Addis Ababa University
College of Natural Sciences

Design of Hidden Web Crawler Using Word2vec Model

Engdawerk Kebede Birru

A Thesis Submitted to the Department of Computer Science in Partial Fulfillment for the Degree of Master of Science in Computer Science

Addis Ababa, Ethiopia
October 09/2020

Addis Ababa University

College of Natural Sciences

Engdawerk Kebede Birru

Advisor: Fekade Getahun (Ph.D.)

This is to certify that the thesis prepared by *Engdawerk Kebede Birru*, titled: Design of Hidden Web Crawler Using Word2vec Model and submitted in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the Examining Committee:

Name	**Signature**	**Date**
Advisor: Fekade Getahun (Ph.D.)	_____	
Examiner: Solomon Atnafu (Ph.D.)	_____	
Examiner: Minale Ashagre (Ph.D.)	_____	

Dedicated to:

My family

Abstract

World Wide Web (WWW) is a huge repository of hyperlinked documents containing useful information. WWW can be broadly classified in two type's i.e. surface web and hidden web from the user's point of view. The surface web consists of static hyperlinked web pages that can be crawled and index by general search engine. On the other hand the hidden web refers to the dynamic web pages which can be accessed through specific query interfaces. Web crawler is program that is specialized in downloading web contents. Conventional web crawler can easily search and analyze the surface web having interlinked html pages but they have the limitations in fetching the data from deep web due to the query interface. To access deep web, a user must request for information from a particular database through a query interface.

Traditional web crawler can easily crawl surface web, but not able to crawl the hidden portion of the web. These traditional crawlers retrieve contents from web pages, which are linked by hyperlinks ignoring the information hidden behind form pages, which cannot be extracted using simple hyperlink structure. Thus, they ignore large amount of data hidden behind search forms.

In this work, we propose a hidden web crawler using word2vec model the proposed crawling approach use e-commerce product review text word2vec model to extract relevant keyword from e-commerce hidden web page. Once automatically extract keywords considering semantics relatedness between words to fill fields of a hidden web form leads to more accurate and relevant result. The results of the proposed approach are analyzed and found as per our expectation.

Keywords: Hidden Web, Surface web, Hidden Web Crawler, word2vec model.

Acknowledgments

First of all, I would like to say thanks for the people who were part of this thesis in many ways. My teachers and my friends made valuable suggestions which have been incorporated in this work. It is not possible to acknowledge all of them individually. I would like to take this opportunity to present my profound gratitude to them. And I like to thank God and gratefully acknowledge the help and support of my lord in my whole life for giving me the direction, strength and knowledge in exploring thing in this world.

Beside this I would like to express my gratitude and I am deeply thankful to my advisor, **Dr. Fekade Getahun.** (Ph.D.) for his kind cooperation, excellent suggestions, moral support and guidance, continuous help, and assistance during this thesis work. I would like to express my special thanks to my best friends and staff members, Bezawit Lakew, Hiwot Wenago, Ashagre Tibebu and Addis Ababa university computer science graduate program teachers for their valuable support throughout the course, advice and follow-ups. The love, affection and encouragement that I got from my family members especially my father is unforgettable, He is my strength and my pride.

Table of Contents

List of Figures iv

List of Tables v

Lists of Algorithm vi

Acronyms vi

CHAPTER 1: INTRODUCTION 1

1.1 Background 1

1.2 Motivation 2

1.3 Statement of the Problem 3

1.4 Objectives 4

1.5 Methods 5

1.6 Scope and Limitations 5

1.7 Application Result 6

1.8 Thesis Outline 6

CHAPTER 2: LITERATURE REVIEW 7

2.1. INTRODUCTION 7

2.2 INFORMATION RETRIEVAL 7

2.2.1. Approaches of Information Retrieval 8

2.2.2. Evaluation of information retrieval 9

2.3 WEB SEARCH ENGINE 10

2.4 SURFACE WEB vs HIDDEN WEB 12

2.4.1 The Surface Web 13

2.4.2 The Hidden web 13

2.5 WEB CRAWLER 14

2.5.1 Hidden web Crawler 15

2.5.2 Challenges of hidden web crawler 16

2.6 Keyword extraction techniques 17

2.7. WORD EMBEDING 18

CHAPTER 3: RELATED WORK 25

3.1 A HIDDEN WEB CRAWLER FOR MEDICAL DOMAIN 25

3.2 GOOGLE'S DEEP-WEB CRAWL 25

3.3. DEEPBOT: A FOCUSED CRAWLER FOR ACCESSING HIDDEN WEB CONTENT 26

3.4 SIPHONING HIDDEN-WEB DATA THROUGH KEYWORD-BASED INTERFACES 26

3.5. A NEW HIDDEN WEB CRAWLING APPROACH 27

3.6. CRAWLING THE HIDDEN WEB: AN APPROACH TO DYNAMIC WEB INDEXING 27

3.7. CRAWLING THE HIDDEN WEB 28

3.8. DOWNLOADING TEXTUAL HIDDEN WEB CONTENTS TROUGH KEYWORD QUERIRES 28

3.9. DESIGN AND IMPLMENTATION OF SEMANTIC HIDDEN WEB CRAWLER 28

3.10. DESIGN OF A DARK WEB CRAWLER AND OFFLINE LANGUAGE IDENTIFIER FOR AMHARIC DOCUMENTS 29

3.8 Summary 31

CHAPTER 4: DESIGN OF HIDDEN WEB CRAWLER USING WORD2VEC MODEL 32

4.1. Overview 32

4.2. Hidden web crawling challenges 32

4.3. System Architecture 33

4.3.1 Form detector 35

4.3.2 Pre-processing 36

4.3.3. Word2vec Embedding 40

4.4 Clustering 42

4.4.1 Select keywords 44

4.4.2 Automatic form filler and submitter 44

4.5 Summary 45

CHAPTER 5: IMPLEMENTATION AND EXPERIMENTAL RESULTS 46
5.1 Overview 46
5.2 Dataset preparation 46
5.3. Word2vec model 47
5.4 Implementation 48
5.5 System Prototype 48
5.6 Evaluation 50
CHAPTER 6: CONCLUSION AND FUTURE WORK 52
6.1 CONCLUSION 52
6.2 CONTRIBUTION OF THIS WORK 52
6.3 FUTURE WORK 53
Reference 54

List of Figures

Figure 2-1 Components of a Web Search Engine 11

Figure 2-2 High level architecture of crawler 15

Figure 2-3 CBOW model architecture 22

Figure 2-4 Skip-gram model architecture 24

Figure 4-1 Architecture of hidden web crawler 34

Figure 4-2 Architecture of word2vec model 42

Figure 5-1 Example of contextually similar words 47

Figure 5-2 Hidden web crawler interface prototype 48

Figure 5-3 Search interface of Alibaba e-commerce web page: www.alibaba.com 49

Figure 5-4 Result page of web crawler after submitting electronics keyword electronics html result 49

List of Tables

Table 2.1. Example of count vector…….....19

Table 3.1. Related work summary 36

Table 5.1. Evaluation of keyword extraction 50

Lists of Algorithm

Algorithm 4.1: Filter text from web page…….41

Algorithm 4.2: Tokenization algorithm.…….42

Algorithm 4.3: Stop word removal……...43

Algorithm4.4: Normalization Algorithm……..44

Algorithm 4.5: Frequency count algorithm……...45

Algorithm 4.6: Hierarchical agglomerative clustering algorithm……....48

Algorithm 4.7: Automatic Form frequency……....49

Acronyms

CBOW	Continuous Bag of word
F	F-measure
HW	Hidden Web
HTML	Hypertext markup language
NLP	Natural language processing
P	Precision
PIW	publicly indexable web
R	Recall
SGD	stochastic gradient descent
SG	Skip-gram
TF-IDF	Term frequency -inverse document frequency
URL	Uniform resource locator
WWW	World Wide Web
Word2vec	Word to vector

CHAPTER 1: INTRODUCTION

1.1 Background

The World Wide Web (WWW) [16, 17] is a huge repository of hyperlinked documents containing useful information. In the recent years, the exponential growth of the information technology has led to large amount of information available through the WWW. The searching of WWW for the useful and relevant information has become more challenging as the size of the web continues to grow. The WWW can broadly be divided in two type's i.e. surface web and deep web according to their depth of data [56]. As presented in Argawal et al.[17], Surface web is also called as open web, visible web or indexable net which can be accessed by the traditional search engines.

Search engines access web pages which are linked to one another and are generally static and accessible to all web users. Deep web is also known as 'invisible web' as these pages are out of reach from traditional search engines. It is also called as hidden web, deep net or under net. In 2001, Bergman coined it as "Deep Web"[59]. In brief, deep web, part of Internet, which can be defined as the information concealed behind Hyper Text Markup Language (HTML). It is also believed that deep web is a vast source of methodized content on the World Wide Web and to access contents of deep web has been a challenge in the web community.

Most important components of search engine to access web page is Web crawler which is a program that is specialized in downloading web content. The author in [18] argues that, it is basically software that starts from a set of seed URLs (Uniform Resource Locater), and downloads all the web pages associated with these URLs. After fetching a web page associated with a URL, the URL is removed from the working queue. The web crawler then parses the downloaded page, extracts the linked URLs from it, and adds to the list of seed URLs. This process continues iteratively until all of the contents reachable from seed URLs are reached. This traditional definition of a web crawler assumes that all the content of a web application is reachable through URLs [18]. However, soon in the history of web crawling, it became clear that such web crawlers cannot deal with the complexities added by interactive web applications that

rely on user input to generate web pages. This scenario often arises when the web application is an interface to a database and it relies on user input to retrieve contents from the database. The new field of hidden Web-Crawling was born to address this issue.

So that, to access hidden web data one needs to fill various search forms available on different websites. There are two ways to fill these field values, First is that a user manually sits on each website fill the required information and get the result, the other way is to design a specialized crawler which automatically fill these values.

A lot has been done on hidden web crawling but most of the works consider only hidden web with structured database which is easy to fill form by simply matching label name and task specific database. There also works on crawling hidden web behind simple search interfaces that accept keyword-based queries. But they require human effort for an initial start of the crawler and do not consider semantic relatedness between words while select keywords.

