of its components and which therefore has to be listed in a lexicon". They use the following structural patterns of Arabic collocation (N+N, N+ADJ, V+N, V+ADV, ADJ+ADV, ADL+N), then, they used the joint tagging and segmenting algorithm that used for Arabic tagging by [31] and produced a bigram collocation depending on POS and previous patterns. Then, they selected four association measures (LLR, X^2, Mutual Information (MI), Enhanced Mutual Information (EMI)), and they found that the log-likelihood ratio clearly outperforms the other association measures. In their work they are also strict themselves with the patterns that limits the number of collocation that could be extracted. They eliminate the terms with low frequencies (see Table 2.4) which could be more representative for specific domain than others; and they work on general corpus with no domains that ignore the distributional behavior of the term over the domain and across the other domains.

Table 2.4: The number of candidate pairs in collocations [6]**.**

Patterns	Freq>10	Freq<10
Noun + Noun	1284	53726
Noun + Adjective	1651	31888
Noun + Verb	286	8521
Verb + Adverb	251	6523
Adjective + Adverb	365	7852
Adjective + Noun	985	9564
Collocation	5092	150534

A new weighting method for terms is proposed by [13] for multi domain corpus that employs distributional behaviors of term candidates within the target domain; and also across different domains as statistical evidence their method consists of a series of base and derived measures for recognizing terms. The base measures, namely, domain prevalence (DP) and domain tendency (DT) capture the statistical evidence that appear in the form of intra-domain and cross-domain term distributional behavior. Using these base measures, four additional measures, namely discriminative weight (DW), modifier factor (MF), average contextual discriminative weight (ACDW), and adjusted contextual contribution (ACC) were derived to quantify linguistic evidences in the form of candidates, modifiers and context words. Together, these base and derived measures contribute to the computation of a final weight known as Termhood (TH) that is used for the ranking of candidates and selection of terms.

The mechanism for scoring and ranking candidate terms by employing distributional behaviors within the target domain and also across different domains as statistical evidence to quantify the linguistic evidences in the form of candidate, modifier and context is applied on English documents only [13].

Most of the works reviewed above are dealing with one domain. This could give a false indicator of the relation between the term and the domain. On the other hand, the number of domains in the corpus increases the representatives of the extracted terms for the domains. The number of the domains increases the probability of the term to appear in several domains and competition of the domains for the term increases. Moreover these works depend on dedicated patterns for extracting candidate terms. This could exclude a large number of terms that might have a significant relation to the domain. They use ranking methods that quantify the term depending on one domain. These approaches for term candidate ranking might be inappropriate for multi domain corpus. Ranking candidate terms depends on both domain and cross domain validates the distributional behavior effect as a linguistic evidence for the term membership in a domain.

Chapter 3: Designing the Model of Term Extraction

In this chapter we design the model that serves our objective in this study and explain the rationale behind our choices to develop the model. The design beginning with corpus selection and the specification of the selected corpus, the preprocessing tasks determining which process suitable for our model, the methods for term extraction that increase the accuracy of our model, determining the best ranking method to evaluate the term weight, and finally choosing the term distribution process to assign a term to a domain.

3.1 The primitive model

The overall primitive model can be summarized in the following steps:

1. Preprocessing
2. Term extraction
3. Iteration counting.
4. Term candidate ranking process.
5. Ranked term distribution over the domains process.

The overall model architecture is shown in Figure 3.1.

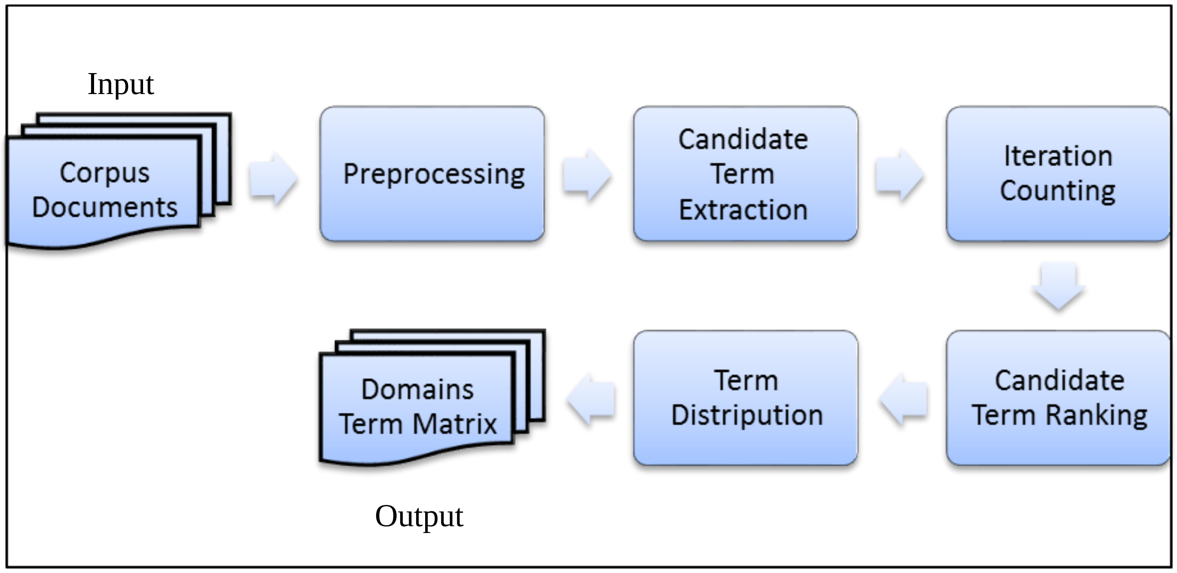


Figure 3.1: General model architecture

The Term extraction model begins by choosing the text corpus which should contain several domains. This corpus is then tokenized. For each token we apply preprocessing and add the resulting token to the word vector. Preprocessing step includes removing punctuation, diacritics, non-letters and stop words. If the extracted token is blank we add the blank to the word vector because it is important for term extraction.

After that, we use a sliding window with lengths from one to four that slides over the resulting word vector and add the extracted term to the term candidate vector. If the term extracted by the sliding window contains blank we do not add it to the term candidate vector. Simultaneously, we count the term iteration over the domain and the number of the documents the term appears in for each domain and save the resulting vector to files.

Next, we use the saved statistics for ranking each domain term candidate to the domain and do this for all the domains. The ranking method measures the distributional behavior of the candidate term over the domain and across the rest of the corpus.

Finally, we compare the ranking value for candidate term over the entire domain and save the term to the vector of the domain with the highest rank value. On the whole, we present these steps in more details.

3.2 Corpus selection stage

The model should extract the domain relevant terms from Arabic corpus so it needs to handle a corpus with the following properties:

1. A big corpus that could give a good distributional behavior for the terms.
2. The corpus should be separated into domains.
3. It should be gathered from several sources.

There are several corpora on the Internet which have been used for term extraction and we will review them depending on the above properties:

The corpus gathered by [54] is quite big (800 Mb), contains 113 million words and taken from newspaper sites but it is not separated into domains [1].

Tashkeela (Arabic diacritics) is an Arabic vocalized texts corpus contains 6 million words, 122 Mb compressed taken from books from Al-Shamela library. Its size is acceptable but it is not separated into domains[2].

[1] http://aracorpus.e3ra-b.com/argistestsrv.nmsu.edu/AraCorpus/

[2] http://sourceforge.net/projects/tashkeela

ALWatan&AlKhaleej corpus was gathered by [55] from Alwatan and Alkhalej newspapers. It's about 14 Mb size. It is separated into six domains. This corpus is from one source and it could be some bias[3].

Another corpus we have reviewed is the corpus gathered by [56] from Arabic newspapers. This corpus is separated into 14 domains but comparing to its size 3.27 Mb. it's small and we are not sure that it will clarify the real distributional behavior of the extracted terms; but, we could use this corpus for testing and evaluation[4].

Finally, we examine the OSAC (Open Source Arabic Corpora) [35] corpus which is gathered from a specific domain sites and some newspapers, this corpus is[5]:

1. A big corpus (18 Mb) that could give a good distributional behavior for the terms.
2. The corpus is separated into 10 domains.
3. It is gathered from several sources.

The size of this corpus is sufficient to characterize its domains. On the other hand, the number of domains in the corpus increases the representatives of the extracted terms for the domains. That means the extracted terms will represent the domain.in other words the number of terms appear in more than one domain will increase and the unique terms will have more weight than others.

3.3 Preprocessing, term extraction, and iteration counting stage

The second stage in this model is preprocessing, term Extraction, and iteration counting stage. As shown in Figure 3.2 this stage consists of three processes start with preprocessing which uses light stemmer that removes diacritics, punctuations, non-Arabic letters, the definite article, and stop words. The stemmed word vector matrix then passes to the candidate term extraction process which extracts the terms from the stemmed word vector depending on a sliding window saving them to candidate term vector matrix. The term iteration and document iteration counting process counts the number of times the candidate term appears in the domain, and also counts the number of document the candidate term appears in. Next each of these steps is described and discussed in detail.

[3] http://sourceforge.net/projects/arabiccorpus/

[4] http://www.comp.leeds.a-c.uk/eric/latifa/research.htm

[5] http://sourceforge.net/projects/ar-text-mining/files/

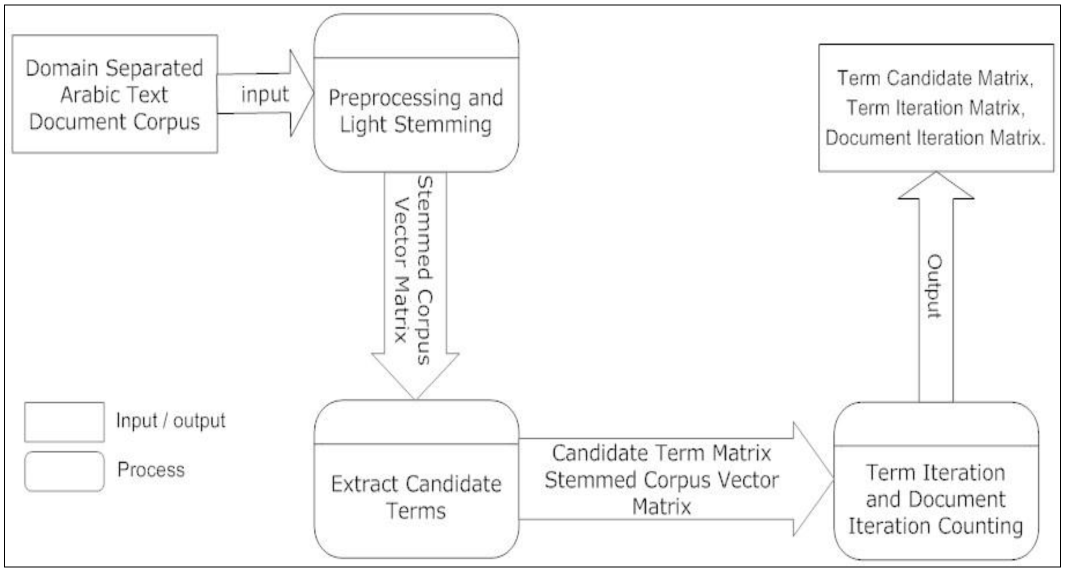

Domain Separated Arabic Text Document Corpus
input
Preprocessing and Light Stemming
Stemmed Corpus Vector Matrix
Term Candidate Matrix, Term Iteration Matrix, Document Iteration Matrix.
Output
Input / output
Process
Extract Candidate Terms
Candidate Term Matrix Stemmed Corpus Vector Matrix
Term Iteration and Document Iteration Counting

Figure 3.2: Preprocessing, term Extraction, Iteration counting Process.

3.3.1 Preprocessing

Preprocessing could be trivial process for some applications but in the Arabic language applications it is very tricky process as the Arabic language is a highly morphological language [43][47][52]. To increase the efficiency of this model a special preprocessing steps have been implemented such as definite article, the non-letter characters, diacritics and punctuation removal to increase the frequencies of word without serious effect on the meaning of the word or the term. For example when extract the root for the two words (كاتب، كتاب) it gives the root (كتب) although the Symantec of the two words are deferent.

The preprocessing is performed on a row data which is a list of folders and each folder represents a domain and contains text files encoded in UTF8. The proposed sequence of steps for the preprocessing is as follows:

1. Start with reading the folders within the corpus folder. Each folder represent a domain.
2. For each domain we read the file list within the folder.
3. For each file we read the content of the file in a vector.
4. For each word in the vector we do the following :
   A. Remove the definite article.
   B. Remove the non-Arabic letters.
   C. Remove the punctuation.
   D. Remove the diacritics.
   E. Check if the remaining word length is greater than two if yes
      i. Check if the word is not a stop word write the word to domain stemmed word vector

ii. Else add blank to the stemmed word vector.

5. Finally, write the domain stemmed word vector to a file.

The overall diagram of the preprocessing step is shown in Figure 3.3. As we see in the figure a folder reader reads the folder names and put them in a list of domain names. This reader generalizes our model to work with any number of domains.

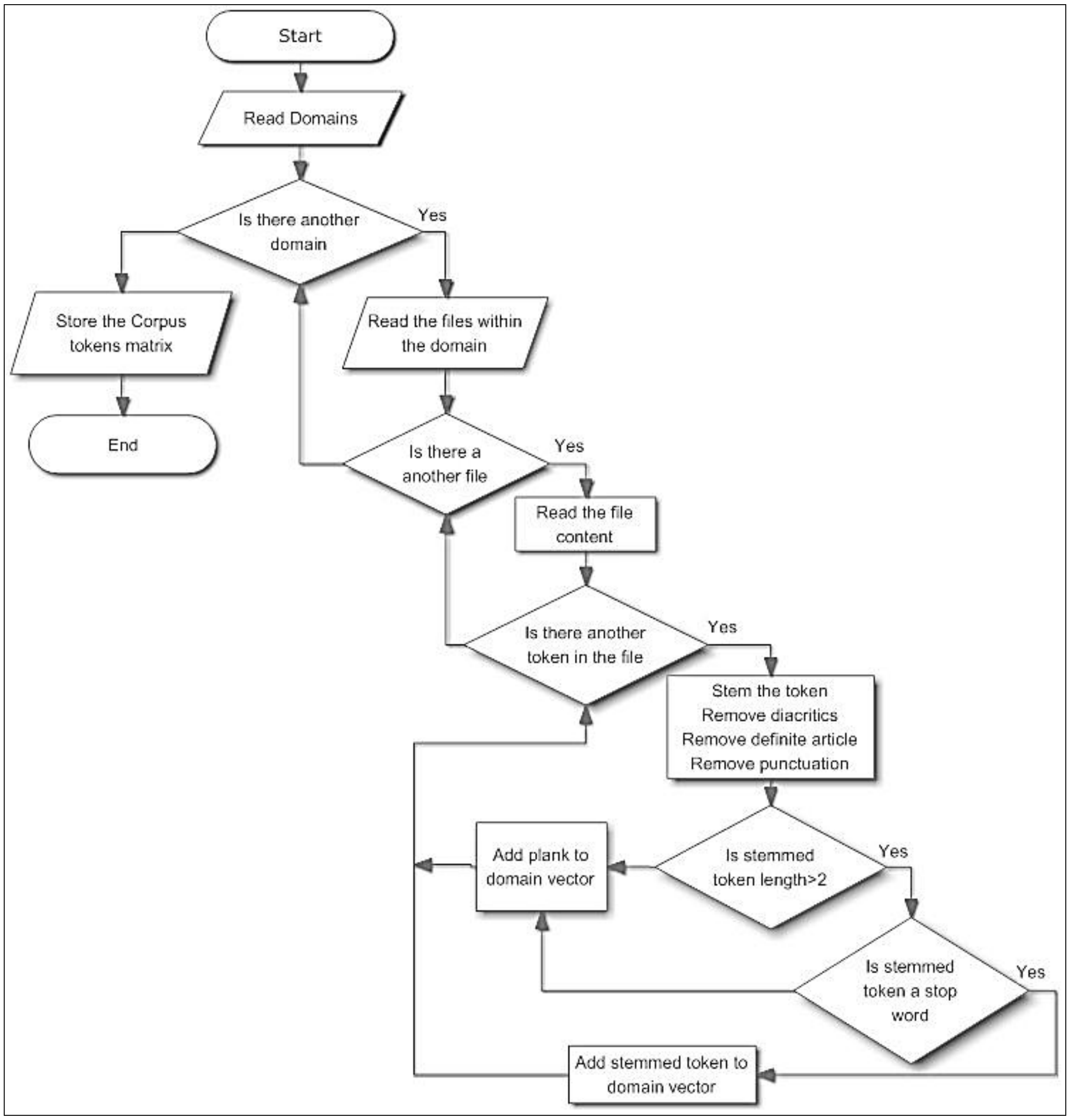

Start
Read Domains
Is there another domain
Yes
Store the Corpus tokens matrix
End
Read the files within the domain
Is there a another file
Yes
Read the file content
Is there another token in the file
Yes
Stem the token
Remove diacritics
Remove definite article
Remove punctuation
Add plank to domain vector
Is stemmed token length>2
Yes
Is stemmed token a stop word
Yes
Add stemmed token to domain vector

Figure 3.3: The overall diagram of the preprocessing phase

Then for each domain we read the list of files within the domain and construct a file list reader. Also this reader generalizes our model to work with any number of files within the domain and work with different number of files for each domain.

After that we read the contents of each file within the domain files listed in a single token vector. Add each token as an element in the vector. We use Khoja single token file reader[6].

Table 3.1: Results of preprocessing step

original text	**Remove definite article**	**Remove diacritics**	**Remove punctuation**	**Remove non Arabic letters**	**Remove stop words**
BBC	BBC	BBC	BBC		
Arabic	Arabic	Arabic	Arabic		
خَطّ	خَطّ	خط	خط	خط	خط
أنابيب	أنابيب	أنابيب	أنابيب	أنابيب	أنابيب
غَازَ	غَازَ	غاز	غاز	غاز	غاز
يُمِرُّ	يُمِرُّ	يمر	يمر	يمر	يمر
عِبَرُ	عِبَرُ	عبر	عبر	عبر	عبر
تُرْكَيَا	تُرْكَيَا	تركيا	تركيا	تركيا	تركيا
إِلَى	إِلَى	إلى	إلى	إلى	
الْاِتِّحَادِ	ْاِتِّحَادِ	اتحاد	اتحاد	اتحاد	اتحاد
الأوروبي	أوروبي	أوروبي	أوروبي	أوروبي	أوروبي
وَمِنْ	وَمِنْ	ومن	ومن	ومن	
المنتظر	منتظر	منتظر	منتظر	منتظر	منتظر
أَنْ	أَنْ	أن	أن	أن	
يَكْتَمِلَ	يَكْتَمِلَ	يكتمل	يكتمل	يكتمل	يكتمل
مَشْرُوعُ	مَشْرُوعُ	مشروع	مشروع	مشروع	مشروع
خَطَ	خَطَ	خط	خط	خط	خط
أنابيب	أنابيب	أنابيب	أنابيب	أنابيب	أنابيب
نابوكو	نابوكو	نابوكو	نابوكو	نابوكو	نابوكو
،	،	،			
الْبالِغَ	ْبالِغَ	بالغ	بالغ	بالغ	بالغ
طُولَهُ	طُولَهُ	طوله	طوله	طوله	طوله
3300	3300	3300	3300	3300	
كيلومترا	كيلومترا	كيلومترا	كيلومترا	كيلومترا	كيلومترا
،	،	،			
فِي	فِي	في	في	في	
عَامِّ	عَامِّ	عام	عام	عام	عام
2014	2014	2014	2014	2014	
بِتَكْلِفَةِ	بِتَكْلِفَةِ	بتكلفة	بتكلفة	بتكلفة	بتكلفة
تُقْدَرْ	تُقْدَرْ	تقدر	تقدر	تقدر	تقدر
ب	ب	ب	ب	ب	
7	7	7	7	7	
.	.	.			
9	9	9	9	9	
مِلْيَارَاتِ	مِلْيَارَاتِ	مليارات	مليارات	مليارات	مليارات
يورو	يورو	يورو	يورو	يورو	

6)The site for Shereen Khoja stemmer code http://zeus.cs.pacificu.edu/shereen/research.htm.

For each token we modified the Khoja stemmer to light stem each token. We need to be careful in choosing the type of stemming we use so that we do not affect the iteration counting and term extraction process.

In this stemmer we remove the definite article, none Arabic letter, diacritics, and the punctuations. Then we check if the token length is less than two letters we add blank to the stemmed word vector. After that, we check if the resulting token is a stop word we add blank to the vector if not we add it to the vector.

Stop words are very frequent tokens and do not have any effect on the results and not linked to specific text or domain so we exclude them.

Table 3.2 shows a preprocessing example of the statement:
"BBC Arabic خَطَّ أنابيب غَازَ يُمِرُّ عِبَرُ تُرْكَيَا إِلَى الْاِتِّحَادِ الأوروبي وَمِنْ المنتظر أَنْ يَكْتَمِلَ مَشْرُوعُ خَطِّ أنابيب نابوكو ، الْبالِغَ طُولَهُ ٣٣٠٠ كيلومترا ، فِي عَامِّ ٢٠١٤ بِتَكْلِفَةِ تُقْدَرْ ب ٧,٩ مِلْيَارَاتِ يورو".

The original tokens of the text are listed in the first column and the second column contain the same word vector after removing the definite article, the third contains the tokens without diacritics, the fourth contains the vector without punctuation, The fifth column shows the removal of the non-Arabic letter to give the stemmed token vector shown in the last column. For the given statement the result of the overall preprocessing is:

" خط أنابيب غاز يمر عبر تركيا اتحاد أوروبي منتظر يكتمل مشروع خط أنابيب نابوكو بالغ طوله كيلومترا عام بتكلفة تقدر مليارات".

3.3.2 Candidate term extraction

The second step in this stage is the term extraction, which begins with the merging of the resulting words vectors from the preprocessing step into one vector for each domain; so that, we could extract the terms for each domain and count the iteration on it. There are several methods for the term extraction.

For example, in morpho-syntactic patterns method (MP) the combination of n-grams words is done by following a pattern of grammatical categories, such as NA, or NPN. The MP method is a linguistic based method, and since the grammatical composition of a term determines if this term will be considered as a term.

Also the noun phrase method (NP) tries to identify n-grams annotated as a noun phrase by the parser that is, a set of n words organized around the head of a noun phrase. So, the NP method has more linguistic complexity, since it is based on full syntactic analysis of the terms.

In previous methods a tagger is needed but Arabic taggers is inaccurate as the percentage of words that not found by Buckwalter Morphological Analyzer (BMA) is about 25 percent [31].

Also they do not cover all the possible collocations in the text that could have a big effect on the extracted collocations (terms). So, we found that the n-gram method (NG) is the best method that cover all the possible collocation. N-gram extracts sequences of n words from the text and uses statistical measurements to evaluate the probability that each of the sequences has to be classified as a term, that is, the more frequently these words appear together, the higher is the chance that they can be considered a term [57][58][49].

For the term extraction in this model, we use a sliding n-gram window with one to four words length to extract the candidate terms from the domain word vector. It can be used as the length of the term that exceeded this length is statistically les stronger. The proposed algorithm for term extraction is as follows:

1. For each domain read the stemmed word vector file and do:
2. For each term length (1 to 4) do
   - A. Slide the window with term length N over the domain stemmed word vector.
   - B. For each extracted window do
     - i. If the window contains a stop word we ignore the term else add the term to a domain term vector.
3. Finally write the domain term vectors to files.

For example, statement stated in section 3.3.1 above, the resulting term vectors is shown in Table 3.2. The first column represents the stemmed word vector as an output of the previous stage. When moving a window with one word length and drop the blanks we will get the second column in the table. The same thing is done in the third, fourth, and fifth columns, but the window size is become two, three, four words length simultaneously and dropping any window that contains a blank.

3.3.3 Iteration counting

The third step is iteration counting. The kind of counting we need is related to the ranking method we intend to use for the extracted candidate terms. There are several kinds of counters and measures. The weighting method proposed by [13] for ranking a term over multi domains employs distributional behavior of term candidates within the target domain and across the rest of the corpus as statistical evidence presented in chapter two needs the following counts and frequencies to be calculated:

1. The total frequencies of all the candidate terms (F_{TC}).

2. The frequencies of a term within the domain (f_{ad}).
3. The frequencies of a term outside the domain ($f_{a\bar{d}}$).
4. The number of document the term appears in over the domain (d_{ad}).
5. The number of document the term appears in over the rest of the corpus ($d_{a\bar{d}}$).
6. The total number of term candidate (N).
7. The total number of document (D).

Table 3.2: Term extraction with 1 to 4 words length

Original text	One word term	Two words term	Three words term	Four words term
	خط	خط أنابيب	خط أنابيب غاز	خط أنابيب غاز يمر
خط	أنابيب	أنابيب غاز	أنابيب غاز يمر	أنابيب غاز يمر عبر
أنابيب	غاز	غاز يمر	غاز يمر عبر	غاز يمر عبر تركيا
غاز	يمر	يمر عبر	يمر عبر تركيا	يكتمل مشروع خط أنابيب
يمر	عبر	عبر تركيا	يكتمل مشروع خط	مشروع خط أنابيب نابوكو
عبر	تركيا	اتحاد أوروبي	مشروع خط أنابيب	
تركيا	اتحاد	يكتمل مشروع	خط أنابيب نابوكو	
	أوروبي	مشروع خط		
اتحاد	منتظر	أنابيب نابوكو		
أوروبي	يكتمل	بالغ طوله		
	مشروع	بتكلفة تقدر		
منتظر	نابوكو			
	بالغ			
يكتمل	طوله			
مشروع	كيلومترا			
خط	عام			
أنابيب	بتكلفة			
نابوكو	تقدر			
	مليارات			
بالغ				
طوله				
كيلومترا				
عام				
بتكلفة				
تقدر				
مليارات				

We use this methodology for term ranking because we are using several domain corpus and this methodology quantify the three types of linguistic evidences (Candidate evidence, Modifier evidence, Contextual evidence) derived from the prevalence and tendency measures and adjust the contribution of the contextual weight.

Our model calculate all the previous frequencies in the term extraction stage and save the results in files for each term length and domain to be retrieved in the ranking process. How we use these frequencies is discussed next in the term ranking process.

The proposed algorithm for iteration counting is as follows:

1. For each domain read extracted term candidate vectors with length (1 to 4).
2. For each term within the vector.
   - A. Count the frequencies of a term within the vector.
   - B. Count the number of document the term appears in over the domain.
3. Finally, write the domain iteration counting vector to a file.

The overall process for term extraction and iteration counting is presented in Figure 3.4. The process starts by reading the domains. For each domain we read the stemmed token vector.

Beginning with the first token we move a sliding window over the vector from the beginning to the end; and, for each window we check if the term snapped by the window contains a blank we ignore the current term if not we check if the term have been already taken; before we increase the term counter if not we add the term to the term candidate vector and check if it is the first time appear in the current document we increase the document counter. This process is repeated for each window size.

Finally, we store the term candidate, term iteration, and document iteration into files to be retrieved in the ranking stage.

In this process we exclude the windows that contain blanks these blanks replaced the stop words and other nun Arabic word in the original text.

The resulting term candidate and iteration matrix depending on the example we use in section 3.3.2 above is shown in Table 3.3. As we see in this table for each term length we count the number of times the term appear in the domain and the number of document the term appears in. We notice that the number of iteration decreases when the term length increases but the rank of the term increase as we will see in the section.

For example, the simple term (عام) iterate 8127 times over 1877 document. This means that the term is frequent over the document but this does not mean the term is domain representative. If it's frequent over the rest of the corpus is greater than in this domain this means it is not domain relevant. On the other hand, a term like (نابوكو) could be domain representative if it does not located in the rest of the corpus although it frequent in the domain is 2. As for the complex term the evidence will be calculated depending on the prevalence and tendency of the term itself and also on the head and the modifier of the term over the domain, and across the rest of the corpus. A detailed example will describe the use of these frequencies in calculating the rank value of the term in the next section.

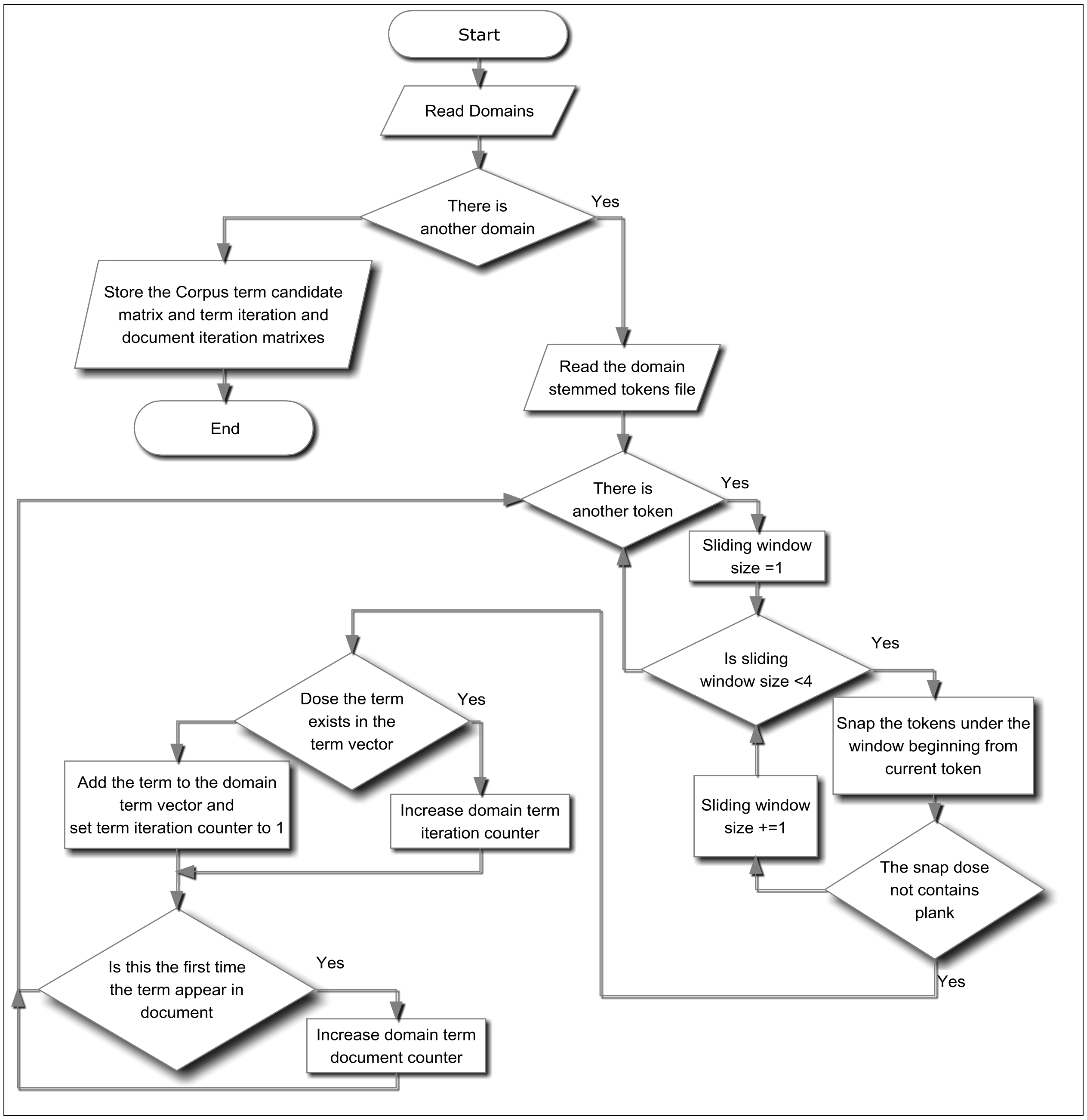

Start
Read Domains
There is another domain
Yes
Store the Corpus term candidate matrix and term iteration and document iteration matrixes
End
Read the domain stemmed tokens file
There is another token
Yes
Sliding window size =1
Is sliding window size <4
Yes
Snap the tokens under the window beginning from current token
Sliding window size +=1
The snap dose not contains plank
Yes
Dose the term exists in the term vector
Yes
Add the term to the domain term vector and set term iteration counter to 1
Increase domain term iteration counter
Is this the first time the term appear in document
Yes
Increase domain term document counter

Figure 3.4: Term candidate extraction and iteration counting

3.4 Term candidate ranking stage

Term candidate ranking is the third stage of this model. In this stage we are going to give a value for each term candidate this value will be used in the evaluation of the relevancy of the term to the domain. Then, we store these values in a matrix with two columns for each domain one for the term and the other for the rank value[59][60]. The ranking methodology used by [13] will be as follows:

The Termhood of term a ($TH(a)$) is the final ranking value of the term and as we see in equation 3.1. The rank value depends on the candidate evidence, in the form of discriminative weight of the term ($DW(a)$ Equation 3.1), and the adjusted contextual contribution of this term ($ACC(a)$ Equation 3.7) contextual evidence [13].

$$TH(a) = DW(a) + ACC(a) \qquad (3.1)$$

Table 3.3: The iteration matrix for economy domain

One word term	Term iteration	Doc iteration	Two words term	Term iteration2	Doc itteration2	Three words term	Term iteration3	Doc itteration3	Four words term	Term iteration4	Doc itteration4
خط	169	117	خط أنابيب	24	15	خط أنابيب غاز	1	1	مشروع خط أنابيب نابوكو	1	1
أنابيب	87	52	أنابيب غاز	3	3	مشروع خط أنابيب	1	1			
غاز	550	161	يمر عبر	6	4	خط أنابيب نابوكو	1	1			
يمر	59	53	عبر تركيا	3	3						
عبر	769	500	اتحاد أوروبي	470	187						
تركيا	66	50	يكتمل مشروع	1	1						
اتحاد	1287	475	مشروع خط	5	5						
أوروبي	1052	356	أنابيب نابوكو	1	1						
منتظر	64	57	بالغ طوله	1	1						
يكتمل	9	7									
مشروع	937	336									
نابوكو	2	2									
بالغ	227	191									
طوله	7	5									
كيلومترا	9	7									
عام	8127	1877									
بتكلفة	52	33									
تقدر	101	92									
مليارات	878	537									

The discriminative weight is measured using the equation 3.2. As shown in the equation, this measure depends on Cross-domain distributional behavior (domain tendency of the term $DT(a)$) and Intra-domain distribution (domain prevalence of the term $DP(a)$).

$$DW(a) = DP(a)DT(a) \qquad (3.2)$$

The domain tendency of the term is measured depending on the frequencies of a term within the domain and frequencies of a term outside the domain as shown in equation 3.3.

$$DT(a) = \log_2\left(\frac{f_{ad}+1}{f_{a\bar{d}}+1} + 1\right) \quad (3.3)$$

Where f_{ad} Is the frequencies of a term within the domain, $f_{a\bar{d}}$ is frequencies of a term outside the domain.

The domain prevalence of the term depends on the term itself for simple term (one word term) it is measured using equation 3.4 and for complex term (more than one word term) it is measured using equation 3.5. The prevalence for simple term is measured depending on the frequencies of the term over the domain and across the rest of the corpus and the total frequencies of it to the total terms iterations. On the other hand, the prevalence for complex term depends on the prevalence for the header of the term and the value of the modifier evidence of the term.

$$DPh(a) = \log_{10}(f_{ad} + 10)\log_{10}\left(\frac{F_{TC}}{f_{ad} + f_{a\bar{d}}} + 10\right) \quad (3.4)$$

$$DP(a) = \log_{10}(f_{ad} + 10)\, DPh(a)MF(a) \quad (3.5)$$

Where F_{TC} is the summation of frequencies of all the terms. f_{ad} is the frequencies of a term within the domain. $f_{a\bar{d}}$ frequencies of a term outside the domain. $MF(a)$ the modifier factor. $DPh(a^h)$ the domain prevalence of the term header.

The modifier evidence of term (in the form of modifier factor) is calculated using the equation 3.6. As shown in the equation the modifier factor depends on the summation of frequencies of all the modifiers of the term over the domain and across the rest of the corpus.

$$MF(a) = \log_2\left(\frac{\sum_{m\in M_a \cap TC} f_{md} + 1}{\sum_{m\in M_a \cap TC} f_{m\bar{d}} + 1} + 1\right) \quad (3.6)$$

Where M_a is all the modifiers of term a. and TC is all the term candidate.

The adjusted contextual contribution of the term ($\mathrm{ACC}(\mathrm{a})$) as contextual evidence is calculated using equation 3.7. From the equation we found that adjusted contextual contribution depends on the adjustment of the contextual discriminative weight and the discriminative weight itself.

$$ACC(a) = ACDW(a)\left(\frac{e^{\left(1-\frac{ACDW(a)+1}{DW(a)+1}\right)}e^{\left(1-\frac{DW(a)+1}{ACDW(a)+1}\right)}}{\log_2 \frac{ACDW(a)+1}{DW(a)+1}+1}\right) \qquad (3.7)$$

Where $ACDW(a)$ is the average contextual discriminative weight.
$DW(a)$ is the discriminative weight.

The adjusted contextual discriminative weight of the term ($\mathrm{ACDW(a)}$) is calculated using equation 3.8. From the equation we found that it depends on discriminative weight of all the context words of the term and the similarity between the term and its context words (equation 3.9).

$$ACDW(a) = \left(\frac{\sum_{c\in C_a} DW(c) * sim(a,c)}{|C_a|}\right) \qquad (3.8)$$

$$sim(a,c) = 1 - NGD(a,c) * \theta \qquad (3.9)$$

Where C_a is all the context words of term a and $|C_a|$is the number of these words. And $sim(a,c)$ is the similarity between $and\ c$. Where θ is a constant for scaling the distance value of NGD (Normalized Google Distance).

The similarity is calculated using Google normalized distance ($\mathrm{NGD(a,c)}$) equation 3.10 which depends on the number of the documents the term and its context words appear in it.

$$NGD(x,y) = \frac{\max\{\log f(x), \log f(y)\} - \log f(x,y)}{\log \mathrm{M} - \min\{\log f(x), \log f(y)\}} \qquad (3.10)$$

Where M is the total number of documents $f(x), f(y)$is the number of document x, y appears in and $f(x,y)$ is the number of document both x and y appears in.

From the previous we found that ranking method we use quantifies the three types of linguistic evidences derived from the prevalence and tendency measures in the form of Candidate evidence, Modifier evidence, and Contextual evidence. Furthermore to adjust the contribution of the contextual weight to the overall termhood they employ two measures the adjusted contextual contribution and the normalized Google distance.