In this paper, we study how we can build a Hidden-Web crawler that can automatically download pages from the Hidden Web behind simple search interface and provide a design of hidden web crawler using word2vec which can extract relevant keywords from web page for automatic submission of keyword-based queries.

1.2 Motivation

Search engines have become one of the most important services in the web that facilitate the organization of data in the Internet. Increasingly advanced features are being developed to improve the standard of service provided to search engine users. However, there is a hidden web which is inaccessible to most search engine crawlers, which means that many users are unaware of its rich content. According to Internet studies, shopping is one wide part of Internet that covers of almost 35 % of it [20]. Now, thinking as a user, if one wants to shop on Internet for a particular product, the best way to begin is to start browsing the Websites popular in shopping. For example amazon.com, ebay.com, bestbuy.com and many more. More likely a shopper will surf for like 3 different shopping Websites before he decide as to where to buy. Now imagine why not to use the search engines to shop. Each time a buyer has to visit at least 3 different shopping Websites, making searches and gathering all the information he needs. Observing

closely, the buyer is manually doing the work of the Web Crawler, which is visiting the shopping Websites to gather necessary information and then picking up the right one for him. General purpose search engines crawls only what we call as surface Web. Whenever any WebCrawler visits a Web server, they gather information of all the Web pages been stored on that Web server. They are not able to penetrate deep into the Web server to access their databases and use the Web services served by that particular Web server to retrieve information. According to studies conducted in 2000, deep Web contains 7500 terabytes of information in comparison to 19 terabytes on surface Web. There are more than 200,000 deep Web sites [21]. There is a need for Hidden Web (HW) crawler which can crawl and extract this vast source of information – hidden web.

1.3 Statement of the Problem

Web crawlers are programs that traverse the web and automatically download pages for search engines. Working of conventional crawler mainly depends on the hyperlink on the publically indexable web to discover and download page. Due to the lack of links pointing to the hidden web pages, the current search engines are unable to index the hidden web pages. In order to access Hidden Web pages, one must fill an HTML form which will be submitted as a query to the underlying database. The basic requirement in designing an efficient hidden web crawler is identification of entry points, automatic relevant query selection, and automatic form submission and download response page.

Identification of entry points**:** identification of a proper entry point among the searchable forms is a major issue for crawling the hidden web contents [22][23][24].If entry to the hidden web pages is restricted only through querying a search form, two challenges arises i.e. to understand and model a query interface as well as handling of these query interfaces with significant quires.

The hidden web crawler [25, 26] crawl a web page with single attribute, such as keyword-based search interface. They extract seed keywords from the words on the form page, by identifying the words most relevant to its contents using TF-IDF Information Retrieval measure. Then they select additional keywords from the words of the web pages generated by the form submissions.

However, TF-IDF only mines information according to word frequency and inverse document frequency and cannot reflect the semantic information of the words. As the goal is to download the maximum number of unique web pages, selecting keywords only by their frequency may not give good results.

There are also research works that design a hidden web crawler [27-29]. Their works focus on the hidden web with multi-attribute form interfaces for structured data sources, which is easy to fill form by matching label name and task specific database. But they ignore forms with single-attribute.

This thesis will addresses problems towards accessing hidden web contents and provides a hidden web crawler which identify proper entry point of hidden web and select relevant query by clustering the semantically similar words that have more coverage of the content of the web page. To calculate semantic similarity we will use word2vec model. So that selecting keywords from clustered words by their semantic similarity help to download unique hidden web pages.

1.4 Objectives

GENERAL OBJECTIVE

The general objective of this thesis is to design a hidden web crawler using word2vec model.

SPECIFIC OBJECTIVE

In order to achieve the general objective we identify the following specific objectives

- Review related literature in the area of information retrieval, surface web, hidden web, and web crawler
- Study specific hidden web crawler requirements
- Study word2vec model with respect to keyword extraction from a web page
- Collect a data set to train word2vec model
- Design the architecture of hidden Web crawler.
- Design Prototype

- Collect a data set for test and evaluation purpose
- Evaluate the proposed hidden web crawler.

1.5 Methods

In order to achieve the objectives of this study, the methods mentioned below will be used.

Literature Review

In order to get better knowledge and understanding we will make through review of different related work on the area of hidden web crawling this helps to understand challenges of hidden web crawler and techniques in keyword extraction.

Data collection

The data needed for this research will be collected from different online source and from deep websites that can be used for test and evaluation purpose later on.

Tools and Development Environments

Different free and open source tools will be used during prototype development. Python programming language with Jupyter notebook will be used. The word2vec model will be trained in Anaconda with genism python library.

Experimentation and Testing

The proposed solution will be evaluated in terms of its goals and performance using different measuring matrices like precision and recall.

1.6 Scope and Limitations

The scope of this research is designing a hidden web crawler and identifies relevant keywords from web page itself. We will not consider indexing and ranking of hidden web data.

1.7 Application Result

The result of this study will allow an average Web user to easily explore the vast amount of information that is mostly “hidden” at present. Since a majority of Web users rely on search engines to discover pages, when pages are not indexed by search engines, they are unlikely to be viewed by many Web users. Unless users go directly to Hidden-Web sites and issue queries there, they cannot access the pages at the sites. And it will significantly reduce the user wasted time and effort while visiting all potentially relevant site.

1.8 Thesis Outline

The rest of this thesis report is organized as follows. Chapter 2 reviews concepts relevant to the realm of the proposed approach. Chapter 3 presents related works. Chapter 4 presents the proposed design of hidden web crawler using word2vec model. Prototype development and experimental results are presented in Chapter 5. Finally, Chapter 6 concludes the overall work presented in this research work, presents contributions of the study and draws future directions.

CHAPTER 2: LITERATURE REVIEW

2.1. INTRODUCTION

This Chapter presents a review of concepts relevant to obtain basic understanding of the ideas of the proposed research work and concepts related to research works towards hidden web crawler using word2vec and clustering for keyword extraction. The review presented in this chapter deals on concepts important to the realm of the proposed approach. Specifically, concepts on information retrieval, web search engine, hidden web, web crawler, keyword extraction technique and other related issues are presented in detail.

2.2 INFORMATION RETRIEVAL

Web information retrieval is a technology for helping users to accurately, quickly, and easily find information on the Web. The World Wide Web is a very large provider of digital information space. From its origins in 1991 as an organization-wide collaborative environment at CERN [29] for sharing research documents in nuclear physics, the Web has grown to encompass diverse information resources: personal home pages; online digital libraries; virtual museums; product and service catalogs; government information for public dissemination; research publications; and mail servers.

As presented in [30] Web documents are known as 'pages', each of which can be addressed by an identifier called a uniform resource locator (URL). For example: http://www.acm.org/pubs/contents/proceedings/series/sigir/. Web pages are usually grouped into 'sites', sets of pages published together. For example: http://www.acm.org

Pages are remarkably versatile. A web page can play the same roles as a news article in a conventional IR corpus, such as providing information for a report or helping to answer a question — indeed these days most news articles are also published as web pages. However, the Web also contains pages of many other types.

Many web pages exist purely to help users navigate to other pages in the site, such as the 'entry page' or 'home page' of a site. Some pages provide an interactive service such as a search form. Some pages provide a commercial service, allowing users to shop for products online. Some

pages are generated dynamically and intended to be used once, such as a search engine outputs list page. Other pages are front-ends to databases, as result each page represents part of an underlying relational or XML database. That database might be finally useful (http://imdb.com) or very large and specialized (http://www.ebi.ac. uk/flybase/ – A Database of the Drosophila Genome) to the extent it is not clear that a general-purpose engine should index it. And some pages are error pages, indicating that the user has reached a bad URL, and others are intended to direct the user to other pages ('you are now leaving NASA' or 'doorway page to my site'). And some pages exist to contain a media file or an interactive game. Web pages are works in these ways and many more, and we often observe a single page that can be used in several of these ways.

Web search engines find pages by 'crawling' the Web, discovering new pages by following hyperlinks. Access to certain web pages may be restricted in various ways. For example, pages on a corporate intranet may be visible only to those who can authenticate themselves as organization employees. Web publishers may also impose additional restrictions on access by search engines, using the robots.txt Robots Exclusion Protocol [31]. They may do this to intercept load on their servers, to help search engines avoid wasting resources crawling ephemeral or low value content, or for commercial reasons. Generally, search engines can, in general, only crawl pages which are linked to from pages already in the crawl. A page with no incoming links is extremely unlikely to be indexed by a general Web search engine.

The collection of web pages which cannot be included in search engine indexes is most a time called 'the hidden web', 'the deep web', or 'web dark matter' [32].

2.2.1. Approaches of Information Retrieval

Generally, there are two main categories of IR technology and research. These are Semantic and Statistical. Semantic approaches attempt to implement some level of syntactic and semantic analysis; In other words, they try to reproduce to some (perhaps modest) degree the understanding of the natural language text that a human user would provide [33]. And in statistical method, the documents that are retrieved or that are highly ranked are those that match the query most closely in terms of some statistical measure. There are different models in statistical approaches; including Boolean model, Extended Boolean model, Vector space

mode and probabilistic model. Mostly to all statistical approaches query terms are collected from the entire text document and measured statistically. Then the terms usually undergo into preprocessing operation. The Preprocessing operation like stemming, stop word removal and root form formation words for every word is applied. This enhances efficiency, accuracy and also eliminates the ambiguity that arises from the occurrence of different grammatical forms of the same word, e.g., “research,” “researched,” “researches,” and “researching” should all be recognized as forms of the same word [34].

2.2.2. Evaluation of information retrieval

As presented in [35], there are a number of ways for measuring performance. Four standard retrieval (IR) evaluate of performance are precision, recall, overload and mismatch.

Precision is a standard IR measure of performance. It is set out as the number of relevant document retrieved divided by the total number of document retrieved:

Precision=number of relevant document retrieved/Total number of documents retrieved. The goal of an IR system is to achieve 100% precision, however as this can be achieved by returning only one document the system should also try to maximize recall. Recall is a standard IR measure of performance. It is described as the number of relevant documents retrieved divided by the total number of relevant documents in the collection:

Recall=number of relevant document retrieved / Total number of relevant documents in the collection

The goal of an IR system is to achieve 100% recall. However as this can be achieved by returning all the documents, the system should also try to maximize precision as well. Recall can be improved by selecting collections that retrieve different relevant documents [36].