3.4.1 How the ranking process work

To clarify the practical implementation for the ranking stage we will rank the following term (يطرح أسهم شركة مدمجة). Depending on the ranking methodology we discuss before the rank is given by the equation described in equation (3.1) for the domain (اقتصاد)[7].

$$TH(\text{يطرح أسهم شركة مدمجة}) \\ = DW(\text{يطرح أسهم شركة مدمجة}) + ACC(\text{يطرح أسهم شركة مدمجة})$$

$$DW(\text{يطرح أسهم شركة مدمجة}) \\ = DP(\text{يطرح أسهم شركة مدمجة})DT(\text{يطرح أسهم شركة مدمجة})$$

$$DP(\text{يطرح أسهم شركة مدمجة}) \\ = \log_{10}\left(f_{\text{يطرح أسهم شركة مدمجة}\ d} \\ + 10\right) DPh\left(\text{يطرح أسهم شركة مدمجة}^{h}\right) MF(\text{يطرح أسهم شركة مدمجة})$$

$$f_{\text{يطرح أسهم شركة مدمجة}} = 1$$

$$DPh\left(\text{يطرح أسهم شركة مدمجة}^{h}\right) = DP(\text{يطرح}\ ) = \log_{10}\left(f_{\text{يطرح}\ d} + \\ 10\right)\log_{10}\left(\frac{F_{TC}}{f_{\text{يطرح}\ d} + f_{\text{يطرح}\ \bar{d}}} + 10\right)$$

$$f_{\text{يطرح}\ d} = 42$$

$$f_{\text{يطرح}\ \bar{d}} = 193$$

$$F_{TC} = 22702550$$

$$DPh(\text{يطرح}\ ) = 45.35437395692971$$

$$MF(a) = MF(\text{يطرح أسهم شركة مدمجة}) = \log_2\left(\frac{\sum_{m\in M_a \cap TC} f_{md} + 1}{\sum_{m\in M_a \cap TC} f_{m\bar{d}} + 1} + 1\right)$$

$$MF(\text{يطرح أسهم شركة مدمجة}) = \frac{(f_{\text{أسهم}\ d} + f_{\text{شركة}\ d} + f_{\text{مدمجة}\ d} + 1)}{f_{\text{أسهم}\ \bar{d}} + f_{\text{شركة}\ \bar{d}} + f_{\text{مدمجة}\ \bar{d}} + 1} + 1$$

$$f_{\text{أسهم}\ d} = 5094$$

$$f_{\text{أسهم}\ \bar{d}} = 2798$$

[7] * The direction of reading of the terms is from left to right. and the transliteration of tem يطرح أسهم شركة مدمجة is (yatrah ashm shrkh mdmjh) an its meaning is (Raises merged stocks company)

$f_{\text{شركة } d} = 4568$

$f_{\text{شركة } \bar{d}} = 3018$

$f_{\text{مدمجة } d} = 8$

$f_{\text{مدمجة } \bar{d}} = 5$

$\mathrm{MF}(\text{مدمجة شركة أسهم يطرح}) = 1.386698584277142$

$\mathrm{DP}(\text{مدمجة شركة أسهم يطرح}) = 150.810458492347$

$$\mathrm{DT}(\text{مدمجة شركة أسهم يطرح}) = \log_2\left(\frac{f_{\text{مدمجة شركة أسهم يطرح } d} + 1}{f_{\text{مدمجة شركة أسهم يطرح } \bar{d}} + 1} + 1\right)$$

$f_{\text{مدمجة شركة أسهم يطرح } \bar{d}} = 0$

$\mathrm{DT}(\text{مدمجة شركة أسهم يطرح}) = 1.5849625007211563$

$\mathrm{DW}(\text{مدمجة شركة أسهم يطرح}) = 239.02892142693446$

$$\mathrm{ACC}(\text{مدمجة شركة أسهم يطرح}) = ACDW(a)\left(\frac{e^{\left(1-\frac{ACDW(a)+1}{DW(a)+1}\right)}e^{\left(1-\frac{DW(a)+1}{ACDW(a)+1}\right)}}{\log_2\frac{ACDW(a)+1}{DW(a)+1} + 1}\right)$$

$$ACDW(\text{مدمجة شركة أسهم يطرح}) = \left(\frac{\sum_{c\in C_a} DW(c) * sim(a,c)}{|C_a|}\right)$$

$$ACDW(\text{مدمجة شركة أسهم يطرح}) = \left(\frac{\begin{array}{l}(DW(\text{يطرح}) * sim(\text{مدمجة شركة أسهم يطرح}, \text{يطرح}) + \\ DW(\text{أسهم}) * sim(\text{مدمجة شركة أسهم يطرح}, \text{أسهم}) + \\ DW(\text{شركة}) * sim(\text{مدمجة شركة أسهم يطرح}, \text{شركة}) + \\ DW(\text{مدمجة}) * sim(\text{مدمجة شركة أسهم يطرح}, \text{مدمجة}))\end{array}}{|C_a|}\right)$$

$sim(\text{مدمجة شركة أسهم يطرح}, \text{يطرح}) = 1 - NGD(a,c)$

$$NGD(\text{يطرح أسهم شركة مدمجة}, \text{يطرح}) = \frac{\max\{\log f(x), \log f(y)\} - \log f(x,y)}{\log M - \min\{\log f(x), \log f(y)\}}$$

$$NGD(\text{يطرح أسهم شركة مدمجة}, \text{يطرح}) = \frac{\max\{\log f(\text{يطرح أسهم شركة مدمجة}), \log f(\text{يطرح})\} - \log f(\text{يطرح أسهم شركة مدمجة})}{\log M - \min\{\log f(\text{يطرح أسهم شركة مدمجة}), \log f(\text{يطرح})\}}$$

Where $f(x)$, $f(y)$ is the number of document x, y appears in.

$f(\text{يطرح أسهم شركة مدمجة}) = 1$

$f(\text{يطرح}) = 198$

$M = 17759$

$NGD(\text{يطرح أسهم شركة مدمجة}, \text{يطرح}) = 0.5404657569855092$

$sim(\text{يطرح أسهم شركة مدمجة}, \text{يطرح}) = 0.4595342430144907 5$

$sim(\text{يطرح أسهم شركة مدمجة}, \text{أسهم}) = 0.21141613920164826$

$sim(\text{يطرح أسهم شركة مدمجة}, \text{شركة}) = 0.24694890345474285$

$sim(\text{يطرح أسهم شركة مدمجة}, \text{مدمجة}) = 0.7460402536935407$

DW for one word term is calculated by the equation $DW(a) = DP(a)DT(a)$

$DW(\text{يطرح}) = 0$

$DW(\text{أسهم}) = 68.02777610376341$

$DW(\text{شركة}) = 67.49336833015924$

$DW(\text{مدمجة}) = 41.54344097198535$

$ACDW(\text{يطرح أسهم شركة مدمجة}) = 15.510665580993571$

$ACC(\text{يطرح أسهم شركة مدمجة}) = -1.81562164476214E - 5$

$TH(\text{يطرح أسهم شركة مدمجة}) = 239.02890327071802$

Then, the ranking of term يطرح أسهم شركة مدمجة over the domain اقتصاد value is: 239. If the term occurs in other domains we compare the ranking value and assign the term to the domain with large rank value.

The overall diagram for term ranking is shown in Figure 3.5. From the flowchart we found the process is begin by reading the vector of term candidate and term iteration and document iteration from the saved files for each domain; then, for each term we read the document iteration and term iteration for the term and pass these values to the ranking procedure. Then, save the rank value for the term into the rank vector. Finally saving the vectors to a file.

The term will have a rank value for each domain it appears in. An example, of the resulting rank vector values will be like displayed in Table 3.4 depending on the example we use in sections 3.3.1 above, 3.3.2 above, and 3.3.3 above. This table is a ranking matrix for one domain. In this matrix the value zero means this term is weakly relevant to this domain and any term with 0 rank value will be excluded from the term comparison between domains, the bigger the value the strongest the relation will be. That does not mean the term with big value is related to this domain. May be, this term have a larger value in other domain. Also, we can see that the more the term size increase the stronger the relation to the domain will be.

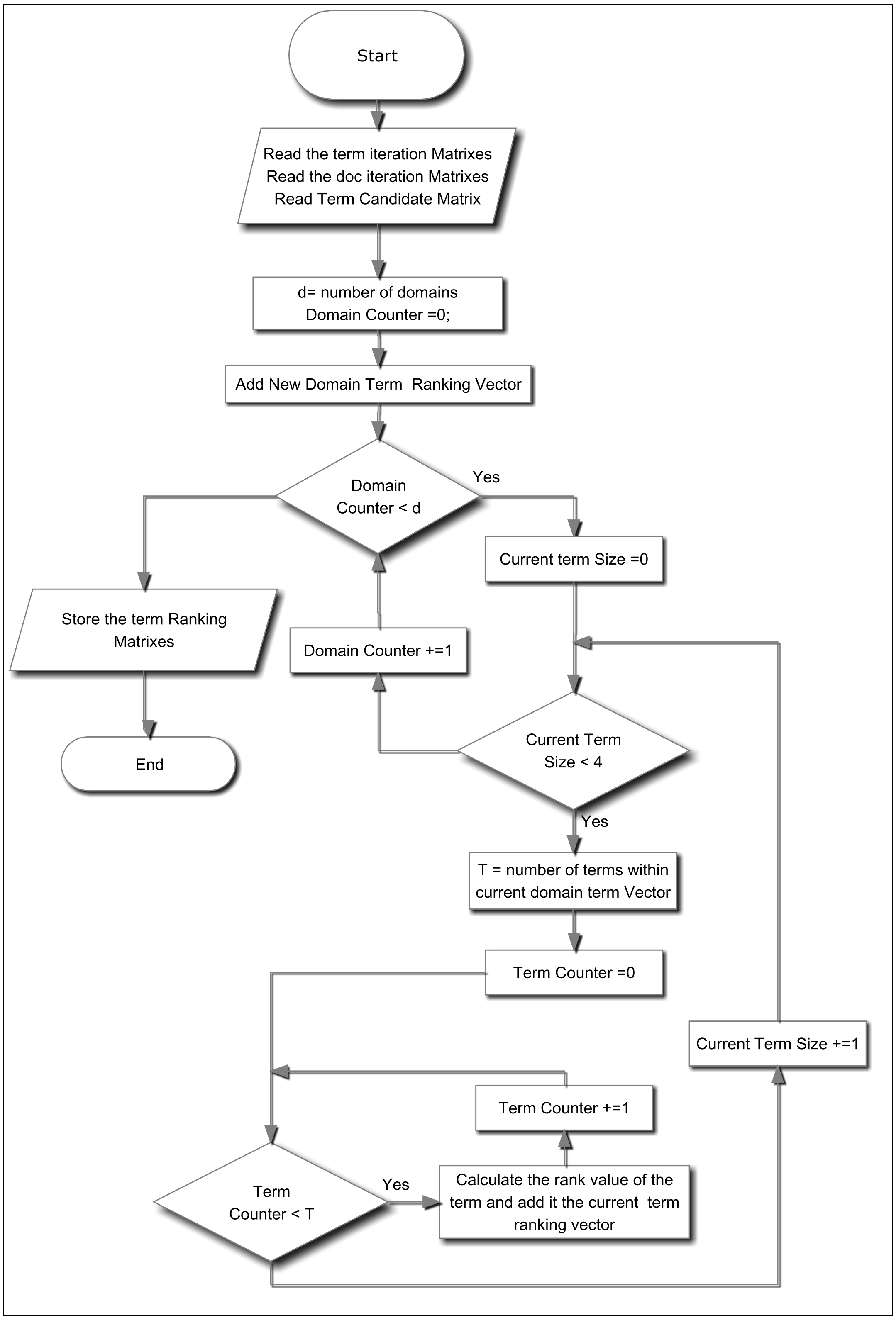

Start
Read the term iteration Matrixes
Read the doc iteration Matrixes
Read Term Candidate Matrix
d= number of domains
Domain Counter =0;
Add New Domain Term Ranking Vector
Domain Counter < d
Yes
Current term Size =0
Store the term Ranking Matrixes
Domain Counter +=1
End
Current Term Size < 4
Yes
T = number of terms within current domain term Vector
Term Counter =0
Current Term Size +=1
Term Counter +=1
Term Counter < T
Yes
Calculate the rank value of the term and add it the current term ranking vector

Figure 3.5: The flow chart of the ranking process

Table 3.4: Term ranking matrix for one domain

One word term	Rank value	Two word term	Rank value	Three word term	Rank value	Four word term	Rank value
خط	0	خط أنابيب	239	خط أنابيب غاز	165	مشروع خط أنابيب نابوكو	١٣٧
أنابيب	110	أنابيب غاز	0	مشروع خط أنابيب	137		
غاز	128	يمر عبر	0	خط أنابيب نابوكو	57		
يمر	0	عبر تركيا	84				
عبر	0	اتحاد أوروبي	471				
تركيا	0	يكتمل مشروع	0				
اتحاد	0	مشروع خط	263				
أوروبي	213	أنابيب نابوكو	318				
منتظر	0	بالغ طوله	0				
يكتمل	0						
مشروع	0						
نابوكو	161						
بالغ	0						
طوله	0						
كيلومترا	0						
عام	0						
بتكلفة	245						
تقدر	0						
مليارات	381						

3.5 Term Distribution stage

The fourth stage in this model is term distribution over the domains. This process is done by assigning each term in the candidate term matrix to a specific domain depending on the rank value of the term. It's needed to construct a matrix for domain terms to be used in a classifier for testing the accuracy of term extraction model.

In this stage we use a simple method for term distribution. If the term exists in several domains we put the term in the domain which have the highest rank value and remove it from the other domains.

Depending on the same example in section 3.4.1 above and after ranking the example term vector to the ten domains we got the Table 3.5.

Table 3.5: Comparing rank results of candidate terms from the sample over the domains

term	Rank values for the domains									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
خط	0	0	no	no	no	no	no	no	0	0
أنابيب	110	0	no	no	no	no	no	0	0	no
غاز	128	0	no	no	no	no	no	0	0	0
يمر	0	0	no	no	no	no	no	0	0	0
عبر	0	0	no	no	no	no	no	0	0	0
تركيا	0	٣٤٥	no	no	no	no	no	0	0	0
اتحاد	0	0	no	no	no	no	no	0	0	no
أوروبي	213	0	no	no	no	no	no	0	no	no
منتظر	0	0	0	161	0	no	no	0	0	no
يكتمل	0	0	0	0	0	0	no	0	0	no
مشروع	0	0	0	0	0	0	0	0	0	no
نابوكو	161	no	0	no	no	no	no	no	no	no
بالغ	0	0	0	161	128	0	0	0	0	0
طوله	0	0	0	0	0	0	0	0	0	0
كيلومترا	0	0	0	no	0	0	0	no	0	no
عام	0	0	0	0	0	0	0	0	0	0
بتكلفة	245	0	0	no	77	0	0	no	no	no
تقدر	0	0	0	0	0	0	0	0	0	no
مليارات	381	0	0	0	no	0	0	0	0	no
خط أنابيب	239	no	no	no	no	no	no	no	no	no
أنابيب غاز	0	no	no	no	no	no	no	no	no	no
يمر عبر	0	no	no	no	no	no	no	no	no	no
عبر تركيا	84	no	no	no	no	no	no	no	no	no
اتحاد أوروبي	471	no	no	no	no	no	no	0	no	no
يكتمل مشروع	0	no	no	no	no	no	no	no	no	no
مشروع خط	263	no	no	no	no	no	no	no	no	no
أنابيب نابوكو	318	no	no	no	no	no	no	no	no	no
بالغ طوله	0	0	no	no	no	no	no	no	no	no
خط أنابيب غاز	165	no	no	no	no	no	no	no	no	no
مشروع خط أنابيب	137	no	no	no	no	no	no	no	no	no
خط أنابيب نابوكو	57	no	no	no	no	no	no	no	no	no
مشروع خط أنابيب نابوكو	١٣٧	no	no	no	no	no	no	no	no	no

As shown in the table there is a rank values for domain 1 more than the other domains because the example is taken from it.

Also we notice that there are some terms do not appear in the other domains these terms are marked as (no). Other terms, there rank value is (0) this means the term is weekly relevant to the domain.

Some terms are ranked over several domain like (بتكلفة), ranked for domain 1 and domain 5. The winner domain is the domain with higher rank value.

Some domains do not rank any term of the sample although they are existing in the candidate terms of the domain. This means all the terms of the example are not related to these domains.

Terms like (تركيا) is not ranked in its original domain (domain 1) and ranked in other domain (domain2). This means the term is strongly related to the other domain.

Also we found that the complex terms (terms with two and more words) are stronger than the terms with one word to the target domain because the complex terms is less frequent the simple term.

Finally, the strongest relation between term and domain is always found in the complex terms.

The final domain term matrix for the previous sample will be as shown in Table 3.6. In this table we exclude the terms with rank value zero and the term go to the winner domain.

Table 3.6: Sample of Domain term matrix

1	2	3	4	5	6	7	8	9	10
أنابيب	تركيا		منتظر						
غاز			بالغ						
أوروبي									
نابوكو									
بتكلفة									
مليارات									
خط أنابيب									
عبر تركيا									
اتحاد أوروبي									
مشروع خط									
أنابيب نابوكو									
خط أنابيب غاز									
مشروع خط أنابيب									
خط أنابيب نابوكو									
مشروع خط أنابيب نابوكو									

Chapter 4: Realization of the Model

In this chapter we present the implementation of the model over a real corpus and the difficulties that faced the implementation of the model. We discuss some programming problems and the solution to these problems like corpus size, reading the corpus, stemmer modification, and other problems.

4.1 Component diagram

To realize the model we divide it into four main components: candidate term extraction component, term ranking component, term distribution component, and the classifier component. As shown in the component diagram in Figure 4.1, it's clear that the preprocessing component needs the corpus as input and give term candidate matrix, term iteration matrix, and document iteration matrix outputs are needed as output. These three outputs are needed to the ranking component to give the rank matrix for term distribution component. The term distribution distribute the terms and give a domain term matrix to the classifier component to classify the testing documents giving the classifier results report to evaluate the model.

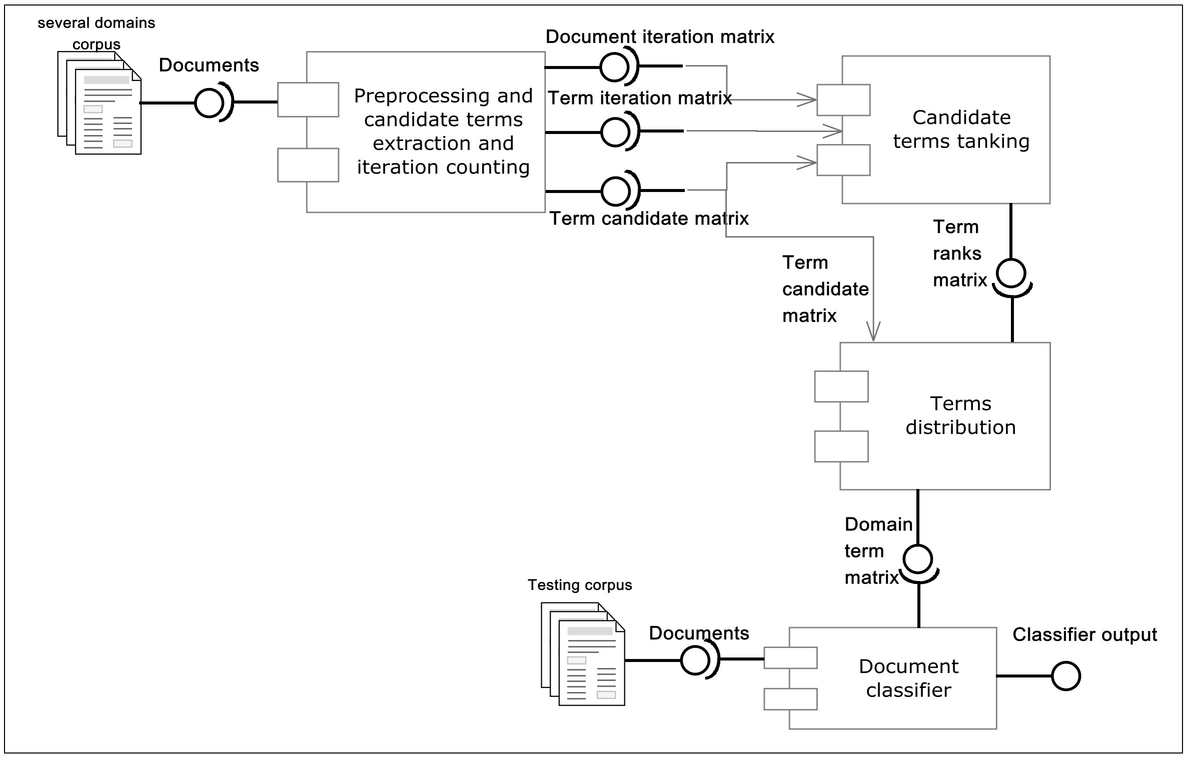


Figure 4.1: The model component diagram

Next we will discuss each component separately.