Overload is where returned documents are not relevant. Overload is defined as the number of irrelevant documents retrieved divided by the total number of irrelevant documents in the collection:

Overload=number of irrelevant document retrieved/Total number of irrelevant documents in the

collection

Overload is made worse if the irrelevant documents returned are highly rated.

Mismatch is where relevant documents are retrieved from the collection. Mismatch is defined as the number of relevant document not retrieved divided by the total number of relevant documents:

Mismatch=number of relevant documents not retrieved/Total number of relevant documents in the collection

Mismatch is difficult to avoid when working with short or unspecific queries.

2.3 WEB SEARCH ENGINE

First off, it's crucial to know that when a person performs a web search, he's actually not searching the web but the search engine's index of the web. Due to speed, costs, and capabilities, it is plain not possible to search through all the web pages every time a user clicks 'search' on an engine [37].Search engine is a computer program that searches for particular keywords and returns a list of documents in which they were found, especially a commercial service that scans documents on the Internet. A search engine finds information for its database by accepting listings sent in by author's who want exposure, or by getting the information from their "Web crawlers," "spiders," or "robots," programs that roam the Internet storing links to and information about each page they visit.

The steps to search a specific piece of information by a general query are as follows:

An Internet user submits a query by typing a phrase or word in the search box.

1. Regarding the query, search engine looks through all the pages that it keeps in its database.

2. Search engine identifies the relevant web pages

3. Results are displayed on the Search Engine Results Page (SERP) in an order, starting with the most relevant results to least relevant pages.

Within a fraction of second, the whole search process completes but more complex

algorithms are running behind the search process. Web search engines consist of three basic parts: Web crawler, indexer, and query processor. The components and tasks of web search engines, which are illustrated in Figure 2.1.

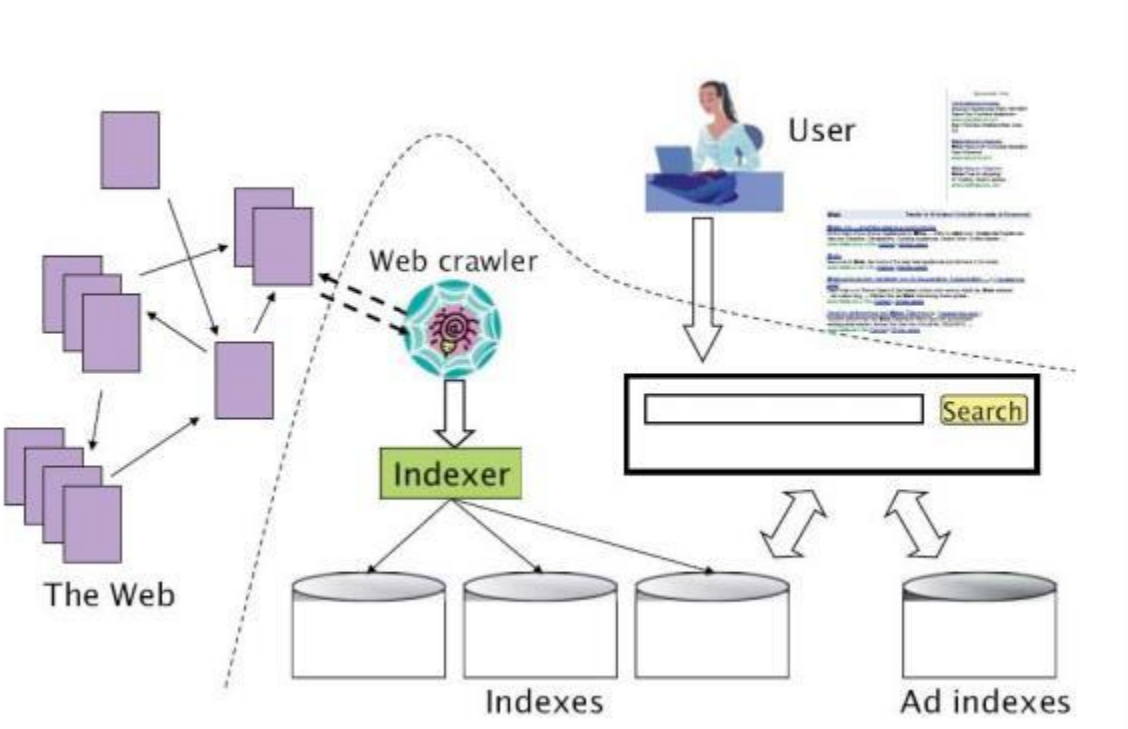

Figure 2-1 Components of a Web Search Engine

Components of search engine

All search engines includes:

1. A Web crawler
2. A parser
3. A ranking system
4. A repository system.
5. A front-end interface.

These components are discussed individually below

The starting point of a Web Crawler is (or spider) to retrieve all Web pages: it simply traverses the entire Web or a certain subset of it, to download the pages or files it encounters and save for

other components to use. The verified traversal algorithm varies depends on the implementation; depth first, breadth first, or random traversal are all being used to meet different design goals.

The parser extracts all downloaded raw results, analyze and eventually try to make sense out of them. In the condition of a text search engine, this is done by extracting keywords and checking the locations and/or frequencies of them. Commonly a scoring system is involved to give a final point for each keyword on each page.

Simple or complicated, a search engine must have a way to determine which pages are more important than the others, and present them to users in a particular order. This is called the Ranking System. The most familiar one is the Page Rank Algorithm published by Google founders.

A reliable repository system is precisely critical for any application. A Search engine also needs everything to be stored in the most efficient way to ensure maximum performance. The choice of database vendor and the schema design can make huge difference on performance for metadata such as URL description, crawling date, keywords, etc. More challenging part is the huge volume of downloaded files to be saved before they are picked up by other modules.

Generally, a front-end interface for users: This is the face and presentation of the search engine. When a user submits a query, mostly in the form of a list of textual terms, an internal scoring function is applied to each Web page in the repository, and the list of outputs is presented, usually in the order or relevance and importance. Google has been known for its simple and straight forward interface, while some most recent competitors, such as Ask.com1, provide much richer user experience by adding features like preview or hierarchy displaying.

2.4 SURFACE WEB vs HIDDEN WEB

From a search engine perspective, the Web can be divided into two parts. The surface Web is the smaller, the visible and the indexable part while the hidden Web contains much more data, is mostly hidden from the search engines and is not trivial to index. The surface web contains those Web documents that the search engines can gather for the public and display them as a query result [38].

2.4.1 The Surface Web

The surface Web is the smaller, the visible and the indexable part of the web. The surface Web contains those Web documents that the search engines can gather for the public and display them as a query result. The technical details have changed during the last decade as some elements, which were listed as hidden Web elements, have become surface Web elements. This phenomenon occurs when the search engines are developed further and the technical obstacle to index a deep Web element has been overcome [38].

2.4.2 The Hidden web

The Hidden Web refers to content hidden behind HTML forms. In order to get to such content, a user has to perform a form submission with valid input values. The Deep Web has been acknowledged as a significant gap in the coverage of search engines because web crawlers employed by search engines rely on hyperlinks to discover new web pages and typically lack the ability to perform such form submissions.

Various accounts have hypothesized that the Deep Web has an order of magnitude more data than the currently searchable World Wide Web [39, 35, and 40]. Furthermore, the Deep Web has been a long-standing challenge for the database community [41, 35] because it represents a large fraction of the structured data on the Web.

The Web content behind forms is referred to as the hidden (deep) Web, which is estimated to comprise several millions of pages**.** The Deep Web is also believed to be the biggest source of structured data on the web and hence accessing its contents has been a big challenge in the data management community [40].According to Bright planet [42], Deep Web contains 400-500 times more information and 15% larger visit capacity than that of Surface Web. Moreover, the quality of data is also relatively higher. Another research indicated that search engines have reached only 0.03 percent of the information that is available; it's like a tip of the iceberg. Bergman argued that still much of the rest of the Web content reside in the Deep web [42].There are many terminology used to describe the "Hidden web", as the "Invisible web" and "Deep web" and the relation with "Dark Net / web" will be discussed on following section.

The Dark web refers to any web page that has been concealed to hide in plain sight or reside within a separate, but public layer of the standard Internet [42]. Darknets are composed of host machines that cannot be accessed by conventional means, which is why the content hosted is typically not indexed by traditional search engines like Google and Bing. Virtual private networks are another aspect of the Dark Net that exists within the public Internet, which often requires additional software to access. The term “Dark Net” interchangeably used to refer “Dark web”; Websites found on Dark Net networks.

2.5 WEB CRAWLER

Search engines use a method called crawling to index the World Wide Web to a database. This means that a program is unleashed to run through the Web. Crawling through the surface Web is mostly made possible by moving from one Web anchor to another and storing that information for retrieval. Anchors are HTML objects that allow the user to move to another point of interest. For example, an anchor tag Peikkoluola would allow a user to enter the Internet site by clicking on the element. Therefore, the basic idea behind a crawler is a trivial one: Select an anchor from a set of candidates, download the associated Web pages, extract the anchors contained in the pages and add those anchors that have not been encountered before to the candidate set. Let us define a Web crawler as follows:

“A Web crawler is a program that given one or more seed URL’s, downloads the Webpages associated with these URLs, extracts any hyperlinks contained in them and recursively continues to download the Web pages identified by extracted hyperlinks”[43].

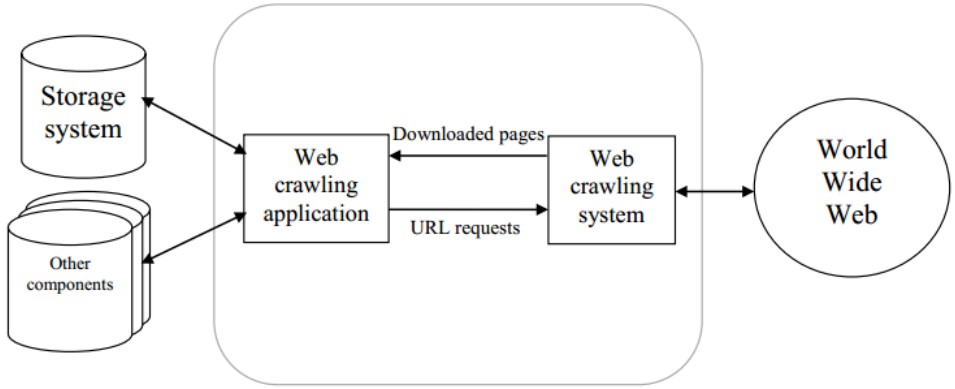

Figure 2-2 High level architecture of crawler

Conventional web crawlers cannot reach to a very significant fraction of the web, which is usually called the "hidden web" or the "deep web". Several works have studied and characterized the hidden web [44][45]. They concluded that it is substantially larger than the publicly indexable web and that it usually contains data of higher quality and with a higher degree of structure.

2.5.1 Hidden web Crawler

The basic actions of a deep web crawler are similar to those of other traditional crawlers. A traditional web crawler selects URL"s, retrieve pages, process the pages and extract links from the retrieved pages. The traditional crawlers do not distinguish between pages with and without forms [34]. Whereas, a Hidden web crawler performs additional sequence of actions for each form on a page [35]:

1) Form detection: the search form extractor looks for any <FORM> tags in the HTML web page to extract the associated search forms.

2) Form Analysis: Parse and process the form to build an internal form representation.

3) Value assignment: Use approximate string matching between the form labels and the labels in the database to generate a set of candidate value

4) Form Submission: send the filled form to the web server.

2.5.2 Challenges of hidden web crawler

The Web is extremely complex than it seems on the surface. The surface web or publicly indexable web (PIW) or is that part of which is recognized by the various search tools. Search engines recognize content on the web by indexing web pages whose contents are stored. On the other hand the deep web or hidden web or invisible web refers to World Wide Web (WWW) content that is not part of the surface web indexed by search engines.

Working of conventional crawler mainly depends on the hyperlink on the publically indexable web (PIW) to discover and download pages. Due to the lack of links pointing to the deep web pages, the current search engines are unable to index the deep web pages.

The major issues and challenges in designing an efficient deep web crawler to crawl the deep web information are given as follows:

Identification of entry points

Identification of a proper entry point among the searchable forms is a major issue for crawling the hidden web contents [22][23][24].If entry to hidden web pages is restricted only through querying a search form ,two challenges arises issues and challenges in designing an efficient deep web crawler to crawl the deep web information are given as follows: i.e. to understand and model a query interface as well as handling of these query interface with significant queries.

Interaction with the search interfaces

An Efficient hidden web crawler [4] should automatically parse process and interact with the form-based search interfaces. A general PIW crawler only submit the request for the url to be crawled where as a hidden web crawler also supply the input in the form of search queries in addition to the request for url.

Handling of form-based queries

Many websites presents query form to the users to further access their hidden database. The general web crawler cannot access these hidden databases due to inability to process these query forms.

2.6 Keyword extraction techniques

Key phrases provide a high level abstraction of a document's content, which play an important role in many areas of document processing, including document indexing, classification, clustering and summarization. A key phrase is a lexical unit or a chain of lexical units that can be a single word, the habitual co-occurrence of two words, or a frequent recurrent uninterrupted string of words [6]. Key phrase extraction is a task that identifies and extracts meaningful lexical unit chains that can describe a given document.

In general, key phrase extraction can be classified into either supervised or unsupervised techniques. Supervised techniques treat key phrase extraction as either a classification [9, 10] or a learning to rank problem [7, 8] requiring annotated training data to build models. In contrast, unsupervised techniques treat key phrase extraction as a ranking problem, scoring each candidate by considering signs such as word occurrence and co-occurrence frequencies, occurrence position, linguistic features, or information from external semantic networks.

An important step, common to both supervised and unsupervised key phrase extraction, is candidate phrase selection. The candidate phrase selection process often involves text cleaning, text normalization, and filtration, such as the removal of stop words and punctuation marks, stemming, tokenizing, part of speech (POS) tagging, and n-gram filtration.

The keywords extraction methodologies are grouped into four main categories: statistical, linguistic, machine learning, and mixed methodologies [2]**.**

Statistical keyword extraction

These methods are simple and do not need the training data. The statistical information of the words can be used to identify the keywords in the document. The techniques such as n-gram and TF measures are used for creating the list of the statistically frequent words.

Linguistic keyword extraction

These approaches use the linguistic features of the words mainly sentences and documents. The linguistic approach includes the lexical analysis, syntactic analysis, semantic analysis, discourse analysis and so on. The linguistic methodologies have shown better performance than the

statistical methodologies [2, 11].

Machine learning keyword extraction

Keyword Extraction can be seen as supervised learning; Machine Learning approach employs the extracted keywords from training documents to learn a model and applies the model to find keywords from new documents. This approach includes Naïve Bayes, Support Vector Machine, etc.

Mixed methodologies

The mixed methodologies involve any combination of statistical, linguistic, and machine learning algorithms. The mixed-methodologies methods often use clustering techniques to group similar words together [12] [13].

2.7. WORD EMBEDING

Word embedding is the numerical representation of words, usually in a shape of a vector in R^d. More specifically, word embedding's unsupervised learned word representation vectors whose relative similarities correlate with semantic similarity. In computational linguistics they are often referred as distributional semantic model or distributed representations [16]**.**Bengio et al. coined the term word embedding's in 2003[2] and trained them in a neural language model. In 2008,Collobert and Weston established word embedding's as a useful tool for downstream tasks and introduced a neural network architecture that forms the foundation form any current approaches [6][17].Word embedding's refers to low dimensional, real valued dense vectors encoding the semantic information of words. Word embedding's a vector representation of the meaning of words. It also acquires more complex geometric structures as a side effect of some algorithms. An example of this is real-world analogies that can be discovered using simple vector arithmetic's: king - man + woman = queen. Word embedding trained on the most popular algorithms learns these vector representations just by scanning big corpora, but it can be trained also on knowledge graphs. Both of these algorithms rely on a single assumption: words that appear in similar contexts have similar contextual meanings.

Frequency based Embedding

In this category there are generally three types of vectors included: count vector, TF-IDF vector and co-occurrence vector.

A. Count Vector

Before modeling every document, it learns the vocabulary from the given documents. The modeling is by finding the frequency of the word in the document. For instance, Given a document D and V the number of unique words in the vocabulary, then the size of the count vector-matrix will be given by D*V. Let us take the following examples: - Doc1: "The book is on the table" Doc2: "The boy is crying". From these documents the vocabularies are {the, book, is, on, table, boy, crying} and D = 2, V = 7 so the matrix is 2X7 number of document * number of unique words.

	The	Book	Is	On	Table	Boy	Crying
Doc1	2	1	1	1	1	0	0
Doc2	1	0	1	0	0	1	1

Table 2-1: example of count vector

B. TF-IDF Vector

This is another method which is based on the frequency method next to the count vector. It takes into account the sense of a text from the occurrence of a word in the entire corpus. Some words may present high frequencies in every document (e.g. "the", "an", "a", "is" in English), those words are low information-bearing (stop-word) in the actual contents of the given document. Terms that are very frequent shadow the frequencies of rarer and more interesting terms. This method removes such a problem by representing or assigning them a lower weight.

C. Co-Occurrence Vector

Words co-occurrence matrix method describes how words occur together. Contextually similar words tend to occur together and this model captures the relationships among those words. By counting the number of word co-occurrences in a given text corpus, it computes words co-

occurrence matrix. To produce more accurate word vector representations, first it preserves the semantic relationship between words. The drawback is requiring a big memory size to store the co-occurrence matrix and factorizing matrix is out of the system.

Prediction based Vector

These methods were a prediction based on the sense that they provided probabilities to the words and proved to be state of the art for tasks like word analogies and word similarities. Word2vec model is used today to generate word vectors.

Word to Vector

It is prediction-based machine learning model which produces a low dimensional real valued vector called word embedding. Word2vec is a two-layer neural network that is used to process text. Its input is unlabeled text corpus and its output is a set of vectors which consists of contextual meaning of words. It turns text into a numerical form that deep neural network can understand but the word vector by itself is not a deep net. It factorizes a word-word matrix that contains co-occurrence counts. In word2vec factorization is done by scanning the corpus with a given window (W) size. The center word in the given window is called target (T) word and a few neighboring words in the left and right of the target word are called the context (C) words. Word vector model starts finding unique vocabulary in a text corpus and then the set of word vectors are initialized randomly and perform a linear scan in the whole document. Then, apply dot products between words and their contextual related words. Besides, it tries to minimize this metric performing SGD (stochastic gradient descent). Each stochastic sample is a consecutive window in the corpus. In each step, a target word T is surrounded by context words, their vectors are pushed together slightly, depending on the learning rate. The model is very simple and efficient way to capture contextual meaning of words and it can be applied in a very huge text corpus for any language.

To address this Word2Vec model with hierarchical softmax regulator was proposed for the first time. Later on, they proposed an optional method which is negative sampling. The optional negative sampling method has been shown to be more effective and simpler for small size data. The basic premise is that each time few random words are sampled to minimize their distance between vectors [50].

The word2vec algorithm uses a feed-forward neural network that used (is an artificial neural network where in connections between the nodes do not form a cycle) to predict the vector representation of words within N-dimensional language model. Word2vec model has two types of models: those are Continuous Bag-Of-Words (CBOW) and skip-gram (SG). Word2vec semantic representations were generated with the open source Gensim Python library [51]. It has different parameters that can be set based on trainers need. For instance, window size, the number of negative samples, model type, and the representation number of dimensionality are parameters of the model and used to maximize the performance. Most of the parameter usually was set to default gensim values except those listed above. Using the gensim similarity model, the semantic similarity of words is measured. The given vector representation between two words is calculated as follow cos (v1, v2) [55]. The semantic distances between two words d (v1, v2) was calculated as 1 minus the semantic similarity which is also known as cosine distance.

$$D(v1, v2) = 1 - \text{Cosine Similarity}(v1, v2) \quad [2.1]$$

A word vector represents every word with one at the index of the word and all other zeros, an example of vector representation is show below.

$$W^1 = \begin{pmatrix} 1 \\ 0 \\ \cdot \\ \cdot \\ 0 \end{pmatrix} \qquad W^2 = \begin{pmatrix} 0 \\ 1 \\ \cdot \\ \cdot \\ 0 \end{pmatrix} \qquad W^n = \begin{pmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ 1 \end{pmatrix}$$

Here, all ones represent the words at index 1, 2… up to n. Then, using word co-occurrence counts find the embedding. The co-occurrence of words can be counted via the window-based matrix. The method counts the number of times each word exists inside in the defined windows size [52].