4.2 Class diagrams

- **Class diagram for term extraction and iteration counting.**

As shown in Figure 4.2 the class diagram of candidate term extraction and iteration counting is composed of 10 classes. The main class of this diagram is the StartTermCandidateExtractionProcess class. This class uses the ListOfDomain class to read the directories of the corpus and save their name as the list of domain. Then, use the ListOfFiles class to read the files within the domain directory and save them into a list of files to let the SingleTokenFileReader class read the content of the files as tokens and put them in a token vector. While reading the file the StartTermCandidateExtraction instantiate a ModifiedLightStermmer class that starts the LoadStemmerFile class to read the stemmer files and save their content to vectors that will be used in the ModifiedLightStermmer to stem each token in the token vector and then save the outputs of the stemming process into a stemmed token vector. This vector is used to extract candidate terms and count the iteration for these terms. Finally, it calls the vector to file writer to write the vectors to files.

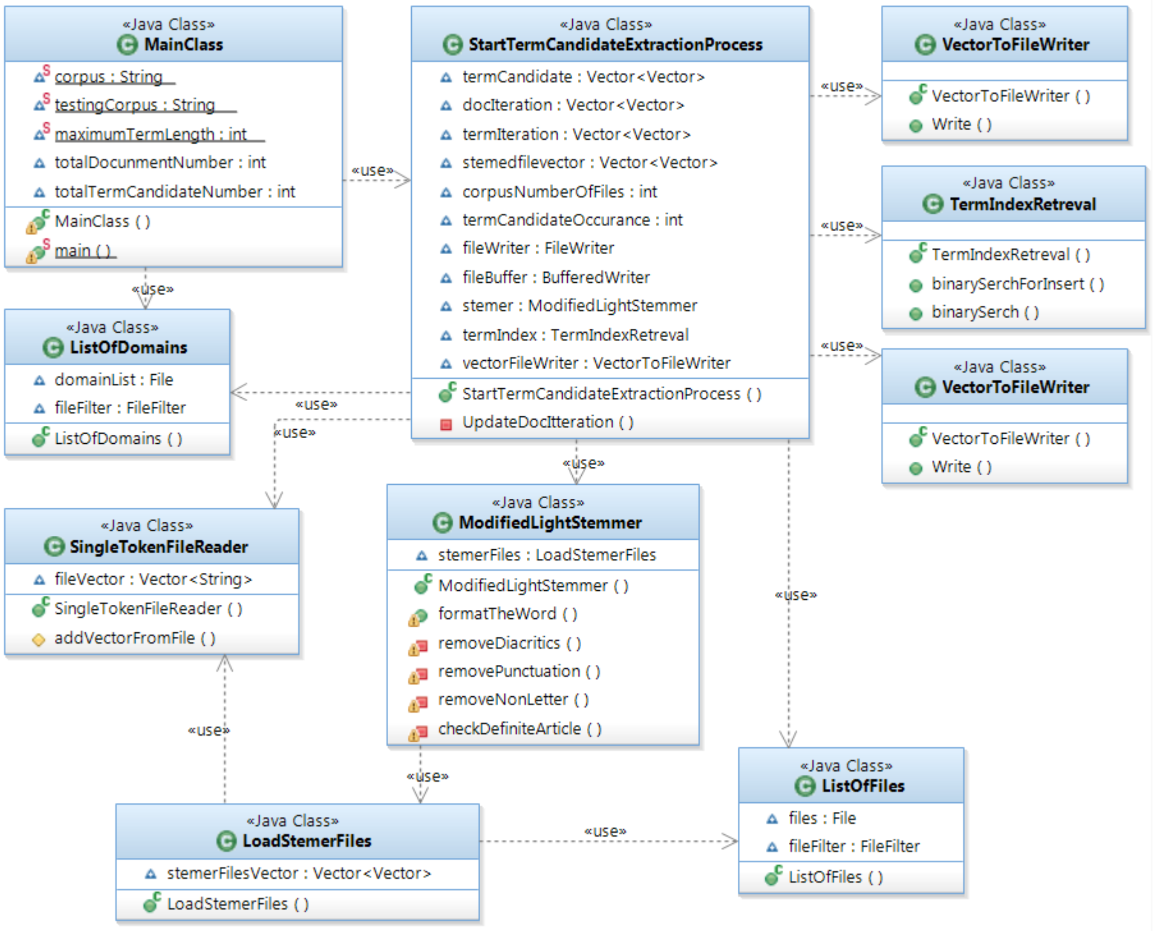

«Java Class» MainClass
corpus : String
testingCorpus : String
maximumTermLength : int
totalDocunmentNumber : int
totalTermCandidateNumber : int
MainClass ( )
main ( )
«Java Class» StartTermCandidateExtractionProcess
termCandidate : Vector<Vector>
docIteration : Vector<Vector>
termIteration : Vector<Vector>
stemedfilevector : Vector<Vector>
corpusNumberOfFiles : int
termCandidateOccurance : int
fileWriter : FileWriter
fileBuffer : BufferedWriter
stemer : ModifiedLightStemmer
termIndex : TermIndexRetreval
vectorFileWriter : VectorToFileWriter
StartTermCandidateExtractionProcess ( )
UpdateDocItteration ( )
«Java Class» VectorToFileWriter
VectorToFileWriter ( )
Write ( )
«Java Class» TermIndexRetreval
TermIndexRetreval ( )
binarySerchForInsert ( )
binarySerch ( )
«Java Class» VectorToFileWriter
VectorToFileWriter ( )
Write ( )
«Java Class» ListOfDomains
domainList : File
fileFilter : FileFilter
ListOfDomains ( )
«Java Class» SingleTokenFileReader
fileVector : Vector<String>
SingleTokenFileReader ( )
addVectorFromFile ( )
«Java Class» ModifiedLightStemmer
stemerFiles : LoadStemerFiles
ModifiedLightStemmer ( )
formatTheWord ( )
removeDiacritics ( )
removePunctuation ( )
removeNonLetter ( )
checkDefiniteArticle ( )
«Java Class» LoadStemerFiles
stemerFilesVector : Vector<Vector>
LoadStemerFiles ( )
«Java Class» ListOfFiles
files : File
fileFilter : FileFilter
ListOfFiles ( )
«use»

Figure 4.2: The candidate term extraction class diagram

When adding a candidate term the class calls the TermIndexRetreval class to find if the term is already exists; if not, it adds it to the statistics vectors and instantiate the counters of the term.

- **Class diagram for term ranking.**

The class diagram of the second component of the model described in Figure 4.3 contains the StartRankingProcess class that starts the ranking process by calling the ListOfDomains class that reads the list of domains then starts LoadStatistics class to read the data stored by the previous component then start ranking each candidate term in the list for all the domains by calling the TermRanker class. TermRanker calls TermIndexRetreval to retrieve the index of the term to be used for calling the statistics of the term for completing the rank process.

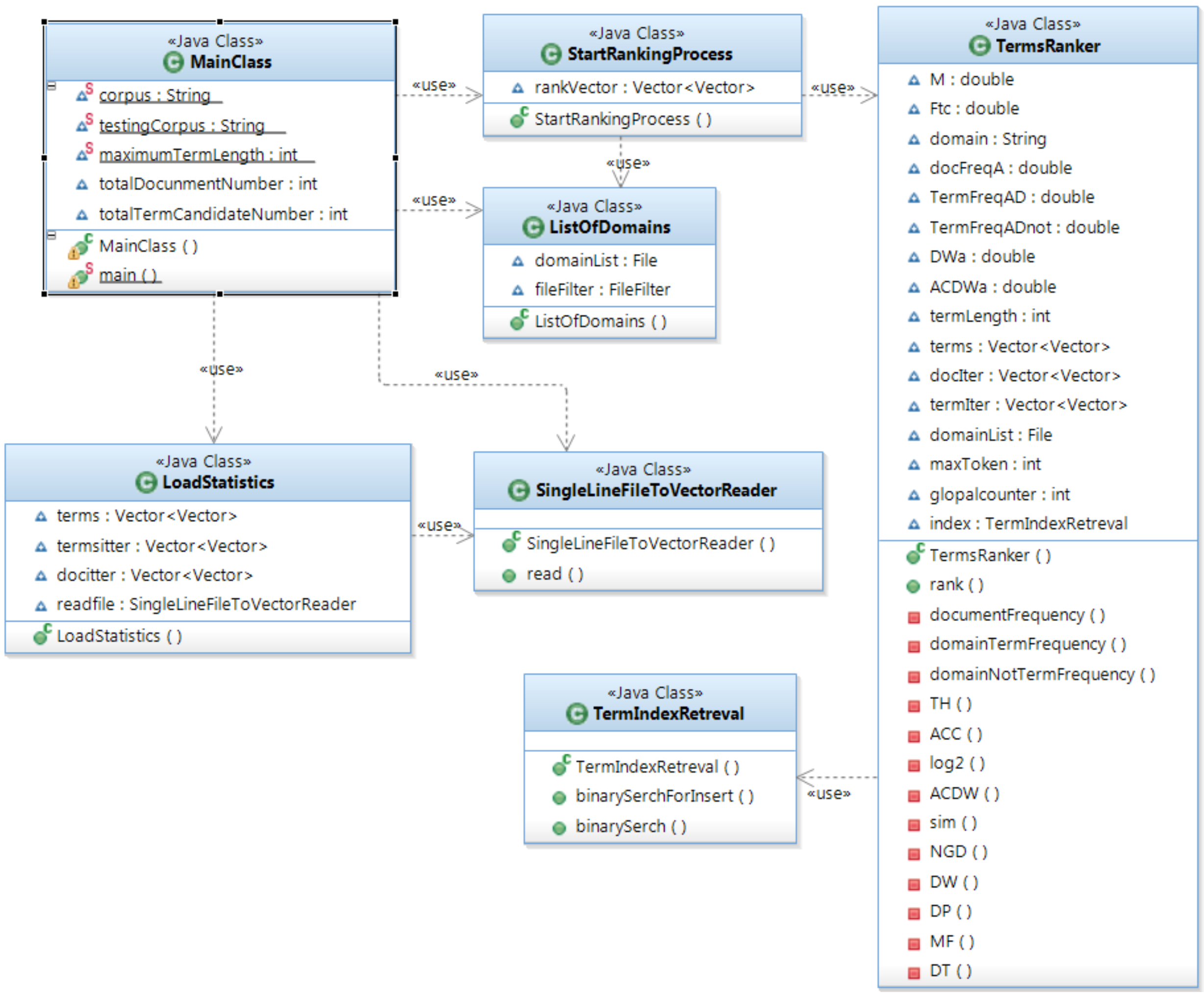


Figure 4.3: The term ranking class diagram

- **Class diagram for term distribution.**

  This class diagram describes the content of term distribution component as shown in Figure 4.4. This class starts with calling LoadRankresultsAndTerms to read the rank matrix and candidate term matrix for the entire domain by calling ListOfDomains class. Rank matrix and candidate term matrix are then passed to TermDistriputionProcess to start the candidate term distribution. At the end, TermDistriputionPr-ocess calls VectorToFileWriter to write the domain term matrix to a file to be used in the next component.

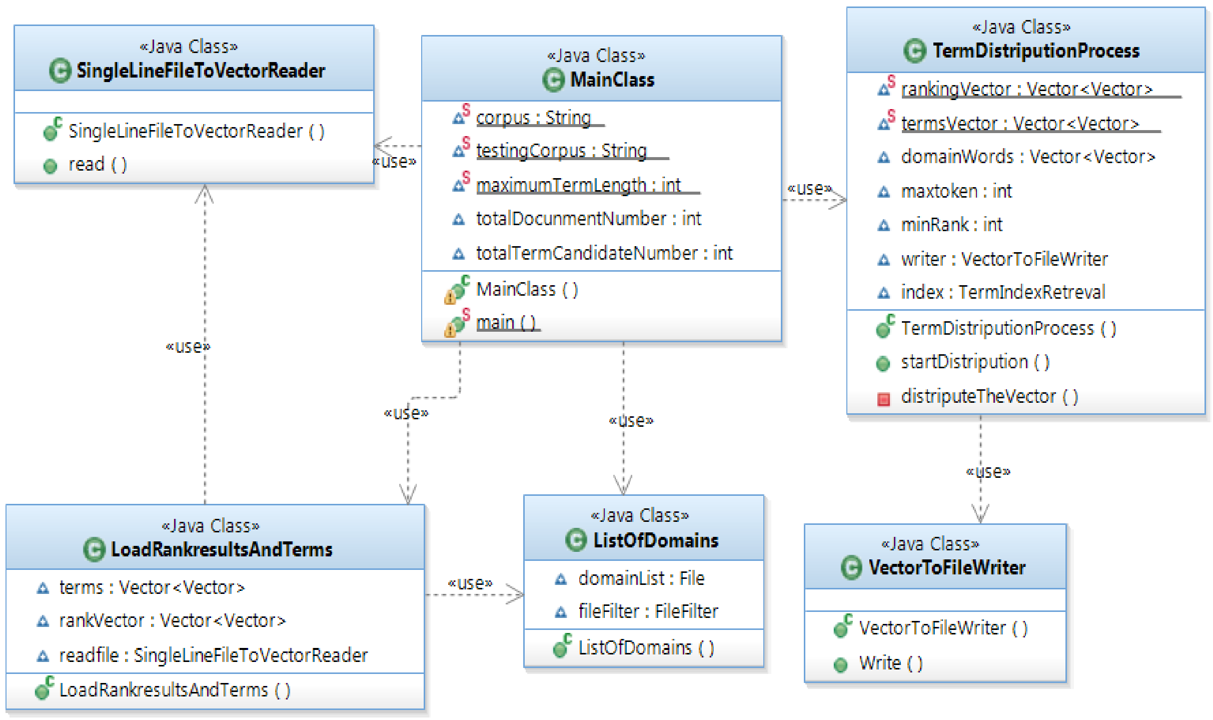


Figure 4.4: The term distribution class diagram

- **Class diagram for classifying documents.**

  The final class diagram is the classifier diagram shown in Figure 4.5. It loads the domain term matrix using the LoadDistriputedDomainWords class to be passed to TestingStage class that reads the document to be classified and there domains using three classes ListOfDomains, ListOfFiles, and VectorToFileWriter. Then passes all the data to ClassifyDocument class to be classified. It calls the ModifiedLightStemmer class to stem the document before being classified. Finally, it calls the VectorToFileWriter class to write the classify report to a file.

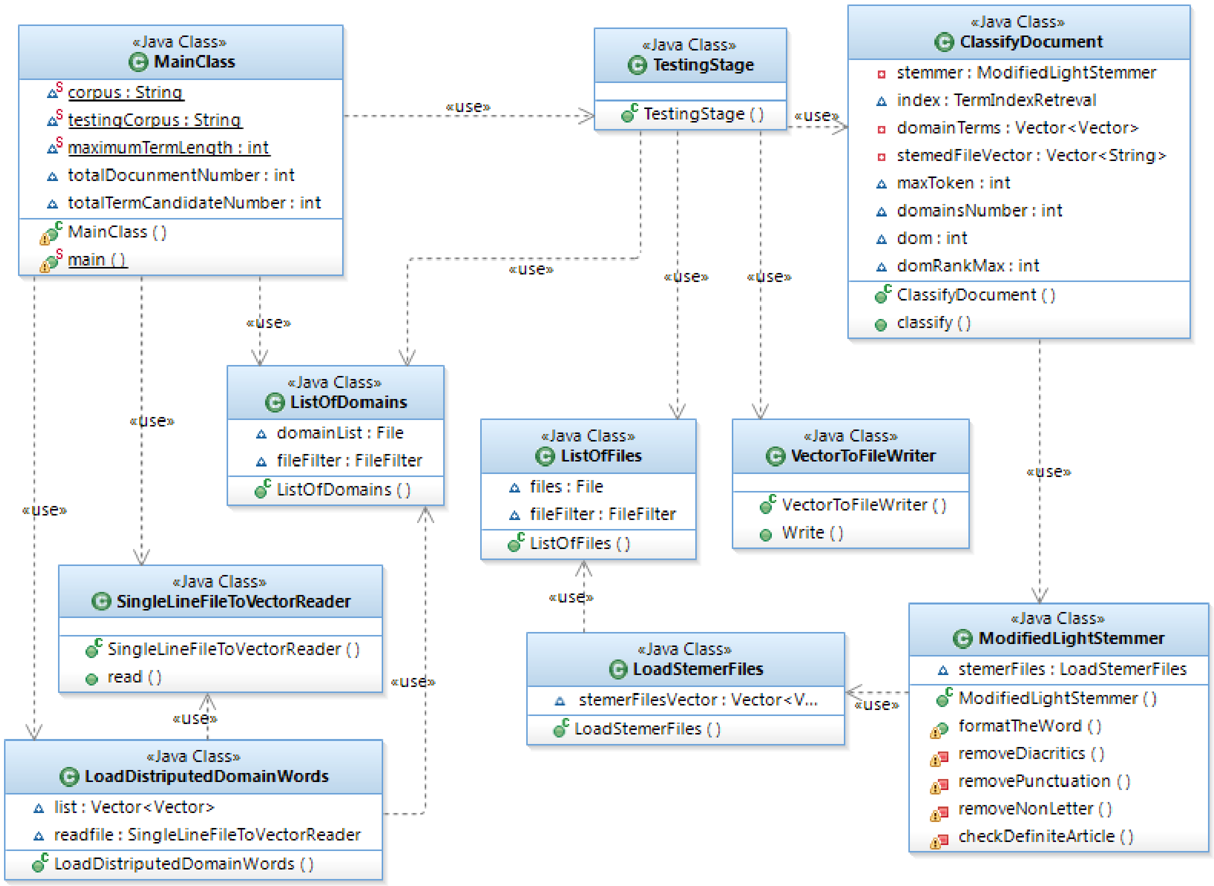


Figure 4.5: The classifier class diagram

4.3 Tools used

We use Eclipse as a java programming language because we have some experience with it and it is a very flexible environment and java have a lot of tools that could be very helpful. Also, java is very efficient language in memory management and have the ability to construct vectors that have the ability to change its size in running time.

We use Shereen Khoja Stemmer after modifying it for light stemming the tokens from the corpus because it is written in java, easy to understand, and modify.

We used IBM Rational Software Architect version 8.0. For class diagram drawing because it support reversal engineering.

We used Smart draw 2010 for drawing flow charts and component diagram.

We have programmed and develop some tools that help us in realizing the model. These tools are:

- The domain reader that reads each folder as a domain.