Continuous Bag of Word Model

Continuous bag-of-words approach uses log-linear classifier to classify the predicted middle word given the surrounding future and history words. The architecture of CBOW can be shown in figure 2.3. CBOW maximizes equation (2.1) [53] where w(t) represents the current word while the context words are represented using the following symbols{ w(t-c) , … ,w(t-2) , w(t-1) , w(t+1) , w(t+2) , … , w(t+c) }.

$$\frac{1}{|V|}\sum_{t=1}^{|V|}\log\left[p\left(wt\middle|\begin{matrix}wt-c,\dots,wt-2,wt-1,\\ wt+1,wt+2,\dots,wt+c\end{matrix}\right)\right] \quad \text{..... [2.2]}$$

The size of the sliding window determines the number of the words in the context, such that if the size of the sliding window is five then the number of the context words is four and the value of c will be equal to four. Moreover, in order to predict a word, the preceding two words and the following two words, of the middle word to be predicted, must be considered in the context.

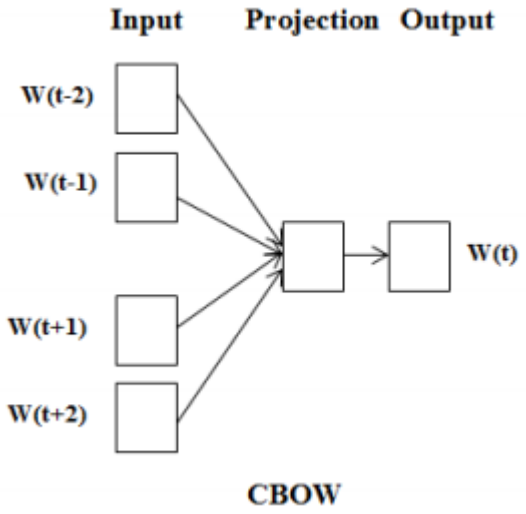

Figure 2-3 CBOW model architecture

Continuous Skip-gram Approach (Skip-gram)

CBOW and Skip-gram approaches of word2vec model are very similar in the structure. However, there is only one difference between the two approaches. While in CBOW the input for neural network are the context words and the output is the middle word, in Skip-gram model the input is the current word (middle word) and the output are the context words. For example, if the window size is five then, in case of CBOW the input will be two history words and two future words while the output will be the middle word. In case of Skip-gram the opposite is true, the input will be the middle word and the output will contain two future words and two history words. Furthermore, if the number of context words or the window size increased, the quality of the model will increase. On the other hand, the computation complexity will increase. The architecture of the Skip-gram model can be shown in Fig.2.4. The context words are represented using the following symbols { w(t-c) , … ,w(t-2) , w(t-1) , w(t+1) , w(t+2) , … , w(t+c) }, while w(t) represents the current word. As in CBOW the objective of the model is to maximize the log of the probability in equation (2) [53]. In addition, the sliding window is used to predict the next word. Finally, in both models the input for the neural network will be the one hot representation of the words.

$$\frac{1}{|V|}\sum_{t=1}^{|V|}\sum_{j=t-c,j<>t}^{t+c}\log[p(wj|wt)] \quad \text{.. [2.3]}$$

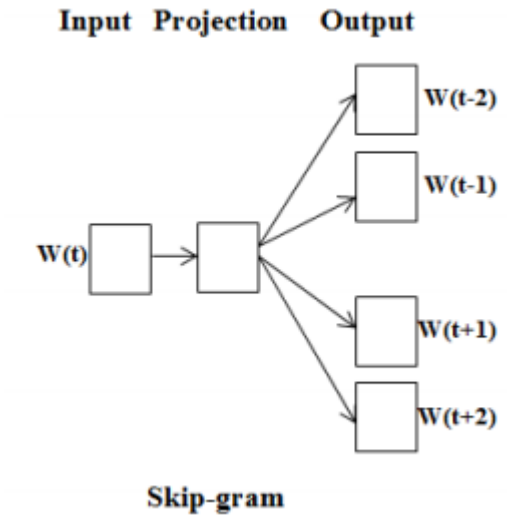

Figure 2-4 Skip-gram model architecture

As a conclusion, each of the previous word embedding's models has advantages and disadvantages. In the case of training small dataset, the Skip -gram model is efficient and the same is true in case of infrequent words, while CBOW is efficient and faster with frequent words [54].

CHAPTER 3: RELATED WORK

3.1 A HIDDEN WEB CRAWLER FOR MEDICAL DOMAIN

The paper [45] provides an approach to guide the crawler in choosing the right query term to be submitted to the search form interface by restricting itself to the processing of keyword based search forms that serve as interfaces to various textual databases in the 'Medical' domain. The system used manually generated top-down classification hierarchy in the same domain. The main components of HiCrawl crawler were a Downloader, a Web Page Analyzer, a Form Analyzer, and the Form Processor that uses a Domain Specific data repository and a Domain Specific Classification Hierarchy. The success of HiCrawl crawler, depends on the accuracy of the proposed approaches for selecting the 'wise' term to be fired on the search box of the candidate search form, using the manually constructed classification hierarchy that considers the 'System' level of living beings having 26 terms, from the hierarchy for evaluating the performance of both the approaches- breadth wise and depth wise.

3.2 GOOGLE'S DEEP-WEB CRAWL

This paper [25], describes a system for surfacing Deep-Web content, i.e., pre-computing submissions for each HTML form and adding the resulting HTML pages into a search engine index. They consider generic text boxes and start by applying the informativeness test on the template for the text box assuming a seed set of input values, i.e., a template informative if the generated pages are sufficiently distinct and uninformative otherwise. They select the seeds from the words on the form page. They adopt an iterative probing approach to identify the candidate keywords for a text box. At a high level, they assign an initial seed set of words as values for the text box and construct a query template with the text box as the single binding input. They generate the corresponding form submissions and extract additional keywords from the resulting documents. The extracted keywords are then used to update the candidate values for the text box. They repeat the process until either they are unable to extract further keywords or have reached an alternate stopping condition. On termination, a subset of the candidate keywords is chosen as the set of values for the text box. The techniques they employ to adapt iterative probing approach were select words from a page by identifying the words most relevant to its contents. For this

they use the popular Information Retrieval measure TF-IDF. The term frequency (TF) measures the importance of the word on that particular web page. The inverse document frequency (IDF) measures the importance of the word among all possible web pages.

3.3. DEEPBOT: A FOCUSED CRAWLER FOR ACCESSING HIDDEN WEB CONTENT

The authors [28], propose DeepBot, which is hidden-web focused crawler based on navigation sequence(NSEQL), and which is able to automatically fill in forms using an advanced form filling model supplied by the user. DeepBot receives a set of domain definitions as an input, each one describing a specific data-collecting task and automatically identifies and learns to execute queries on the forms relevant to them. DeepBot's crawling processes were based on automated "mini web browsers", built by using browser APIs (their implementation was based on Microsoft Internet Explorer). This enables the system to deal with client-side scripting code, session mechanisms, and other complexities related with the client-side Hidden Web. For every domain, the system tries to match its attributes with the fields of the form, using visual distance and text similarity heuristics

3.4 SIPHONING HIDDEN-WEB DATA THROUGH KEYWORD-BASED INTERFACES

In this paper [26], they study the problem of automating the retrieval of data hidden behind simple search interfaces that accept keyword-based queries. Their goal was to automatically retrieve all available results (or, as many as possible).In order to automatically retrieve data (results) through a keyword-based interface, it is necessary to determine which queries to issue and which keywords to use. They proposed a sampling-based approach to discover words that result in high coverage. Their intuition was that words in the actual database or document collection are more likely to result in higher coverage than randomly selected words. Their approach consists of two phases: sample the collection to select a set of high-frequency keywords and use these high-frequency keywords to construct a query that has high coverage. They select a word from the initial page, where the form is located then the Sample Keywords algorithm proceeds to find additional keywords by iteratively sub-mitting queries using keywords obtained in previous iterations. Since their goal is to retrieve as many results as possible, a simple strategy

is to also issue queries using stop words (e.g., a, the), since their frequency in the collection is likely to be high.

3.5. A NEW HIDDEN WEB CRAWLING APPROACH

The author [46], propose a domain specific Hidden Web crawler HiWC, having a Web Page Analyzer, a Form structure and content classifier, the Form filler that uses a Domain Specific data repository and a response page analyzer. Their crawling approach is divided in two phases: Domain Definition phase and Crawling phase. A domain definition is composed of Label table and Repository table. Labels table is an object relational table; it has three attributes label, alias and score. label represents a field that may appear in the search forms that are relevant to the domain. The alias represents alternative labels that may identify the attribute in a query form. The score is a number between 0 and 1 which represent the weight of each label in the domain, they collect search labels from training web sites, and then they find manually the alias of each label and give its score from its frequency in the search form in all the training sites. Repository table has m attributes, each one take a value of the attribute label from the labels table.

3.6. CRAWLING THE HIDDEN WEB: AN APPROACH TO DYNAMIC WEB INDEXING

They [27], have presented a framework for dynamic web indexing. In order to index the dynamic contents hidden behind the search forms, their approach follows six steps, which are Web pages collection, Form interface recognition, Keyword selection, Query submission, Result Processing and Updating the index. Web pages collection is essentially a static crawling, given the initial URL; the crawler recursively fetches all pages that are linked by it. Form interface recognition focus on the forms with one input control binding to a generic text box. They ignore forms with multiple input controls. They also restrict forms which are using get method to submit the form as they tend to produce contents suitable for indexing.

3.7. CRAWLING THE HIDDEN WEB

Hidden Web Exposer (HiWE) by Raghavan et al.[22], proposed architecture of the deep web crawler in the paper Crawling the hidden web (2001) . In this paper, they gave a prototype for a hidden web crawler named HiWE. This is based on the concept of extraction of task specific information. The main advantage of this strategy is that it minimizes the extraction of relevant information. The limitations of HiWE include its inability to support partially filled forms and to identify dependence between certain elements of form.

3.8. DOWNLOADING TEXTUAL HIDDEN WEB CONTENTS TROUGH KEYWORD QUERIRES

Ntoulas et al. [56], proposed a hidden web crawler which download textual hidden web contents behind keyword queries. They uses three different query generation policies, a policy that picks queries at random from a list of keywords, a policy that picks queries based on their frequency in a generic text collection, and a policy which adaptively picks a good query based on the content of the pages downloaded from the Hidden Web site. The choice of initial keyword is set manually; it is not done by the selection of the adaptive algorithm itself.

3.9. DESIGN AND IMPLMENTATION OF SEMANTIC HIDDEN WEB CRAWLER

Bhatia et al.[57], proposed a Domain–Specific Hidden Web Crawler (DSHWC) that automated the process of downloading of search interfaces, finding the semantic mappings, merging them and filling the Unified Search Interface (USI) produced thereof has been designed that finally submits the form to obtain the response pages from Hidden Web. The DSHWC use task specific database and label matching function for semantic label identifying.

3.10. DESIGN OF A DARK WEB CRAWLER AND OFFLINE LANGUAGE IDENTIFIER FOR AMHARIC DOCUMENTS

Daniel adenew [34], proposed Dark Web crawler consist of crawlers that work under a parallel and recursive algorithm called Fork-Join, a language identifier for identification of web documents and download manager components which target to be used in TOR network as Dark Web crawler.

Research Title	Database type	Technique	Strength	Drawback	Authors
HiCrawl: A Hidden Web Crawler for Medical Domain	unstructured	1. Creates a domain representation that is stored in domain specific data repository. 2. Uses a domain specific classification hierarchy for query term identification	1. Makes use of domain specific data repositories and thus can be extended to other domains 2. Can be fully automated if integrated with semantic web technologies	Requires human effort for an initial start of the crawler	Sonali Gupta et.al[45]
Google's Deep-Web Crawl	unstructured	TF-IDF Information Retrieval measure to select words from page by identifying the words most relevant to its contents. iterative probing approach to identify the candidate keywords for a text box	Make informative test that use to evaluate query templates, i.e., combinations of form inputs	Do not consider Semantic relatedness between words.	Jayant Madhavan,et.al [25]
Siphoning Hidden-Web Data through Keyword-Based Interfaces	unstructured	sampling-based approach to discover words that result in high coverage, select a set of high-frequency keywords	Use a simple and completely automated strategy that can be effective in practice, leading to very high coverages	1.their goal is to retrieve as many results as possible, they also issue queries using stop-words (	Luciano Barbosa et.al [26]

				e.g. a, the) 2.do not consider semantic relatedness between words	
Crawling the Hidden Web	Multi attribute or structured	1. Text similarity to match fields and domain attributes. Partial page layout and visual adjacency for identifying form elements	Significant contribution to label matching process	1.ignores forms with fewer than 3 attributes 2. Require significant human input thus performance highly depends on the quality of input data	Raghavan et.al[22]
Crawling the Hidden Web: An Approach to Dynamic Web Indexing	unstructured	Selection of keyword based on term frequency	Higher coverage.	Do not consider Semantic relatedness between words.	Moumie Soulemane et.al[27]
DeepBot: A focused crawler for accessing hidden web Content	multi-attribute	1.receives a set of domain definitions as an input 2.It detects forms relevant to the defined tasks and Executes a set of predefined queries on them.	Support JavaScript form	It require human effort to get domain definitions	Manuel Álvarez et.al[28]

Downloading Textual Hidden Web Content Through Keyword Queries	unstructured	1.Three different query generation policies, a policy that picks queries at random from a list of keywords, a policy that picks queries based on their frequency in a generic text collection, and a policy which adaptively picks a good query based on the content of the pages downloaded from the Hidden Web site.	Combination of policies (random, generic and adaptive) for choosing appropriate queries.	The choice of initial keyword is set manually ,it is not done by the selection of the adaptive algorithm it self	Alexandros Ntoulas et.al[56]
Design and implementation of domain based semantic hidden Web crawler	Single attribute and multiple attribute	1.Task-specific Database 2.Semantic form label identifier with task specific database	Use semantic form label identifier	Do not consider generic text field which accepts keywords	Manvi et.al[57]

Table 3-1 related work summary

3.8 Summary

As shown in Table 3.1 many hidden web crawler with unstructured database require human effort for the initial of the crawler and they only consider term frequency while selecting inputs. They do not consider semantic relatedness between keywords .which helps to download relevant pages. And those hidden web crawlers with structure database or multi attribute receive task specific database or domain definition as input which means they are not fully automated. Generally existing work don't use automatic keyword extraction which consider semantic relatedness between words.

CHAPTER 4: DESIGN OF HIDDEN WEB CRAWLER USING WORD2VEC MODEL

4.1. Overview

In this Chapter, we present our hidden web crawler developed using word2vec model which is used for keyword extraction from the web page. Different components of the proposed hidden web crawler architecture are described along with relevant techniques.

Hidden web is a group of Web data which can be accessed by the crawler only through an interaction with the Web-based search form. The only "entry point" to a Hidden Web site is through query interface. Thus, the content of such web can't be extracted using the regular web crawler. The main challenge of a Hidden Web crawler is how to automatically generate meaningful queries to issues to the site. In this work, generating meaning full query starts from the page itself with word2vec model to capture contextually similarity between words. Then extracted keyword will be automatically submitted to download the response page.

4.2. Hidden web crawling challenges

The following are the list of hidden web crawler challenges that considered while designing and developing the proposed hidden web crawler.

a) **Determining the entry points of Hidden Web:**
   a crawler is required to automatically determine the entry point of hidden web. The Hidden Web pages could be accessed by submitting query terms through the query interfaces or search forms e.g. the query interfaces of book publishers/ sellers contain fields like Title, Author and ISBN etc. Thus, an effective Hidden Web crawler must identify the search forms to generate query terms to download the Hidden Web pages.

b) **Generating/Selecting the Query Terms:**
   The contents of the Hidden Web sources are stored in searchable databases that produce results dynamically in response to a direct request only. Thus, a method must be devised that generate/select query terms to extract hidden Web pages.

c) Automatically Filling the FORM's:

Another challenging task in the Hidden Web crawling is to automatically fill the FORM's. Whenever a keyword for a query submission is selected it will automatically be submitted in the search form to generate more search results.

4.3. System Architecture

As shown in Figure 4-1, the proposed system architecture of a hidden web crawler is composed of four modules: Form detector, Pre-processor, Keyword extractor, and Automatic form submission.

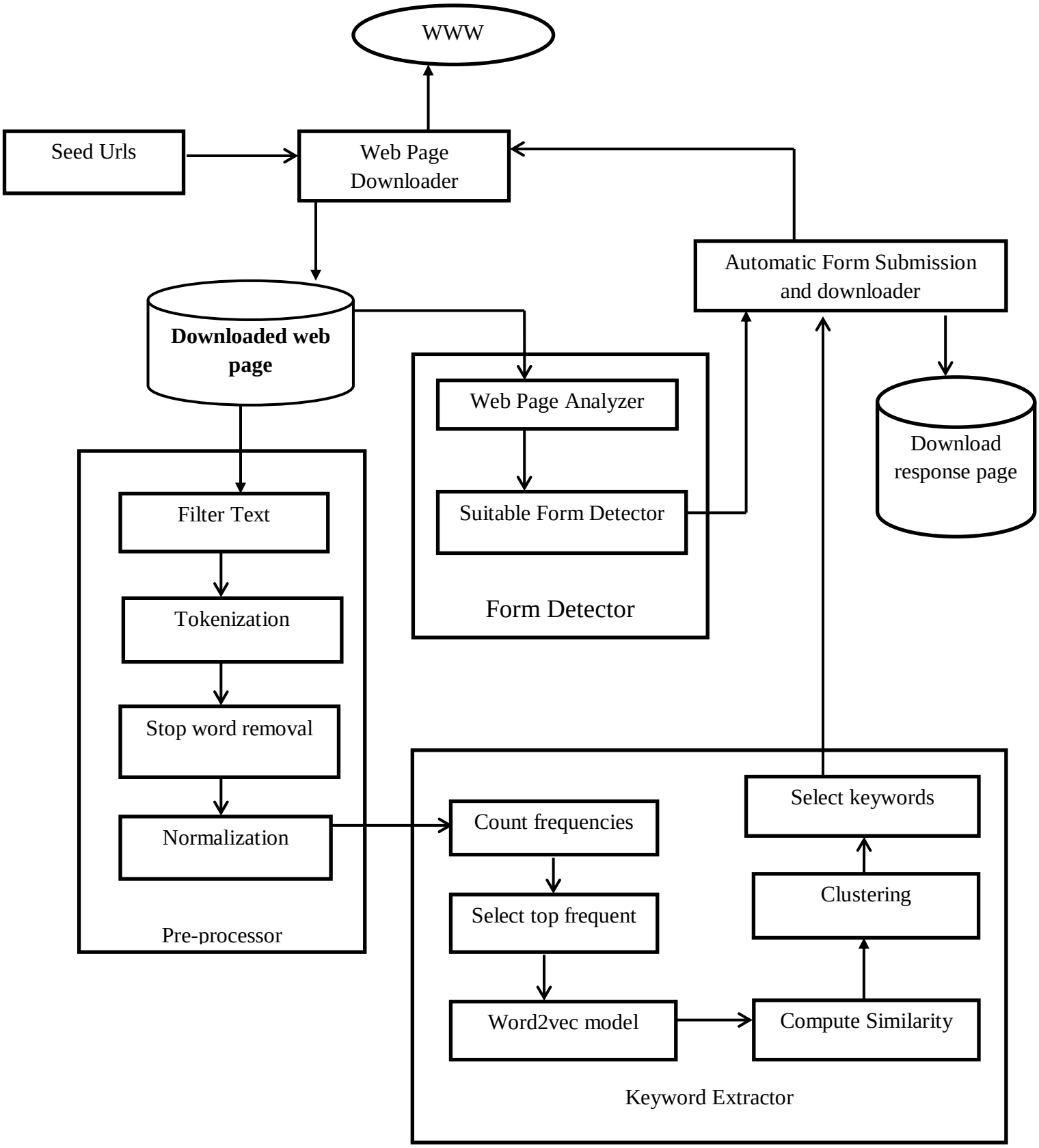

Figure 4-1 Architecture of hidden web crawler

Initially, the crawler takes seed URL as input and the web page downloader download result page of seed url and store it in the downloaded web page repository, then the form detector module receive web page from the downloaded web page repository and analyze the web page if the web page has some form filed or not and suitable form detector parse for various types of form found on the web page to get a suitable form where the query can be made. The pre-processing module receive downloaded web page from downloaded web page repository and it filter text from all the other content of a web page. The filtered text then will be normalized and converted to a token. Then the token passed to the keyword extraction module, after that, it counts the frequency of each word in the text to select top-frequent words as candidate keywords. The semantically similar words among top-frequent words will be clustered together using a word2vec model. The keyword will be extracted from clusters. The extracted keywords were used as input for the automatic form submission module. The automatic form submission module receives the suitable form and submits keywords iteratively to get response pages.