- The list of files name reader which read the files names within a folder into list.
- To read the content of each file we use Khoja single token file reader.
- To save the results into file we programmed a vector to file writer.

4.4 Problems appeared during the implementation of the model

When we try to implement the model on the real data some problems have appeared. We list these problems first and their solutions is listed in the following part:

1. The first problem comes out is the java heap error which means that the memory is not enough.
2. When I run the system it talks a lot of time that exceeded to several days and at the end it terminate without giving any indication for end execution or results.
3. When the system terminates I have to start over the experiment from the beginning.
4. We need to merge the results for the experiment.
5. As the system depend on the term occurrence there is a lot of search tasks which take a lot of execution time which need to be enhanced.

4.5 Solutions for the implementation problems

The problems listed before have different solution but we implement the available one's and we listed them here:

1. For memory error we use the –Xmx command for changing the execution memory option.
2. For execution time that ends with system termination. We try to partition our model to several execution units we found that we could implement the preprocessing, term extraction, and iteration counting for each domain separately.
3. We depend on files to save the results of each stage this help in overriding any stage we complete and continue from the last finished stage. As shown in appendix A for the model main class. We use three types of files:
    - Files for term candidate
    - Files for term iteration
    - Files for doc iteration.
4. In some domains the number of files was big and could not be handled together. So we write a code for domain separation as shown in appendix B and limits the number of files to 500.
5. After separation and processing we need to merge the results for each domain we write a code for merging the results as shown in appendix C.
6. Also we use sorted vectors for saving the results as there is a lot of search processes and we use a binary search algorithm (appendix D) which saves a lot of execution time.

Chapter 5: Experiments and Results

In this chapter we present the common term extraction metrics and the specification for each metric. After that we clarify the simple classifier that we have design using the extracted domain term matrix to help us in evaluating the model by quantifying the output of this classifier. The domain term matrix is extracted using our model. We compare the outputs and construct a confusion matrix to measure the accuracy and reliability of the model based domain term matrix classifier.

There are several metrics for evaluating term extraction methods such as shown in term ranking metrics in section 2.1.3 we use Termhood method that ranks the term according to its distributional behavior over the domain and across the rest of the corpora [13]. This measure do not evaluate the accuracy of the extracted terms. To do this we use the extracted domain term vector matrix to classify several documents and use the precession and recall measures to evaluate the accuracy of the model.

5.1 Evaluation methods

We will use the confusion matrix to evaluate the accuracy of the domain word matrix classifier resulting from our model. A confusion matrix is a specific table layout that allows visualization of the performance of an algorithm or model.

A confusion matrix [61] contains information about actual and predicted classifications done by a classification system. Performance of such systems is commonly evaluated using the data in the matrix. The following table shows the confusion matrix for a two class classifier Table 5.1.

Table 5.1: Confusion matrix

		predicted	
		negative	positive
actual	Negative	a	b
	Positive	c	d

The entries in the confusion matrix have the following meaning:

- a is the number of correct predictions that an instance is negative.
- b is the number of incorrect predictions that an instance is positive.
- c is the number of incorrect of predictions that an instance negative.
- d is the number of correct predictions that an instance is positive.

The following terms are defined for a two by two confusion matrix:

Accuracy

The accuracy (AC) is the percentage of the total number of predictions that were truthful. It is determined using the equation:

$$AC = \frac{a + d}{a + b + c + d} \qquad (5.1)$$

True positive rate (Recall, Sensitivity)

The recall or true positive rate (TP) is the percentage of positive cases that were correctly identified, as calculated using the equation:

$$R = TP = \frac{d}{c + d} \qquad (5.2)$$

True negative rate (Specificity)

The true negative rate (TN) is defined as the percentage of negatives cases that were classified correctly, as calculated using the equation:

$$TN = \frac{a}{a + b} \qquad (5.3)$$

Precision

Precision (P) is the percentage of the predicted positive cases that were correct, as calculated using the equation:

$$P = \frac{d}{b + d} \qquad (5.4)$$

False positive rate

The false positive rate (FP) is the percentage of negatives cases that were incorrectly classified as positive, as calculated using the equation:

$$FP = \frac{b}{a + b} \qquad (5.5)$$

False negative rate

The false negative rate (FN) is the percentage of positives cases that were incorrectly classified as negative, as calculated using the equation:

$$FN = \frac{c}{c + d}. \qquad (5.6)$$

The accuracy determined by equation 5.1 may not be an adequate performance measure when the number of negative cases is much greater than the number of positive cases [61].

Other performance measures account for this by including TP in a product: for example the geometric mean (g-mean) [62], as defined in the following equations, and F measure.

$$g\_mean1 = \sqrt{TP * P} \tag{5.7}$$

$$g\_mean2 = \sqrt{TP * TN} \tag{5.8}$$

$$F = \frac{(\beta^2 + 1) * P * TP}{\beta^2 * P + TP} \tag{5.9}$$

β^2 has a value from 0 to infinity and is used to control the weight assigned to TP and P.

Any classifier evaluated using equations 5.7, 5.8 or 5.9 will have a measure value of 0, if all positive cases are classified incorrectly.

The previous measure is good for binary classification problem but when the classification problem is not binary, the confusion matrix gets more complicated. In this case we can compute classifier accuracy as:

$$Accuracy = \frac{\sum_{i=1}^{n}(true\ clasification)}{total\ number\ of\ casses} \tag{5.10}$$

Where *i* is the class number and *n* is the total number of the classes.

5.2 Experimental design

5.2.1 The data

We choose (OSAc corpus) [35] which is collected from various websites as presented in Table 5.2, the corpus includes 22,429 text documents. Classified into 10 domains (Economics, History, Education and Family, Religious and Fatwa's, Sports, Heath, Astronomy, Law, Stories, and Cooking and Recipes). The corpus contains about 18,183,511 (18M) words and 449,600 keywords after stop words removal.

We use the UTF-8 coding system because it is universal and widely used and any type of document (html, doc, pdf) could be easily converted into text documents so the model will be widely applicable.

We separated the data into two parts one for training and testing. The separation of data is done manually before the training. The testing process will be applied by classifying the document using the extracted domain term matrix from the training stage. The domain term matrix size is shown in Table 5.5. The classifier testing corpus that contains 4670 document distributed into ten domains as shown in Table

5.3 is classified and the results of classification process was described in the confusion matrix in Table 5.6.

Table 5.2: OSAc corpus web site sources [35]**.**

Domain	**Number of documents**	**Web site's source**
Economic	3102	bbcarabic.com - cnnarabic.com -aljazeera.net - khaleej.com - banquecentrale.gov.sy
History	3233	www.hukam.net - moqatel.com - altareekh.com -islamichistory.net
Education and family	3608	saaid.net - naseh.net - almurabbi.com
Religious and fatwa's	3171	CCA corpus - EASC corpus moqatel.com - islamic-fatwa.com - saaid.net
Sport	2419	bbcarabic.com- cnnarabic.com - khaleej.com
Health	2292	dr-ashraf.com - CCA corpus - EASC corpus - W corpus - kids.jo - arabaltmed.com
Astronomy	557	arabastronomy.com- alkawn.net- bawabatalfalak.com- nabulsi.com- www.alkoon.alnomrosi.net
Law	944	lawoflibya.com - qnoun.com
Stories	726	CCA corpus- kids.jo- saaid.net
Cooking recipes	2373	aklaat.com - fatafeat.com
Total	22429	

Table 5.3: The number of documents to be classified for the domains

code	domain	Number of document
0	Economic	647
1	History	615
2	Education and family	712
3	Religious and fatwa's	713
4	Sport	522
5	Health	425
6	Astronomy	122
7	Law	213
8	Stories	173
9	Cooking recipes	528

5.3 The classifier

The classifier we use is a simple classifier that uses the extracted domain term vector to classify a document as shown in Figure 4.5 the API documentation of the classifier in appendix E part vii. This classifier classifies 4670 document distributed into ten domains. The classify process begins by loading the domain term matrix to the memory and for each file of the testing corpus it reads the content of the file and put each token in a vector. For each token we apply the light stemming process over the token and check if the stemmed token is in any domain. If it found in a domain a one is added to the domain counter and we have ten counters for the ten domains. After finishing the document we select the domain with big counter value and classify the document to this domain; after that, we compare the real domain with classified domain. If they are equal, we add one to true counter if not, the one is added to the wrong counter. The final report of the classifier gives the total true hits and the total wrong hits for each domain.

5.4 Results and discussion

Table 5.4: Number of term candidate for the domains

code	domain	Word vector size	Term candidate size			
			1	2	3	4
0	Economic	1618618	63035	435188	442312	339321
1	History	3668139	154943	789543	627274	411164
2	Education and family	2241672	122038	500072	383418	251896
3	Religious and fatwa's	1527183	58452	201014	160079	108847
4	Sport	1266928	47198	231434	235817	188543
5	Health	1490953	46942	157712	124271	84680
6	Astronomy	275469	22892	63914	52381	37312
7	Law	619292	28977	77772	61927	43573
8	Stories	2065902	101488	323691	230145	146663
9	Cooking recipes	268387	14997	62530	68563	54507

After implementing the candidate term extraction process on the corpus we got the following table that we coded the domains of the tested corpus in it from zero to nine as shown in Table 5.4. We write down the word vector length for each domain and for each term length the size of the term candidate. After implementing the rank method and distribute the terms over the domain depending on the rank value for each term to the ten domains we and exclude the terms with rank value equal or less than zero we got Table 5.5.

Table 5.5: Number of distributed terms over the domains

code	domain	Word vector size	Domain term size			
			1	2	3	4
0	Economic	1618618	24281	400464	433403	333269
1	History	3668139	94630	728401	610012	401045
2	Education and family	2241672	60425	447830	370287	244885
3	Religious and fatwa's	1527183	17281	170256	153029	105068
4	Sport	1266928	16623	209520	228048	181979
5	Health	1490953	16800	139945	119822	81926
6	Astronomy	275469	6079	53738	48490	34777
7	Low	619292	7316	66600	58972	41791
8	Stories	2065902	44111	282002	218437	140094
9	Cooking recipes	268387	6594	56600	64668	51641

We can conclude from comparing the one word term from Table 5.4 and Table 5.5that the distribution term process over the domains excludes a number of terms more than the two, three, and four word terms. Figure 5.1 and Figure 5.2 clarifying this effect. This means the one word term is less relevant to the domain.

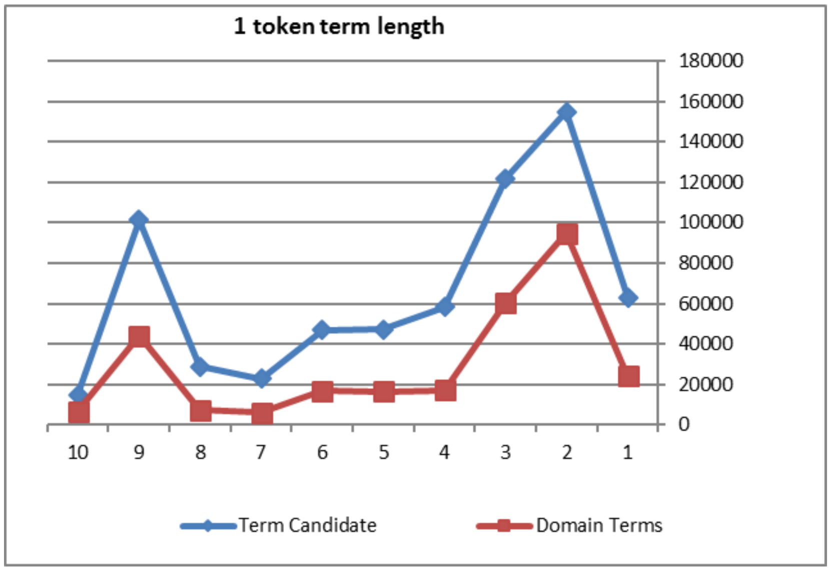


Figure 5.1: Comparing candidate terms with distributed terms for one word length

As result, when the term length increases the term relevancy to the domain increases. On the other hand, we can say that the one word term is very likely to appear in several domains more than others.

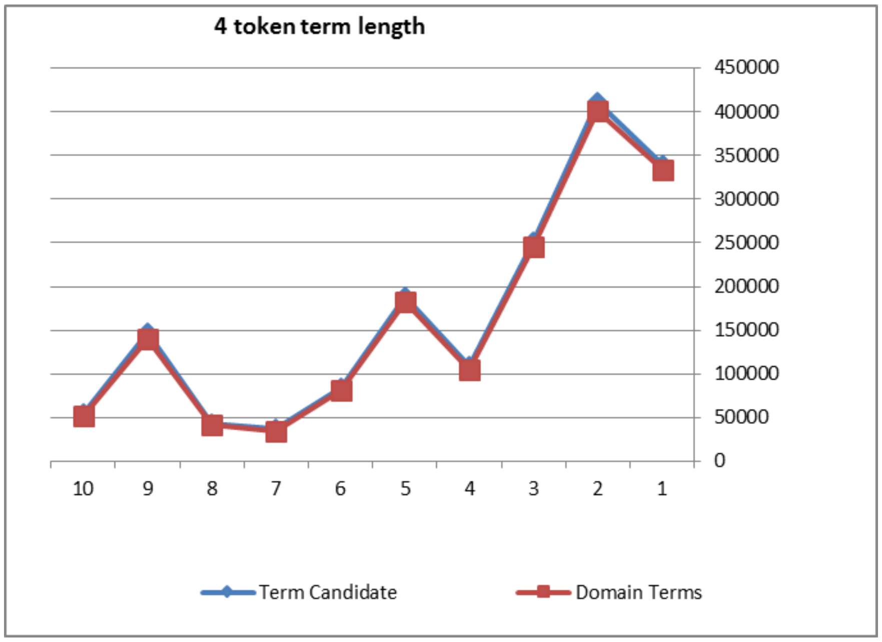


Figure 5.2: Comparing candidate terms with distributed terms for four word length

Furthermore the domain relevancy with term size effect is depicted in Figure 5.3. It represents the economy domain and this effect is true for the other domains. As it shows in the graph when the size of the term increased the excluded terms reduced.

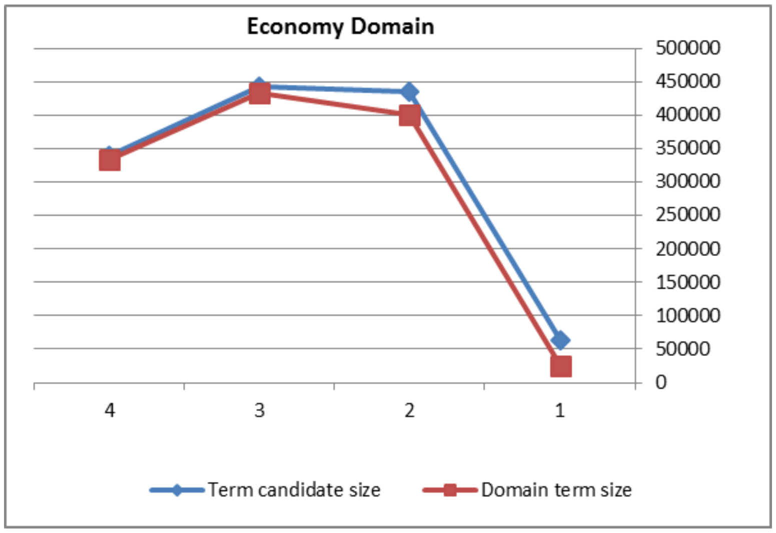


Figure 5.3: Term candidate and domain terms over term size for economy domain

As it shown in Table 5.6 the numbers from zero to nine represent the domains. REL represents the reliability of the classifier to classify the document domain. This means that the classifiers do not classify any document to the target domain. ACC represent the accuracy of the classifier to classify the document domain. Which means that the classifier will not classifies the current domain to any other domain. The confusion matrix is used for evaluating the performance of a system using the data in the matrix. Confusion matrix contains information about actual and predicted classifications done by a classification system [61].

Table 5.6: The classifier confusion matrix for the domains

		Real domain											
		0	1	2	3	4	5	6	7	8	9	**Sum**	**ACC**
Classified domains	0	634	1	0	0	6	1	0	0	0	1	**643**	**0.99**
	1	13	583	8	35	22	1	1	6	35	2	**706**	**0.83**
	2	0	27	682	2	0	5	0	0	1	5	**722**	**0.94**
	3	0	1	10	676	0	0	0	0	0	0	**687**	**0.98**
	4	0	0	0	0	494	0	0	0	1	0	**495**	**1.00**
	5	0	1	0	0	0	418	0	0	0	15	**434**	**0.96**
	6	0	0	0	0	0	0	121	0	0	0	**121**	**1.00**
	7	0	0	0	0	0	0	0	207	0	0	**207**	**1.00**
	8	0	2	12	0	0	0	0	0	136	0	**150**	**0.91**
	9	0	0	0	0	0	0	0	0	0	505	**505**	**1.00**
	Sum	**647**	**615**	**712**	**713**	**522**	**425**	**122**	**213**	**173**	**528**	**4670**	
	Recall	**0.98**	**0.95**	**0.96**	**0.95**	**0.95**	**0.98**	**0.99**	**0.97**	**0.79**	**0.96**		

As shown from the previous table we conclude that:

The rows 4,6,7,8 which represent the domains (Sport, Astronomy, Law, Cooking recipes) respectively could be classified with 100 percent accuracy that this document is not a member of other domains. This returns to the nature of the domain as these domains and the kind of words that are used in these domains.

The rest of the domains is also highly accurate except for row 1 which represent the History domain. It can be found that the History domain do not have a unique terms that could represent it clearly.

All the domains are highly reliable except for the story domain as the number of wrongly classified is high in aspect to the story tested documents. The majority of error goes to history domain as the story and history domains are close to each other.

As it shown in the table the classifier is very accurate for classifying all the domains except for the history domain also the classifier was highly reliable for all the domains except for the low domain. When we have review the corpus and the errors we found that the error in caused by the weakness of the corpus because of the few number of websites the corpus is grabbed from.

The total accuracy of the classifier which is calculated using equation 5.10 is 0.95 percent. This is a highly accurate classifier.

Reliability and accuracy of classifying some domains:

The accuracy of the classifier for the sample domains in Table 5.7, Table 5.8, and Table 5.9 is about 99 percent and the precision is about 100 percent and the recall is 97 percent These values means that the classifier which depends on the domain term matrix is reliable and highly accurate.

Table 5.7:Confution matrix for Economic domain

		predicted	
		negative	positive
actual	Negative	4014	9
	Positive	13	634

Accuracy	AC=	0.995289
Recall	R=	0.979907
Specificity	TN=	0.997763
Precision	P=	0.986003
	FP=	0.002237
	FN=	0.020093
	G-mean1=	0.98295
	G-mean2=	0.988795

Table 5.8: Confution matrix for Sprot domain

		predicted	
		negative	positive
actual	Negative	4147	1
	Positive	28	494

Accuracy	AC=	0.99379
Recall	R=	0.94636
Specificity	TN=	0.999759
Precision	P=	0.99798
	FP=	0.000241
	FN=	0.05364
	G-mean1=	0.971827
	G-mean2=	0.972693

Table 5.9: Confution matrix for Astronomy domain

		predicted	
		negative	positive
actual	Negative	4548	0
	Positive	1	121

Accuracy	AC=	0.999786
Recall	R=	0.991803
Specificity	TN=	1
Precision	P=	1
	FP=	0
	FN=	0.008197
	G-mean1=	0.995893
	G-mean2=	0.995893

Chapter 6: Conclusion and Future Work

To conclude our work and discuss our contribution and state some issues that need further research and enhancement; we developed a new model for domain relevant term extraction from Arabic text corpus. This model is constructed through four stages: First, is preprocessing where we modified the Khoja stemmer to be a light stemmer to suit the domain we work with. Second, is candidate term extraction where we use the sliding window method for length from one to four to extract the candidate terms and we excluded the terms that contains stop words in the window. Third, is the candidate term ranking where we implemented a termhood ranking method that takes into consideration the distributional behavior of the terms over the domain and across the rest of the corpus. Fourth, we used a simple method depends on the rank value for each term over the ten domains and assigned the term to the strongest domain.

After we extracted the domain term matrix we used this matrix as classifier. We programmed a simple classifier that use this matrix to classify the documents that need to be classified in the testing stage. These documents is classified by converted them into stemmed word vector and then calculate the binary distance between the document vector and the domain vectors and give the document the domain with high distance. This process is done for all the documents and domains to be tested. Then calculate the confusion matrix to evaluate the efficiency of the classifier that indicates that the domain term matrix is efficient and effective for a classifier.

Our model takes several criteria into consideration like the specification of the corpus the model work on. The term extraction method used in extracting the candidate terms. The ranking methodology the model use for ranking the terms. The distribution method for distributing the terms over the domains. Finally, the evaluating methods and tools for evaluating the model.

The model deals with several domains so the corpus should be separated into domains. On the other hand, most of the other works deal with general corpus and others with one domain specific corpus.

The model use the sliding window method for candidate term extraction; on the other hand, other works deal with several methods for term extraction like (NLP patterns, Local grammar approach, or syntactic patterns). We used this method because the other method depend on the taggers and the existing taggers has a low accuracy – nearly 25 percent of the words not identified by the tagger [31] which affect the accuracy of the models.

The ranking method we used depends on several domains which measure the term prevalence and tendency over the domain and across the rest of the corpus.

We use a simple method for term distribution over the domains to generate the domain relevant term matrix which depends on the ranking value for the term over all the corpora and assign the term to the domain with high rank. Other works deal with one domain and this differentiation is not exist on other works.

Finally, we design a classifier depending on the domain relevant term matrix to classify a domain known document and use a confusion matrix for evaluating the model.

Although the proposed model uses a several domain corpus, it uses a light stemmer for preprocessing; extract the candidate terms using a sliding window; and, ranks the candidate terms using a termhood method. There are still several ways for improving the model:

- Use several corpuses and study the effect of the corpus change on the results.
- In the preprocessing stage we could evaluate several preprocessing options and compare the effect of each option.
- In the term extraction stage we could use other methods for candidate term extraction like pattern passed, local grammar or other NLP methods and examine the model for these options.
- For the term ranking stage we could experiment several ranking methods and compare the implementation results.

References

[1] J. Sager, *A practical course in terminology processing*. Amsterdam/Philadelphia: John Benjamins, 1990.

[2] S. Love, “Benchmarking the performance of Two Automated Term-extraction systems: LOGOS and ATAO,” university of montrial, 2000.

[3] T. Vu, A. Aw, and M. Zhang, “Term extraction through unithood and termhood unification,” in *International Joint Conference on Natural Language Processing - IJCNLP*, 2008, pp. 631–636.

[4] M. Syafrullah and N. Salim, “Improving Term Extraction Using Particle Swarm Optimization Techniques,” *JOURNAL OF COMPUTING,* vol. 2, no. 2, pp. 116–120, 2010.

[5] R. Mitkov, G. Corpas, and others, “Mutual terminology extraction using a statistical framework,” *Procesamiento del lenguaje Natural,* vol. 41, no. Section 2, pp. 107–112, 2008.

[6] A. M. Saif, M. J. A. Aziz, C. Science, and S. Publications, “An Automatic Collocation Extraction from Arabic Corpus,” *Journal of Computer Science*, vol. 7, no. 1, pp. 6–11, 2011.

[7] J. Nam, “A Local-Grammar-based Approach to Recognizing of Proper Names in Korean Texts,” in *the 5th Workshop on Very Large Corpora (WVLC-5)*, 1997, pp. 273–288.

[8] J. Foo, “Term extraction using machine learning,” Linköping University, LINKÖPING, 2009.

[9] S. Katz, “Distribution of content words and phrases in text and language modelling,” *Natural Language Engineering*, vol. 2, no. 1, pp. 15–59, Mar. 1996.

[10] W. Wong, W. Liu, and M. Bennamoun, “Determining termhood for learning domain ontologies in a probabilistic framework,” *In Proceedings of the sixth Australasian conference on Data mining and analytics,* 2007, vol. 07, pp. 51–60.

[11] J. Foo, “Exploring termhood using language models,” in *NEALT PROCEEDINGS SERIES VOL. 12*, 2011, pp. 32–35.

[12] W. Wong, W. Liu, and M. Bennamoun, “Determining the unithood of word sequences using mutual information and independence measure,” in *Proceedings of*

the 10th Conference of the Pacific Association for Computational Linguistics (PACLING), 2008.

[13] W. Wong, W. Liu, and M. Bennamoun, “Determining termhood for learning domain ontologies using domain prevalence and tendency,” in *Proceedings of the sixth Australasian conference on Data mining and analytics*, 2007, vol. 70, no. AusDM, pp. 47–54.

[14] C. Jacquemin and D. Bourigault, “Term extraction and automatic indexing,” in *Handbook of Computational Linguistics*, 2003, pp. 599–615.

[15] J. S. Justeson and S. M. Katz, “Technical terminology: some linguistic properties and an algorithm for identification in text,” *Natural Language Engineering*, vol. 1, no. 1, pp. 9–27, 1995.

[16] S. N. Kim and L. Cavedon, “Classifying Domain-Specific Terms Using a Dictionary,” *Proceedings of the Australasian Language Technology Workshop*, vol. 09, pp. 57–65, 2011.

[17] J. R. Firth, *Papers in Linguistics 1934-1951*. Oxford University Press, 1957, p. 233.

[18] K. Kageura and B. Umino, “Methods of automatic term recognition: A review,” *Terminology*, vol. 3, no. 2, pp. 259–289, 1996.

[19] S. Crain, “What are Core Linguistic Properties?,” *Proceedings of the 9th Conference of the Australasian Society for Cognitive Science*, pp. 67–71, 2010.

[20] H. Aliane, Z. Alimazighi, and M. Cherif, “Al-Khalil: The Arabic Linguistic Ontology Project,” *in Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC’10)*, 2010.

[21] M. Hahn, “Arabic Relativization Patterns: A Unified HPSG Analysis,” *Proceedings of* HPSG 2012 Conference/Ellipsis Workshop*, Daejeon, Korea*, 2012.

[22] L. Dice, “Measures of the amount of ecologic association between species,” *Ecology*, 1945.

[23] C. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*, 1st ed. Cambridge University Press, 2008.

[24] T. Hisamitsu, Y. Niwa, and S. Nishioka, “Term extraction using a new measure of term representativeness,” *Proceedings of the Second International Conference on Language Recources and Evaluation (LREC 2000)*, pp. 13–20, 2000.

[25] K. T. Frantzi, S. Ananiadou, and J. Tsujii, “The C-value/NC-value Method of Automatic Recognition for Multi-Word Terms,” in *Proceedings of the Second*

European Conference on Research and Advanced Technology for Digital Libraries, 1998, pp. 585–604.

[26] L. Ahrenberg, “Term extraction: A Review,” Linköping University, 2005.

[27] A. Nazarenko and H. Zargayouna, “Evaluating term extraction,” in *International Conference RANLP 2009*, 2009, pp. 299–304.

[28] M. Attia, L. Tounsi, P. Pecina, and J. van Genabith, “Automatic extraction of arabic multiword expressions,” *23rd International Conference on Computational Linguistics Proceedings of the Workshop on Multiword Expressions: from Theory to Applications (MWE 2010)*, no. August, pp. 18–26, 2010.

[29] M. Hong, “Hybrid filtering for extraction of term candidates from German technical texts,” in *International Conference on Terminology and Artificial Intelligence(TIA-2001)*, 2001.

[30] W. Wong, “Determination of unithood and termhood for term recognition,” in *Handbook of research on text and web mining technologies*, 2009.

[31] S. AlGahtani, W. Black, and J. Mcnaught, “Arabic part-of-speech tagging using transformation-based learning,” in *Proceedings of the Second International Conference on Arabic Language Resources and Tools*, 2009, no. 2001, pp. 66–70.

[32] D. Kurz and F. Xu, “Text mining for the extraction of domain relevant terms and term collocations,” in *Proceedings of the International Workshop on Computational Approaches to Collocations*, 2002.

[33] C. Nemallapudi, “Evaluating Term Extraction Methods for Domain Analysis,” Virginia Polytechnic Institute and State University, 2010.

[34] M. Beseiso, A. R. A. R. Ahmad, and R. Ismail, “A Survey of Arabic language Support in Semantic web,” *International Journal of Computer Applications IJCA*, vol. 9, no. 1, pp. 24–28, Nov. 2010.

[35] M. K. Saad and W. Ashour, “OSAC: Open Source Arabic Corpora,” *International Conference on Electrical and Computer Systems*, vol. 18, 2010.

[36] M. Syiam, Z. Fayed, and M. Habib, “An intelligent system for Arabic text categorization,” *International Journal of Intelligent Computing and Information Sciences*, vol. 6, no. 1, pp. 1–19, 2006.

[37] M. K. Saad and W. Ashour, “Arabic Morphological Tools for Text Mining,” *International Conference on Electrical and Computer Systems*, vol. 18, p. 19, 2010.

[38] L. Lopes, P. Fernandes, and R. Vieira, “ExATOlp-an automatic tool for term extraction from Portuguese language corpora,” *Proceedings of the LTC’09*, 2009.

[39] F. Sclano and P. Velardi, “Termextractor: a web application to learn the shared terminology of emergent web communities,” *Enterprise Interoperability II,* pp. 287–290, 2007.

[40] E. Atlam, M. Fuketa, K. Morita, and J. Aoe, “Automatic building an extensive Arabic FA terms dictionary,” *Proceedings of World Academy of Science, Engineering and Technology,* vol. 44, pp. 719–725, 2010.

[41] L. Larkey, L. Ballesteros, and M. Connell, “Light stemming for Arabic information retrieval,” *Arabic Computational Morphology: Knowledge-based and empirical method,,* vol. 38, 2007.

[42] K. Al Khatib and A. Badarneh, “Automatic extraction of Arabic multi-word terms,” in *Computer Science and Information Technology (IMCSIT), Proceedings of the 2010 International Multiconference on*, 2010, pp. 411–418.

[43] R. Al-shalabi and G. Kanaan, “Constructing an automatic lexicon for Arabic language,” *international journal of computing &information sciences*, vol. 2, no. 2, pp. 114–128, 2004.

[44] H. K. Al Ameed, S. O. Al Ketbi, A. A. Al-Kaabi, K. Al Shebli, N. Al Shamsi, N. H. Al Nuaimi, and S. S. Al Muhairi, “Arabic light stemmer: A new enhanced approach,” in *The Second International Conference on Innovations in Information Technology (IIT’05),* 2005, pp. 1–9.

[45] H. Traboulsi, “Arabic named entity extraction: A local grammar-based approach,” in *Computer Science and Information Technology, 2009. IMCSIT’09. International Multiconference on*, 2009, pp. 139–143.

[46] H. Traboulsi, “A local grammar for proper names,” University of Surrey, 2004.

[47] A. Al-Taani, “A rule-based approach for tagging non-vocalized Arabic words,” *Arab Journal of Information Technology (IAJIT),* vol. 6, no. 3, pp. 320–328, 2009.

[48] S. Boulaknadel, B. Daille, and D. Aboutajdine, “A multi-word term extraction program for Arabic language,” *In Proceeding of the Sixth LREC*, pp. 1485–1488, 2008.

[49] S. Khoja, “APT: Arabic part-of-speech tagger,” *proceding of the Student Workshop at NAACL*, 2001.

[50] T. Naseem and B. Snyder, “Multilingual part-of-speech tagging: Two unsupervised approaches,” *Journal of Artificial Intelligence Research*, vol. 36, pp. 1–45, 2009.

[51] S. Mansour, K. Sima'an, and Y. Winter, "Smoothing a lexicon-based pos tagger for Arabic and Hebrew," *Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources*, 2007.

[52] G. Kanaan, R. AL-SHALABI, and M. Sawalha, "Full automatic Arabic text tagging system," *proceedings of the International Conference on Information Technology and Natural Sciences*, pp. 258–267, 2003.

[53] E. Atlam, "A New Weight Function for Constructing Field Association Terms using Concurrent Words," *International Journal of Computer Science Issues*, vol. 8, no. 4, pp. 16–27, 2011.

[54] A. Abdelali, "Building a modern standard Arabic corpus," *In workshop on computational modeling of lexical acquisition*, 2005.

[55] M. Abbas and K. Smaïli, "Comparison of topic identification methods for arabic language," *Recent Advances in Natural Language Processing (RANLP05)*, pp. 14–17, 2005.

[56] L. Al-Sulaiti, "Designing and developing a corpus of contemporary Arabic," The University of Leeds, 2004.

[57] L. Lopes, R. Vieira, M. J. Finatto, and D. Martins, "Extracting compound terms from domain corpora," *Journal of the Brazilian Computer Society*, vol. 16, no. 4, pp. 247–259, Aug. 2010.

[58] M. Diab, K. Hacioglu, and D. Jurafsky, "Automatic tagging of Arabic text: From raw text to base phrase chunks," *In Proceedings of HLT-NAACL 2004: Short Papers (HLT-NAACL-Short '04)*, pp. 149–152, 2004.

[59] R. Basili, A. Moschitti, M. T. Pazienza, and F. M. Zanzotto, "A contrastive approach to term extraction," *International Conference on Terminology and Artificial Intelligence(TIA-2001)*, 2001.

[60] A. Hippisley and D. Cheng, "The head-modifier principle and multilingual term extraction," *Natural Language*, vol. 11, no. 2, pp. 129–157, Jun. 2005.

[61] R. Kohavi and F. Provost, "Special Issue on Applications of Machine Learning and the Knowledge Discovery Process," *Machine Learning*, vol. 30, no. 2/3, pp. 127–271, 1998.

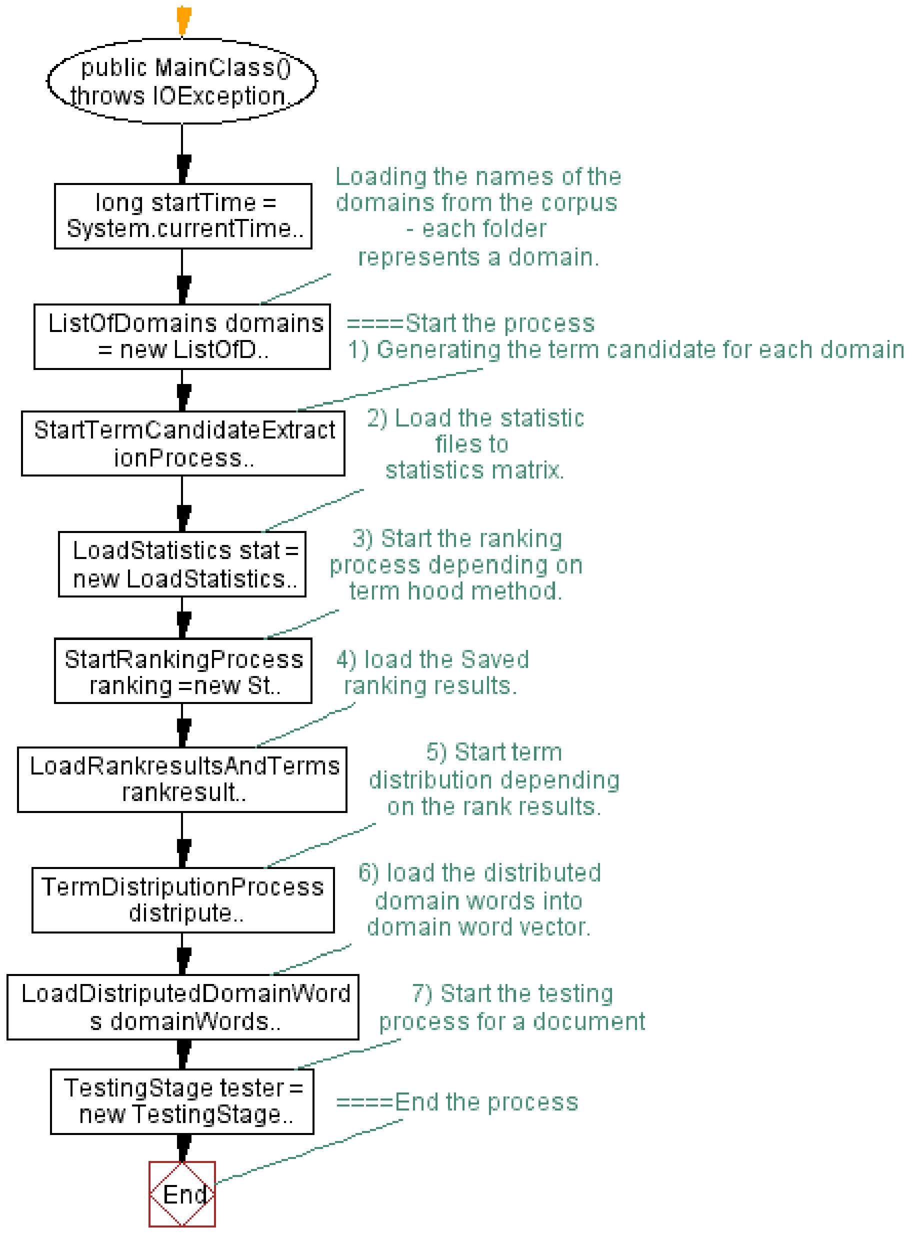
[62] C. Chen, A. Liaw, and L. Breiman, "Using random forest to learn imbalanced data," Clefornia, 2004.