4.3.1 Form detector

The form detector module is responsible to download web page and analyze every Web page the crawler comes across. It accepts and scans the Web page to see if the Web page can be used as a search page to retrieve information or not. The result of the form detector module will be used for pre-processor and automatic form submission.

Web Page Analyzer

The World Wide Web has millions of web pages which can be accessed through the internet that includes images, videos, pages, and other online information**,** some of them will have forms. These forms will act as an entry point for the vast information is hidden behind them. The analyzer will be analyzing every Web page the crawler comes across. It will scan the Web page to see if the Web page has a web form or not.

Suitable Form Detector

Suitable form detector is responsible to look for HTML forms on a Web page and see if those forms are actually the search forms and not registration forms. Based upon the survey has been made on large number of e-commerce website [48], it has been observed that these forms which are designed for search function has specific type of format. Mostly, the submit buttons of these forms are found to be named as “GO” or “Search”.

4.3.2 Pre-processing

The Pre-processing module is responsible to filter text from all the other contents of the web page. The first step in the text data analytics process is to create vector for the text documents to be analyzed a word, because computers are unable to understand the concepts of words. It requires data to be converted into a numeric format to perform any machine learning task. But not every word in the text document is important. For instance, text data often contains some special formats like number, punctuation mark, and stop words. Those do not contribute any meaning that would benefit the keyword extraction process. For this reason, text data must be preprocessed before usage. Preprocessing is about data cleaning or the task that is used to make the text data usable for analysis. The most common pre-processing activities in text data clustering are tokenization, normalization, and stop-word removal.

```
Input:
    Wp=web page

Output:
    Ft=filtered text

Begin : Ft=[ ]
    Soup=beautifulSoup(wp,'html.parser')
    For page in Soup.find_all:
        Ft = Ft +Wp.text
    return Ft
    End
```

Algorithm 4.1 Filter text from web page

a. Tokenization

Tokenization is the process of splitting the given text into units named as tokens or individual words. This is done by locating word boundaries (the ending point of a word and the beginning of the next word). The tokens may be words, numbers, punctuation marks, special symbols, etc. in this work we use nltk.word_tokenize to tokenize each text.

```
Input:
    Ft=filteredText
Output:
    Lt=list of token
Begin:
        For words in filteredText:
            tokens=nltk.word_tokenize(str(filteredText))
        return tokens

        End
```

Algorithm 4.2 Tokenization algorithm

b. Stop-word Removal

Keyword extraction is mainly based on the frequency of words. The words that occur frequently are more nominated to be keywords. However, even that stop words occur frequently they cannot be considered as keywords, thus stop-words are removed in preprocessing phase. We remove English stop words using lists of stop words from the nltk library.

```
Input:
    Lt=list of token
Output:
    Ts=Token with out stop word
Begin:
    for token in Lt:
```

```
        Ts =[ ]
        stop_words={a , and ,am ,any, such ,there …….etc}
        if  not  token  in stop_words
                Ts = Ts +
        token return Ts
    End
```

Algorithm 4.3 stop-word removal algorithm

c. Normalization

The normalization is the process of replacing the inflected forms of a word with the root word. The inflected form represents the different usage of a word in the sentences. For example, finds, finding, and found are the inflected forms of the word find. In natural language processing, the lemmatization is used to find the inflected form of the words with different spellings, such as finds and found for the word find in the above example. Unlike lemmatization, the stemming process takes care of the prefixes and suffixes to find the root word, such as finding in the above mentioned example. But before lowercasing and lemmatization we remove token which are number, non-English character and special symbols the output of the normalization process is the tokens with all the inflected forms replaced with their root word.

```
Input:
    Ts= Token with out stop words
Output:
    Nt= Normalized tokens
Begin:
      for token in Ts:
          if not token.isdigit()
            if not token special char
              if not token nonEnglishChar
```

```
                    lower_token=[]
                    lower_token=lower_token + Ts.lower()
                    return lower_token
                end
            end
        end
    end

for token in lower_token :
        lemmatizer= wordnetlemmatizer()
        lemmatized_token=[ ]
        lemmatized_token= lemmatized_token +  lemmatizer.lematize(token)
        return lemmatized_token
end
```

Algorithm 4.4 normalization algorithm

After the pre-processing module we get lists of token, so that we have to choose relevant keywords from the given token. Keyword extraction module automatically selects a set of words that are relevant to the web page. So in this work keywords are selected from the list of token by identifying the words most relevant to page contents. Referring to Figure 4-4, the first task in the keyword extraction is count the frequency of each word. The count frequency function calculates the frequency of the word in the list by using Algorithm 4-1. Then the top-frequent words are selected as candidate keywords. It then clusters top frequent words based on their semantic similarity using word2vec model. The word2vec word embedding model is used to represent words using vectors. So that, cosine similarity able to compute easily the semantic similarity between vector representations of each word. Therefore, the words that have high context semantic similarity will be grouped in the same cluster. We use an agglomerative clustering to find similar words in the lists. The goal is to group semantically related words together, so that

each cluster represents a frequency of one concept. Semantic measure overcomes the problem of creating several clusters based on the distribution of the word over the page will ensure good coverage to the web page. We use the frequency of the words in the web page as a criterion for selecting keywords from each cluster.

```
Input :
      T:Tokens
Output:
       LTF: list of tokens with their frequency
Begin:
     Freq=[ ]
     For word in T:
            Freq[word]=T.count(word)
            tf=Zib(tokens,freq)

     Return (list(tf)))
     End
```

Algorithm 4.5 frequency count algorithm

4.3.3. Word2vec Embedding

Word2vec is unsupervised machine learning algorithm that does not need any human annotation to learn. The output of w2v (the generated word vectors) can be used in clustering algorithms to create clusters of semantically related words. It is responsible to build word embedding model that embrace the vectors representation of words in the given input e-commerce product review text corpus. This component has the following subcomponents:

Corpus preprocessing

We collect and build a corpus from a set of publicly available e-ecommerce product review text collections.

Removal of Extraneous Characters

The numbers, punctuation marks and control characters in the text of each file were not considered for building a distributional model as they do not provide important information about target word meaning. Words containing numbers, punctuation marks, digits, all non- English character, special symbols ($, %, &, |, _, -?) and stop words were excluded at the first phases of preprocessing.

Normalization

The normalization process is also carried out in the preprocessing phase. The normalization process includes lowercasing all text data and transforming a text into basic forms using lemmatization. After cleaning and normalization, the product review text corpus is transformed into normalized tokens and a vocabulary set of size n contains all words extracted from the dataset is generated. After the entire dataset is preprocessed, it is utilized to construct the neural word embedding model.

Word2vec model

Train word2vec model

The Word2vec algorithm feeds the pre-processed training text corpus as an input and produces the word vectors (embedding) as output. The algorithm first creates a vocabulary from the training text data and then learns vector representations of the words. It is a mapping of words into vectors of floating numbers using the neural network. The vector representation that is formed using the word2vec algorithm is considered as a feature. Those are represented in semantic vector space, in the form of floating numbers between 0-1. Each unique word in the sample corpus being assigned a corresponding vector in the space called dimensions. A vector space includes hundreds of dimensions. In this space, words that share similar contexts in the corpus are placed close to one another. This is based on the cosine similarity of words. The vector representation result is a model. This model is saved as .bin or .txt or .csv or .mod etc. and is used as a feature to our task. To generate the word vectors well, we employ the Continuous bag of words (CBOW) model because it is faster and works well with frequent words. The CBOW Model objective is to find the word representations that are useful for predicting the

middle word of a given context (or surrounding words) in a given pre-processed e-commerce product text.

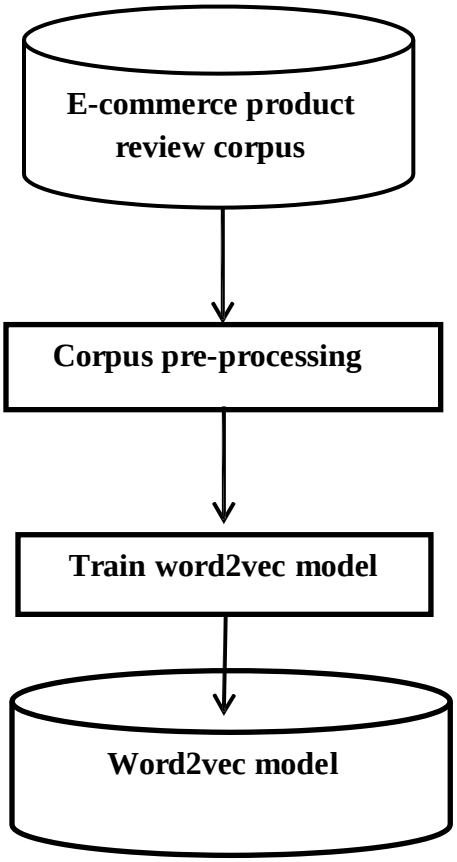

Figure 4-2 Architecture of word2vec model

4.4 Clustering

There are different clustering methods such as partition, hierarchical, grid-based and spectral; however, in this work, we use hierarchical clustering, because the number of clusters can be controlled by simple thresholding. A hierarchical agglomerative clustering start with each data item in its own cluster and then organizes the data as a set of nested clusters within a hierarchical tree, also known as a dendogram. Until all data items belong to the same cluster, are merged iteratively in a pairwise manner that have the smallest distance. Algorithm 4.6 shows hierarchical agglomerative algorithm.