Appendices

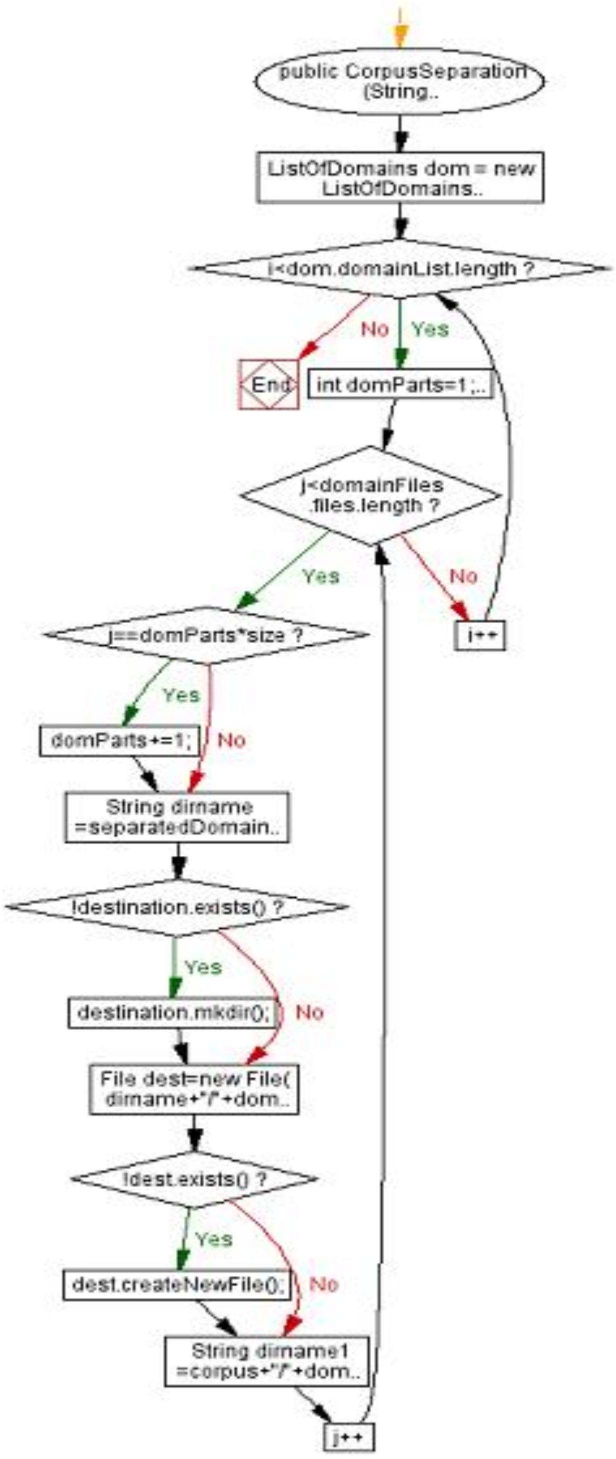
The appendices list the flow charts and java API classes we developed for model:

A. Flowchart for the model main class
B. Flowchart for domain separation
C. Flowchart for result merging for a domain
D. Flowchart for binary search with insert
E. API documentation of the module
    i. Class Modified Light Stemmer
    ii. Class Start Term Candidate Extraction Process
    iii. Class Start Ranking Process
    iv. Class Terms Ranker
    v. Class Term Distribution Process
    vi. Class Testing Stage
    vii. Class Classify Document
F. Tracking the rank for term يوهانسون
G. Tracking the rank for term يطرح أسهم شركة مدمجة

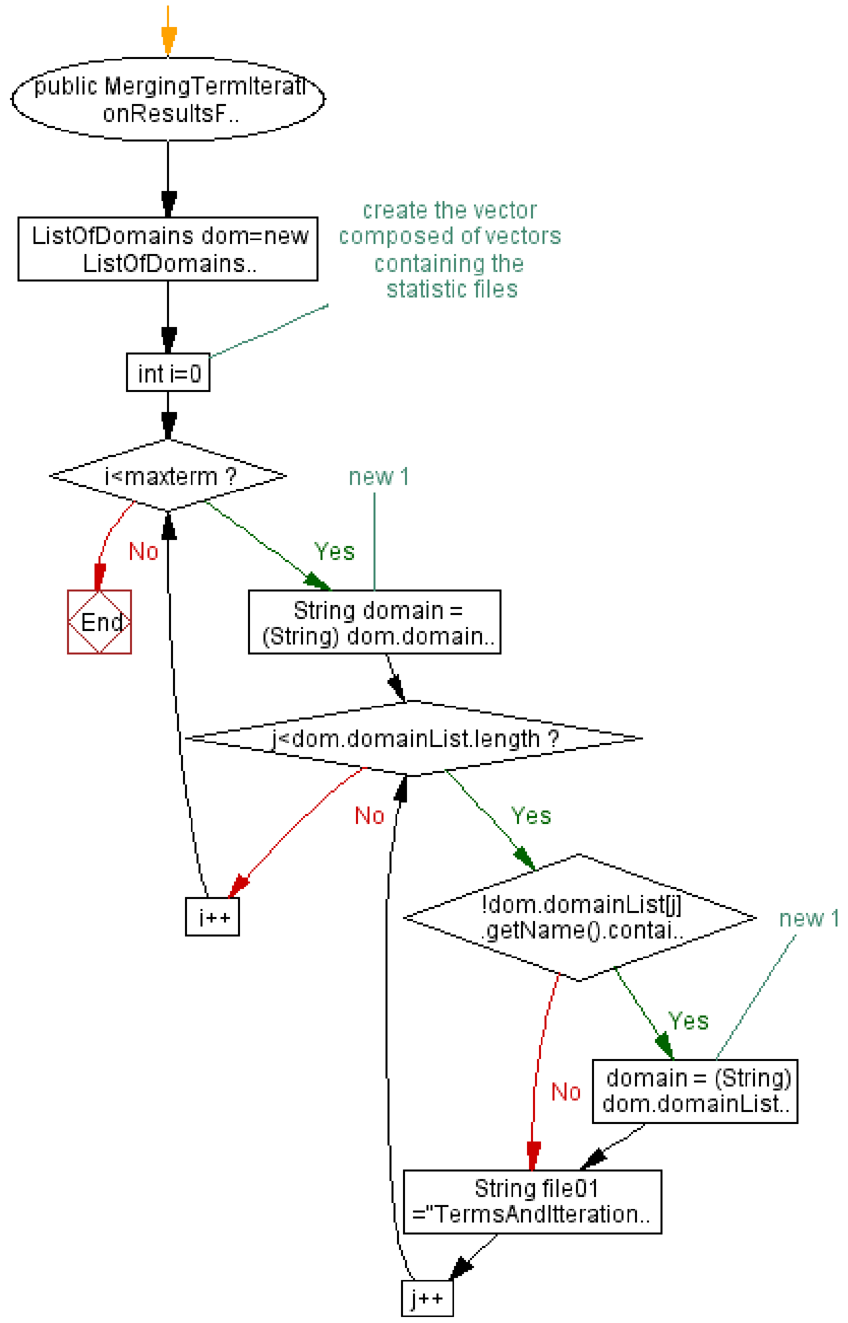
A. Flowchart for the model main class


public MainClass() throws IOException..
long startTime = System.currentTime..
ListOfDomains domains = new ListOfD..
StartTermCandidateExtractionProcess..
LoadStatistics stat = new LoadStatistics..
StartRankingProcess ranking =new St..
LoadRankresultsAndTerms rankresult..
TermDistriputionProcess distripute..
LoadDistriputedDomainWords domainWords..
TestingStage tester = new TestingStage..
End
Loading the names of the domains from the corpus - each folder represents a domain.
====Start the process
1) Generating the term candidate for each domain
2) Load the statistic files to statistics matrix.
3) Start the ranking process depending on term hood method.
4) load the Saved ranking results.
5) Start term distribution depending on the rank results.
6) load the distributed domain words into domain word vector.
7) Start the testing process for a document
====End the process

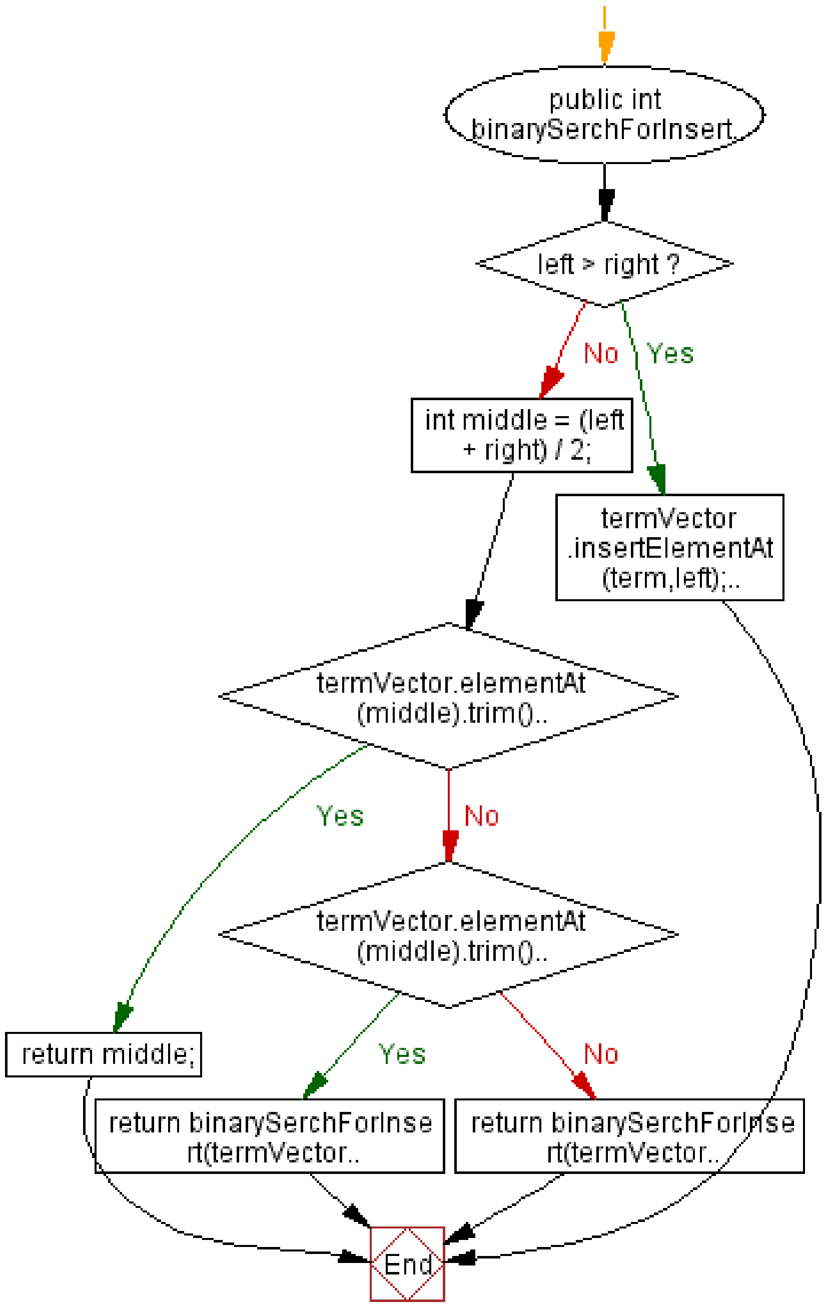
B. Flowchart for domain separation


public CorpusSeparation (String..
ListOfDomains dom = new ListOfDomains..
i<dom.domainList.length ?
No
Yes
End
int domParts=1;..
j<domainFiles .files.length ?
Yes
No
i++
j==domParts*size ?
Yes
No
domParts+=1;
String dirname =separatedDomain..
!destination.exists() ?
Yes
No
destination.mkdir();
File dest=new File(dirname+"/"+dom..
!dest.exists() ?
Yes
No
dest.createNewFile();
String dirname1 =corpus+"/"+dom..
j++

C. Flowchart for result merging for a domain


public MergingTermIterati onResultsF..
ListOfDomains dom=new ListOfDomains..
create the vector composed of vectors containing the statistic files
int i=0
i<maxterm ?
No
End
Yes
new 1
String domain = (String) dom.domain..
j<dom.domainList.length ?
No
i++
Yes
!dom.domainList[j] .getName().contai..
new 1
Yes
No
domain = (String) dom.domainList..
String file01 ="TermsAndItteration..
j++

D. Flowchart for binary search with insert


public int binarySerchForInsert..
left > right ?
No
Yes
int middle = (left + right) / 2;
termVector .insertElementAt (term,left);..
termVector.elementAt (middle).trim()..
Yes
No
termVector.elementAt (middle).trim()..
return middle;
Yes
No
return binarySerchForInsert(termVector..
return binarySerchForInsert(termVector..
End

E. API documentation of the module

Class Summary

Class	Description
AADRTE	The Class AADRTE is the Main Class of the model that calls all the sup classes to perform a complete sequence of the model processes.
ClassifyDocument	The Class ClassifyDocument is used to classify the document within the Testing_corpus folder.
ListOfDomains	The Class ListOfDomains read the folders in the corpus as a domain list.
ListOfFiles	The Class ListOfFiles.
LoadDistriputedDomainWords	The Class LoadDistriputedDomainWords.
LoadRankresultsAndTerms	The Class LoadRankresultsAndTerms for loading the rank results from rank files in the rankrResults directory.
LoadStatistics	The Class LoadStatistics create the vector composed of vectors containing the statistic files.
LoadStemerFiles	The Class LoadStemerFiles.
ModifiedLightStemmer	The Class ModifiedLightStemmer.
SingleLineFileToVectorReader	The Class SingleLineFileToVectorReader.
SingleTokenFileReader	The Class SingleTokenFileReader.
StartRankingProcess	The Class StartRankingProcess.
StartTermCandidateExtractionProcess	The Class StartTermCandidateExtractionProcess.
TermDistriputionProcess	The Class TermDistriputionProcess.
TermIndexRetreval	The Class TermIndexRetreval.
TermsRanker	The Class TermsRanker.
TestingStage	The Class TestingStage.
VectorToFileWriter	The Class VectorToFileWriter.
writeAllDataToOneFile	The Class writeAllDataToOneFile.

i. Class ModifiedLightStemmer

```
java.lang.Object
 └─AADRTE.ModifiedLightStemmer
```

```
public class ModifiedLightStemmer
extends Object
```

The Class ModifiedLightStemmer. This class is a modification of khoja stemmer also we modified the stemmer files to accept the Arabic letters only. It is also loads the stemmer files within StemmerFiles folder. To be used in the stemming process

---

Field Summary

Modifier and Type	Field and Description
(package private) `LoadStemerFiles`	**`stemerFiles`** The stemmer files.

Constructor Summary

Constructor and Description
`ModifiedLightStemmer``()` Instantiates a new modified light stemmer.

Method Summary

Modifier and Type	Method and Description
private `String`	**`checkDefiniteArticle`**`(String word)` Check definite article.
`String`	**`formatTheWord`**`(String currentWord)` Format the word.
private boolean	**`removeNonLetter`**`(String currentWord, StringBuffer modifiedWord)` Removes the non Arabic letter.

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

stemerFiles

`LoadStemerFiles` **`stemerFiles`**

The stemmer files. This variable calls the load stemmer files class to load the stemmer files to a vector from StemmerFiles folder.

Constructor Detail

ModifiedLightStemmer

```
public ModifiedLightStemmer()
```

Instantiates a new modified light stemmer.

Method Detail

checkDefiniteArticle

```
private String checkDefiniteArticle(String word)
```

Check definite article. This method return the word removing the definite article from it.

Parameters:

`word` - is the original word to be checked.

Returns:

the string contains the word without definite article.

---

formatTheWord

```
public String formatTheWord(String currentWord)
 throws IOException
```

Format the word. this method apply all the preprocessing steps on the word

Parameters:

`currentWord` - the current word

Returns:

the string returns the preprocessed word.

Throws:

`IOException` - Signals that an I/O exception has occurred.

---

removeNonLetter

```
private boolean removeNonLetter(String currentWord,
 StringBuffer modifiedWord)
```

Removes the non-Arabic letter.

Parameters:

`currentWord` - the current word

`modifiedWord` - the modified word

Returns:

True, if successful

ii. Class StartTermCandidateExtractionProcess

```
java.lang.Object
 └─AADRTE.StartTermCandidateExtractionProcess
```

```
public class StartTermCandidateExtractionProcess
extends Object
```

The Class StartTermCandidateExtractionProcess. In this class we extract the candidate terms for the corpus and counting the iteration for each term. Also we count the number of the document the terms appear in. all the previous statistics are saved to files.

Field Summary

Modifier and Type	Field and Description
`(package private) int`	**`corpusNumberOfFiles`** The corpus number of files.
`(package private) Vector<Vector<String>>`	**`docIteration`** The document iteration Vector.
`(package private) BufferedWriter`	**`fileBuffer`** The file buffer.
`(package private) FileWriter`	**`fileWriter`** The file writer.
`(package private) Vector<Vector<String>>`	**`stemedfilevector`** The stemmed file vector.
`(package private) ModifiedLightStemmer`	**`stemer`** instantiate The stemmer.
`(package private) Vector<Vector<String>>`	**`termCandidate`** The term candidate Vector.
`(package private) int`	**`termCandidateOccurance`** The term candidate occurrence.
`(package private) TermIndexRetreval`	**`termIndex`** instantiate The term index.
`(package private) Vector<Vector<String>>`	**`termIteration`** The term iteration Vector.
`(package private) VectorToFileWriter`	**`vectorFileWriter`** instantiate The vector file writer.

Constructor Summary

Constructor and Description
`StartTermCandidateExtractionProcess``(String corpus, int maxTokens)` Instantiates a new start term candidate extraction process.

Method Summary

Modifier and Type	Method and Description
`private void`	**`UpdateDocItteration`**`(int vectorIndex)` Update doc iteration.

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

termCandidate

`Vector<Vector<String>> termCandidate`

The term candidate Vector.

---

docIteration

`Vector<Vector<String>> docIteration`

The document iteration Vector.

---

termIteration

`Vector<Vector<String>> termIteration`

The term iteration Vector.

---

stemedfilevector

`Vector<Vector<String>> stemedfilevector`

The stemmed file vector.

---

corpusNumberOfFiles

`int corpusNumberOfFiles`

The corpus number of files.

---

termCandidateOccurance

`int termCandidateOccurance`

The term candidate occurrence.

---

fileWriter

`FileWriter fileWriter`

The file writer.

---

fileBuffer

`BufferedWriter fileBuffer`

The file buffer.

---

stemer

`ModifiedLightStemmer stemer`

instantiate The stemmer.

---

termIndex

`TermIndexRetreval` **`termIndex`**

instantiate The term index.

---

vectorFileWriter

`VectorToFileWriter` **`vectorFileWriter`**

instantiate The vector file writer.

Constructor Detail

StartTermCandidateExtractionProcess

```
public StartTermCandidateExtractionProcess(String corpus,
 int maxTokens)
 throws IOException
```

Instantiates a new start term candidate extraction process.

Parameters:

`corpus` - the folder name of the corpus

`maxTokens` - the max tokens in the term

Throws:

`IOException` - Signals that an I/O exception has occurred.

Method Detail

UpdateDocItteration

```
private void UpdateDocItteration(int vectorIndex)
```

Update doc iteration. this method increase the doc iteration counter for a term

Parameters:

`vectorIndex` - the vector index of the term.

iii. Class StartRankingProcess

```
java.lang.Object
  └AADRTE.StartRankingProcess
```

```
public class StartRankingProcess
extends Object
```

The Class StartRankingProcess. This class starts the ranking process by calling the ListOfDomains class that reads the list of domains then starts LoadStatistics class to read the data stored by the previous component then start ranking each candidate term in the list for all the domains by calling the TermRanker class.

Field Summary

Modifier and Type	Field and Description
`(package private) Vector<Vector<String>>`	**`rankVector`** The rank vector Is the vector that will contain the rank values for the candidate term matrix.

Constructor Summary

Constructor and Description
`StartRankingProcess``(String corpus, Vector<Vector<String>> termCandidate, Vector<Vector<String>> docIteration, Vector<Vector<String>> termIteration, int maxTermLength, int courpusFiles, int totalCandidateTermOcurance)` Instantiates a new start ranking process.

Method Summary

Modifier and Type	Method and Description

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

rankVector

```
Vector<Vector<String>> rankVector
```

The rank vector Is the vector that will contain the rank values for the candidate term matrix.

Constructor Detail

StartRankingProcess

```
public StartRankingProcess(String corpus,
 Vector<Vector<String>> termCandidate,
 Vector<Vector<String>> docIteration,
Vector<Vector<String>> termIteration,
int maxTermLength,
int courpusFiles,
int totalCandidateTermOcurance)
throws IOException
```

Instantiates a new start ranking process.

Parameters:

`corpus` - the corpus

`termCandidate` - contains the term candidates

`docIteration` - contains the doc iteration

`termIteration` - contains the term iteration

`maxTermLength` - is the max term length

`courpusFiles` - is the number of corpus files

`totalCandidateTermOcurance` - is the total number of candidate term occurrence

Throws:

`IOException` - Signals that an I/O exception has occurred.

iv. Class TermsRanker

```
java.lang.Object
 └ AADRTE.TermsRanker
```

```
public class TermsRanker
extends Object
```

The Class TermsRanker.
This class ranks a term by calling TermIndexRetreval to retrieve the index of the term to be used for calling the statistics of the term for completing the rank process.

Field Summary

Modifier and Type	Field and Description
(package private) double	**ACDWa** The ACDwa is the average contextual discriminative weight of term a.
(package private) double	**docFreqA** The document frequency of term a is number of documents the term a appear in.
(package private) Vector<Vector<String>>	**docIter** The dociter is the document iteration of term a.
(package private) String	**domain** The domain name.
(package private) File[]	**domainList** The domain list.
(package private) double	**DWa** The Dwa is the discriminative weight of term a.
(package private) double	**Ftc** The Ftc is frequency summation of all candidate terms within the corpus.
(package private) TermIndexRetreval	**index** The index is the calling for term index retrieval class.
(package private) double	**M** The m is number of documents in the corpus.
(package private) int	**maxToken** The max token.
(package private) double	**TermFreqAD** The Termfreqad is frequency of term a over the target domain d.
(package private) double	**TermFreqADnot** The Termfreqadnot frequency of term a over the rest of

	corpus.
(package private) Vector<Vector<String>>	**termIter** The termiter is the term iteration in corpus.
(package private) int	**termLength** The term length.
(package private) Vector<Vector<String>>	**terms** The terms is the terms vector.

Constructor Summary

Constructor and Description
TermsRanker(File [] domList, int courpusFiles, int totalCandidateTermOcurance) Instantiates a new terms ranker.

Method Summary

Modifier and Type	Method and Description
private double	**ACC**(String a) Acc is a method to calculate the adjusted contextual contribution.
private double	**ACDW**(String a) Acdw is a method to calculate the average contextual discriminative weight of term a .
private int	**documentFrequency**(String a) Document frequency is a method to calculate the document frequencies of term a .
private int	**domainNotTermFrequency**(String a) Domain not term frequency.
private int	**domainTermFrequency**(String a) Domain term frequency.
private double	**DP**(String a) Dp is a method to calculate the domain prevalence for term a.
private double	**DPh**(String h) Dph is a method to calculate the domain prevalence for term header h .
private double	**DT**(String a) Dt is a method to calculate the domain tendency for term a.
private double	**DW**(String a) Dw is a method to calculate the discriminative weight for term a.
private double	**log2**(double num) Log2.
private double	**MF**(String a) Mf is a method to calculate the modifier factor for term a.
private double	**NGD**(String a, String c) Ngd is a method to calculate the normalized google distance between

	term a and term c .
`double`	`rank(String a, String domain1, Vector<Vector<String>> termCandidate, Vector<Vector<String>> docIteration, Vector<Vector<String>> termIteration)` Rank is a method to calculate the total rank for term a.
`private double`	`sim(String a, String c)` Sim is a method to calculate the similarity between term a and term c .
`private double`	`TH(String a)` Th is a method to calculate the termhood for term a.

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

`double` **`M`**

The m is number of documents in the corpus.

`double` **`Ftc`**

The Ftc is frequency summation of all candidate terms within the corpus.

`String` **`domain`**

The domain name.

`double` **`docFreqA`**

The document frequency of term a is number of documents the term a appear in.

`double` **`TermFreqAD`**

The Termfreqad is frequency of term a over the target domain d.

`double` **`TermFreqADnot`**

The Termfreqadnot frequency of term a over the rest of corpus.

`double` **`DWa`**

The Dwa is the discriminative weight of term a.

`double` **`ACDWa`**

The ACDwa is the average contextual discriminative weight of term a.

`int` **`termLength`**

The term length.

`Vector<Vector<String>>` **`terms`**

The terms is the terms vector.

`Vector<Vector<String>>` **`docIter`**

The dociter is the document iteration of term a.

`Vector<Vector<String>> termIter`

The termiter is the term iteration in corpus.

`File [] domainList`

The domain list.

`int maxToken`

The max token.

`TermIndexRetreval index`

The index is the calling for term index retrieval class.

Constructor Detail

TermsRanker

```
public TermsRanker(File [] domList,
 int courpusFiles,
 int totalCandidateTermOcurance)
```

Instantiates a new terms ranker.

Parameters:

`domList` - the domain list

`courpusFiles` - the corpus files

`totalCandidateTermOcurance` - the total candidate term occurrence

Method Detail

ACC

`private double ACC(String a)`

Acc is a method to calculate the adjusted contextual contribution.

Parameters:

`a` - is the term to be evaluated.

Returns:

the double

ACDW

`private double ACDW(String a)`

Acdw is a method to calculate the average contextual discriminative weight of term a .

Parameters:

`a` - is the term to be evaluated.

Returns:

the double

---

documentFrequency

`private int documentFrequency(String a)`

Document frequency is a method to calculate the document frequencies of term a .

Parameters:

`a` - is the term to be evaluated.

Returns:

the int

---

domainNotTermFrequency

`private int domainNotTermFrequency(String a)`

Domain not term frequency.

Parameters:

`a` - the term a

Returns:

the int

---

domainTermFrequency

`private int domainTermFrequency(String a)`

Domain term frequency.

Parameters:

`a` - the term a

Returns:

the int

DP

`private double DP(String a)`

Dp is a method to calculate the domain prevalence for term a.

Parameters:

`a` - the term a

Returns:

the double

DPh

`private double DPh(String h)`

Dph is a method to calculate the domain prevalence for term header h .

Parameters:

`h` - the term header h

Returns:

the double

DT

`private double DT(String a)`

Dt is a method to calculate the domain tendency for term a.

Parameters:

`a` - the term a

Returns:

the double

DW

`private double DW(String a)`

Dw is a method to calculate the discriminative weight for term a.

Parameters:
`a` - the term a
Returns:
the double

---

log2

```
private double log2(double num)
```

Log2.
Parameters:
`num` - the num
Returns:
the double

MF

```
private double MF(String a)
```

Mf is a method to calculate the modifier factor for term a.
Parameters:
`a` - the term a
Returns:
the double

NGD

```
private double NGD(String a,
 String c)
```

Ngd is a method to calculate the normalized google distance between term a and term c .
Parameters:
`a` - the term a
`c` - the term c
Returns:
the double

---

rank

```
public double rank(String a,
 String domain1,
 Vector<Vector<String>> termCandidate,
 Vector<Vector<String>> docIteration,
 Vector<Vector<String>> termIteration)
```

Rank is a method to calculate the total rank for term a.
Parameters:
`a` - the term a
`domain1` - is the term domain name
`termCandidate` - is the term candidate
`docIteration` - is the document iteration
`termIteration` - is the term iteration
Returns:
the double

sim

```
private double sim(String a,
 String c)
```

Sim is a method to calculate the similarity between term a and term c .

Parameters:

`a` - the term a

`c` - the term c

Returns:

the double

TH

```
private double TH(String a)
```

Th is a method to calculate the termhood for term a.

Parameters:

`a` - the term a

Returns:

the double

v. Class TermDistriputionProcess

```
java.lang.Object
 └AADRTE.TermDistriputionProcess
```

```
public class TermDistriputionProcess
extends Object
```

The Class TermDistriputionProcess. a simple distribution process for distributing the terms over the domains depending on there rank value.

Field Summary

Modifier and Type	Field and Description
(package private) Vector<Vector<String>>	**domainWords** The domain words.
(package private) TermIndexRetreval	**index** instantiate The index retrievers.
(package private) int	**maxtoken** The maxtoken.
(package private) int	**minRank** The minrank is the minimum rank value that could be accepted for the term to be distributed.
(package private) static Vector<Vector<String>>	**rankingVector** The ranking vector.
(package private) static Vector<Vector<String>>	**termsVector** The terms vector.
(package private) VectorToFileWriter	**writer** instantiate The vector to file writer.

Constructor Summary

Constructor and Description
TermDistriputionProcess(Vector<Vector<String>> rankVector, Vector<Vector<String>> termCandidate) Instantiates a new term distribution process.

Method Summary

Modifier and Type	Method and Description
private void	**distriputeTheVector**(int i, Vector<String> x) Distribute the vector.
void	**startDistripution**(File [] domainList, int maxtokens) Start distribution.

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

rankingVector

`static Vector<Vector<String>> rankingVector`

The ranking vector.

---

termsVector

`static Vector<Vector<String>> termsVector`

The terms vector.

---

domainWords

`Vector<Vector<String>> domainWords`

The domain words.

---

maxtoken

`int maxtoken`

The maxtoken.

---

minRank

`int minRank`

The minrank is the minimum rank value that could be accepted for the term to be distributed.

---

writer

`VectorToFileWriter writer`

instantiate The vector to file writer.

---

index

`TermIndexRetreval index`

instantiate The index retrievers.

Constructor Detail

TermDistriputionProcess

```
public TermDistriputionProcess(Vector<Vector<String>> rankVector,
 Vector<Vector<String>> termCandidate)
```

Instantiates a new term distribution process.

Parameters:

`rankVector` - the rank vector

`termCandidate` - the term candidate

Method Detail

distriputeTheVector

```
private void distriputeTheVector(int i,
 Vector<String> x)
```

Distribute the vector. This class distributes the candidate terms over the domain depending on there rank value for the domains and assign the term to the domain with higher rank value.

Parameters:

`i` - is the number of the vector to be distributed

`x` - is the vector contains the distributed terms

---

startDistripution

```
public void startDistripution(File [] domainList,
 int maxtokens)
 throws IOException
```

Start distribution.

Parameters:

`domainList` - the domain list

`maxtokens` - the max tokens

Throws:

`IOException` - Signals that an I/O exception has occurred.

vi. Class TestingStage

```
java.lang.Object
 └AADRTE.TestingStage
```

```
public class TestingStage
extends Object
```

The Class TestingStage.

Constructor Summary

Constructor and Description
`TestingStage(Vector<Vector<String>> domainWords, int maximumTermLength, String testingCorpus)` Instantiates a new testing stage.

Method Summary

Modifier and Type	Method and Description

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

TestingStage

```
public TestingStage(Vector<Vector<String>> domainWords,
 int maximumTermLength,
 String testingCorpus)
 throws IOException
```

Instantiates a new testing stage.

Parameters:

`domainWords` - is the domain words matrix

`maximumTermLength` - is the maximum term length

`testingCorpus` - is the folder name for the testing corpus

Throws:

`IOException` - Signals that an I/O exception has occurred.

vii. Class ClassifyDocument

```
java.lang.Object
  └AADRTE.ClassifyDocument
```

```
public class ClassifyDocument
extends Object
```

The Class ClassifyDocument is used to classify the document within the Testing_corpus folder. These document should be butted in folders represents there domain.
The classification Process begins with loading the Modified light stemmer class which loads the stemmer files that contains the stop word and other preprocessing files from the StemmerFiles folder.
Then it loads the distributed domain word matrix from the DistriputedDomainTerms folder. Each document represent a domain.
After that the classifier compute the binary distance between the document vector and the domain vectors and give the document the domain with high distance.

Field Summary

Modifier and Type	Field and Description
`(package private) int`	**dom**
`(package private) int`	**domainsNumber**
`private Vector<Vector<String>>`	**domainTerms**
`(package private) int`	**domRankMax**
`(package private) TermIndexRetreval`	**index**
`(package private) int`	**maxToken**
`private Vector<String>`	**stemedFileVector**
`private ModifiedLightStemmer`	**stemmer**

Constructor Summary

Constructor and Description
ClassifyDocument`(Vector<Vector<String>> domainwords, int maxTerm)` Instantiates a new classify document.

Method Summary

Modifier and Type	Method and Description
`int`	**classify**`(String fileName)` Classify method.

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

stemmer

```
private ModifiedLightStemmer stemmer
```

---

index

```
TermIndexRetreval index
```

---

domainTerms

```
private Vector<Vector<String>> domainTerms
```

---

stemedFileVector

```
private Vector<String> stemedFileVector
```

---

maxToken

```
int maxToken
```

---

domainsNumber

```
int domainsNumber
```

---

dom

```
int dom
```

---

domRankMax

```
int domRankMax
```

Constructor Detail

ClassifyDocument

```
public ClassifyDocument(Vector<Vector<String>> domainwords,
 int maxTerm)
```

Instantiates a new classify document.

Parameters:

`domainwords` - is the vectors of domain term matrix .

`maxTerm` - is the maximum number of words in the term.

Method Detail

classify

```
public int classify(String fileName)
 throws IOException
```

Classify method.

Parameters:`FileName` - is the name of the file to be classified

Returns:

This method returners the document domain as an integer value depending on the number of domains within the corpus folder.
Throws: `IOException` - Signals that an I/O exception has occurred.

F. An Example of ranking the term يوهانسون

```
Start The ranking process for يوهانسون as one word term.
Corpus Files : 17759
Total Candidate Term Occurrence :22702550
Document Frequency of (يوهانسون) is: 1
Domain Term Frequency of (يوهانسون) is = 1
Domain Not Term Frequency of (يوهانسون) is = 0
Domain Term Frequency of (يوهانسون) is = 1
Domain Term Frequency of (يوهانسون) is = 1
Domain Not Term Frequency of (يوهانسون) is = 0
DP(يوهانسون) is:40.61552195880243
Domain Term Frequency of (يوهانسون) is = 1
Domain Not Term Frequency of (يوهانسون) is = 0
DT(يوهانسون) is:1.5849625007211563
DW(يوهانسون) is:64.37407925191854
Domain Term Frequency of (يوهانسون) is = 1
Domain Term Frequency of (يوهانسون) is = 1
Domain Not Term Frequency of (يوهانسون) is = 0
DP(يوهانسون) is:40.61552195880243
Domain Term Frequency of (يوهانسون) is = 1
Domain Not Term Frequency of (يوهانسون) is = 0
DT(يوهانسون) is:1.5849625007211563
DW(يوهانسون) is:64.37407925191854
ACDW(يوهانسون) is :64.37407925191854
Acc(يوهانسون) is :64.37407925191854
The rank of term(يوهانسون) is:128.7481585038371
ranking اقتصاد | يوهانسون Rank value=128
End of the Experiment
```

G. An Example of ranking the term يطرح أسهم شركة مدمجة

Start The ranking process for يطرح أسهم شركة مدمجة `as four words term.`

Corpus Files: 17759

Total Candidate Term Occurrence: 22702550

```
Document Frequency of (مدمجة شركة أسهم يطرح) is: 1
Domain Term Frequency of (مدمجة شركة أسهم يطرح) is = 1
Domain Not Term Frequency of (مدمجة شركة أسهم يطرح) is = 0
Domain Term Frequency of (مدمجة شركة أسهم يطرح) is = 1
Domain Term Frequency of (يطرح ) is = 42
Domain Term Frequency of (يطرح ) is = 42
Domain Not Term Frequency of (يطرح ) is = 193
DP(يطرح ) is:45.35437395692971
Domain Term Frequency of (يطرح) is = 42
Domain Not Term Frequency of (يطرح) is = 193
Domain Term Frequency of (أسهم) is = 5094
Domain Not Term Frequency of (أسهم) is = 2798
Domain Term Frequency of (شركة) is = 4568
Domain Not Term Frequency of (شركة) is = 3018
Domain Term Frequency of (مدمجة) is = 8
Domain Not Term Frequency of (مدمجة) is = 5
MF(مدمجة شركة أسهم يطرح) is:1.386698584277142
DP(مدمجة شركة أسهم يطرح) is:150.810458492347
Domain Term Frequency of (مدمجة شركة أسهم يطرح) is = 1
Domain Not Term Frequency of (مدمجة شركة أسهم يطرح) is = 0
DT(مدمجة شركة أسهم يطرح) is:1.5849625007211563
DW(مدمجة شركة أسهم يطرح) is:239.02892142693446
Domain Term Frequency of (يطرح ) is = 42
Domain Term Frequency of (يطرح ) is = 42
Domain Not Term Frequency of (يطرح ) is = 193
DP(يطرح ) is:45.35437395692971
Domain Term Frequency of (يطرح ) is = 42
Domain Not Term Frequency of (يطرح ) is = 193
DT(يطرح ) is:0.0
DW(يطرح ) is:0.0
Document Frequency of (يطرح ) is: 198
Document Frequency of (مدمجة شركة أسهم يطرح) is: 1
Document Frequency of (مدمجة شركة أسهم يطرح) is: 1
(Math.max(0.0,5.288267030694535)- 0.0)/(9.784647708654596-Math.min(0.0,
5.288267030694535))
NGD(مدمجة شركة أسهم يطرح,يطرح )=0.5404657569855092
Sim(مدمجة شركة أسهم يطرح,يطرح ) is:0.45953424301449075
Domain Term Frequency of (أسهم ) is = 5094
Domain Term Frequency of (أسهم ) is = 5094
Domain Not Term Frequency of (أسهم ) is = 2798
DP(أسهم ) is:68.02777610376341
Domain Term Frequency of (أسهم ) is = 5094
Domain Not Term Frequency of (أسهم ) is = 2798
DT(أسهم ) is:1.0
DW(أسهم ) is:68.02777610376341
Document Frequency of (أسهم ) is: 2244
Document Frequency of (مدمجة شركة أسهم يطرح) is: 1
Document Frequency of (مدمجة شركة أسهم يطرح) is: 1
(Math.max(0.0,7.716015266642587)- 0.0)/(9.784647708654596-Math.min(0.0,
7.716015266642587))
NGD(مدمجة شركة أسهم يطرح,أسهم )=0.7885838607983517
Sim(مدمجة شركة أسهم يطرح,أسهم ) is:0.21141613920164826
Domain Term Frequency of (شركة ) is = 4568
Domain Term Frequency of (شركة ) is = 4568
Domain Not Term Frequency of (شركة ) is = 3018
DP(شركة ) is:67.49336833015924
Domain Term Frequency of (شركة ) is = 4568
Domain Not Term Frequency of (شركة ) is = 3018
DT(شركة ) is:1.0
DW(شركة ) is:67.49336833015924
Document Frequency of (شركة ) is: 1585
Document Frequency of (مدمجة شركة أسهم يطرح) is: 1
Document Frequency of (مدمجة شركة أسهم يطرح) is: 1
(Math.max(0.0,7.368339686311381)- 0.0)/(9.784647708654596-Math.min(0.0,
7.368339686311381))
NGD(مدمجة شركة أسهم يطرح,شركة )=0.7530510965452571
Sim(مدمجة شركة أسهم يطرح,شركة ) is:0.24694890345474285
Domain Term Frequency of (مدمجة ) is = 8
Domain Term Frequency of (مدمجة ) is = 8
Domain Not Term Frequency of (مدمجة ) is = 5
DP(مدمجة ) is:41.54344097198535
Domain Term Frequency of (مدمجة ) is = 8
Domain Not Term Frequency of (مدمجة ) is = 5
DT(مدمجة ) is:1.0
DW(مدمجة ) is:41.54344097198535
Document Frequency of (مدمجة ) is: 12
Document Frequency of (مدمجة شركة أسهم يطرح) is: 1
Document Frequency of (مدمجة شركة أسهم يطرح) is: 1
(Math.max(0.0,2.4849066497880004)- 0.0)/(9.784647708654596-Math.min(0.0,
2.4849066497880004))
NGD(مدمجة شركة أسهم يطرح,مدمجة )=0.2539597463064594
Sim(مدمجة شركة أسهم يطرح,مدمجة ) is:0.7460402536935407
ACDW(مدمجة شركة أسهم يطرح) is :15.510665580993571
Acc(مدمجة شركة أسهم يطرح) is :-1.81562164476214E-5
The rank of term(مدمجة شركة أسهم يطرح) is:239.02890327071802
```

ranking اقتصاد | مدمجة شركة أسهم يطرح Rank value is : 239

End of the Experiment