Input :

a set X of objects($x_1 \ldots\ldots x_n$)

a distance function dist(c_1,c_2)

Output:

Clusters

Begin:

For i=1 to n

$C_i=\{x_i\}$

End for

$C=\{c_1,\ldots,c_n\}$

L=n+1

While C.size>1 do

(c_{min1},c_{min2})=minimum dist(c_i,c_j) for all c_i,c_j in c

Remove c_{min1} and c_{min2} from C

Add{c_{min1},c_{min2}} to c

L=l+1

End while

Algorithm 4.6: Hierarchical agglomerative clustering algorithm

So that, in this research the words are clustered together using agglomerative algorithm. And we used the most popular similarity measure, i.e., cosine similarity, which measures the angle between the word vectors.

4.4.1 Select keywords

We use the frequency of the words in the web page as a standard for selecting keywords from each cluster. We rank the words in each cluster based on their frequency in the page and we select the top frequent words. On the condition of tie, the average similarity to all other words in the same cluster is considered as a decision criterion. This favors words that are more central in the cluster.

4.4.2 Automatic form filler and submitter

After getting a searchable form and list of selected keyword, there is another challenge in submitting the query in the search form automatically, i.e. without any human interaction. Whenever a keyword for a query submission will be selected it would automatically be submitted in the search form to download the response page.

```
Input:
        F=form  and  LK= list of keyword
Output:
         response page
Begin:
For keyword in LK:
              for form in F:
                   while LK=null do
                          insert keyword to text input
                         submit form with the keyword
                          Download response page
                   End while
              end
    End
```

Algorithm 4-7: Automatic Form Submission

4.5 Summary

A hidden web crawler is a process of downloading hidden web pages behind web form. The proposed technique makes use of word2vec model to capture the semantics of words and their context, and clustering algorithms, to identify the essence of the terms and choose the more significant one(s), to fill the HTML search form. The hidden web crawler starts with downloading a web page from the web using a given seed URL in the web page downloader. Then the form detector module analyze downloaded web page to see if the web page can be used as a search page to retrieve information or not. It basically sees if the web page has some form of field or not. Once the form found the suitable form detector looks for various types of form found on that web page, to see if those forms are actually the search form. This suitable form detector picks form which is suitable to search. The result of the form detector module is suitable form, which is used as input for automatic form filler. The preprocessing module starts with text filtering from web page. Then the filtered text will be normalized and tokenized to remove insignificant word to keyword extraction module. The Keyword extraction module is responsible to count the frequency of each token and select the top frequent one. Then it convert top frequent words into vectors using word2vec model to captures the semantic similarity between words using cosine similarity matrices. After that the words that have high semantic similarity will be clustered together. Then we select frequent words in each cluster. The Automatic form filler and submitter module receive the searchable form detector module and list of keyword from the keyword extraction module. Then it iteratively fills keywords and submits a form to download the response page.

CHAPTER 5: IMPLEMENTATION AND EXPERIMENTAL RESULTS

5.1 Overview

A Semantic Hidden web crawler is developed according to the functionalities defined in the previous Chapter. In this Chapter, we have presented different tools and development environments used to develop the prototype of the system. In addition, screen shots are presented to demonstrate the prototype. Finally, we have evaluated accuracy of the keyword extraction from web page using keywords which were manually extracted from ecommerce web page by ecommerce user.

5.2 Dataset preparation

Three different sets of data were required to undertake the experiments. The first set of data is a collection of e-commerce product review text. This data were collected from Kaggle [14] and amazon product data [15]. After collecting the dataset we performed the pre-processing task. The pre-processing task remove extraneous character such as numbers, punctuation marks and control characters in the data text of each file were not considered for building a distributional model as they do not provide important information about target word meaning .We also remove the stop words, clean up the special symbols, and punctuation mark. After that, text will be tokenized into their individual form and normalize tokens in to their basic forms using lemmatization. The normalized tokens used to train word2vec model with genism library.

The second set of data is a collection of e-commerce hidden web pages with generic text box. Since our word2vec model is trained with e-commerce product review text. We only consider e-commerce hidden web pages like alibaba.com, Walmart.com, sheger.net and qefira.com.

The third set of data is a collection of manually extracted keywords from hidden web pages to evaluate the performance of automatically extracted keyword from the page itself.

5.3. Word2vec model

After preprocessing tasks, the unlabeled free-text document is transformed into numeric values (dense vector representation of words that capture something about their meaning). CBOW is fast and also better for capturing frequent words. The implementation is in the Gensim Python library. There are different word2vec parameters that we specify in our training. We reconfigured some of them and use the rest default parameters. The parameters that are reconfigured are listed below.

- **Vector Size:** It is the number of dimensions of the embedding, which is the length of the dense vector to represent each word. We implemented with 300 dimensions.
- **Window**: The maximum distance between a target word and context word. For our implementation, we set 4 as the window size.
- **Min_count**: It is the minimum count of words to include the word in training vocabulary. Words with an occurrence of fewer than the frequency specified will be ignored. In our implementation, we set it 4.
- **Training architecture**: The training algorithm, either CBOW (0) or skip-gram (1). We use CBOW (0).

Finally, the training result is returned in the file type that we want. It may be text file, or .model or .bin. This file consists of the number of unique vocabulary and dimension number. The length of vocabulary in our trained word2vec model is 943639. In addition, there is the vocabulary with its context in real number representation. For instance, a vocabulary “garden” represented as follow.

```
: similar_words = modelFinal.most_similar('garden')
  print(similar_words)

  C:\Users\engda\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: DeprecationWarning: Call to deprecated `most_similar` (Meth
  od will be removed in 4.0.0, use self.wv.most_similar() instead).
    """Entry point for launching an IPython kernel.

  [('gardening', 0.6538162231445312), ('botanic', 0.642959713935852), ('houseplant', 0.6196558475494385), ('flowerbed', 0.6173162
  460327148), ('planting', 0.6135658621788025), ('planter', 0.5974619388580322), ('backyard', 0.595092236995697), ('hydroponic',
  0.5935441851615906), ('covent', 0.5879462957382202), ('landscaping', 0.5850338935852051)]
```

Figure 5-1 Example of contextually similar words

5.4 Implementation

The prototype is implemented using python programming language; Python is an object-oriented programming language that provides rapid application development. It is used for general purpose and high-level programming. Anaconda, it the most popular python distribution tool, Anaconda simplifies package management and deployment. From Anaconda supported programming languages and IDEs, we used python. We use Conda to install package. Jupyter notebook which allows creating and sharing documents from code to full blown reports. Beautiful soup html parser, it is a python library to parse html.

We performed our training using Gensim (a library that installed in anaconda) open-source library which provides the word2vec class for working with a word2vec model. Natural language Toolkit (NLTK) is the other module that we use in the experiment.

5.5 System Prototype

To demonstrate the validity of the proposed system, we develop a prototype system. The prototype developed show the user interface and output of the system.

The prototype is initialized by using a seed URL as it is shown on the graphical user interface shown in Figure 5.1

System User Interface

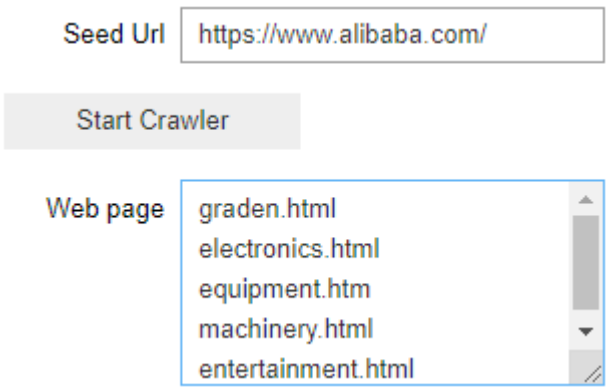

Figure 5-2 Hidden web crawler interface prototype

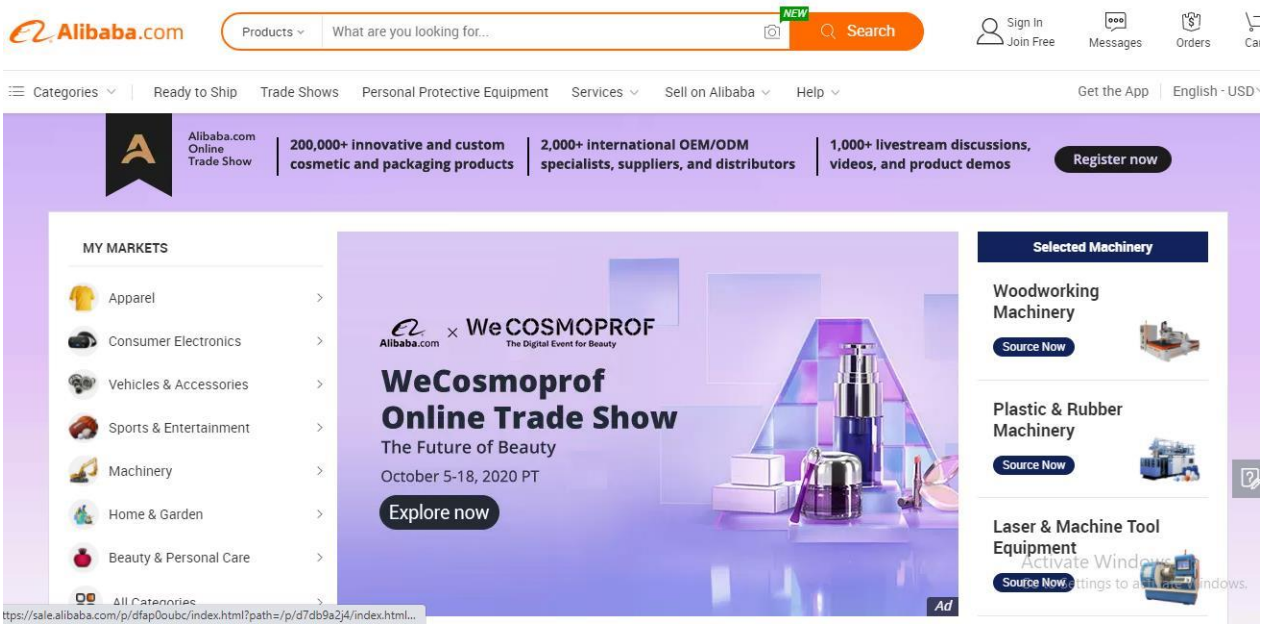

Figure 5-3 Search interface of Alibaba e-commerce web page: www.alibaba.com

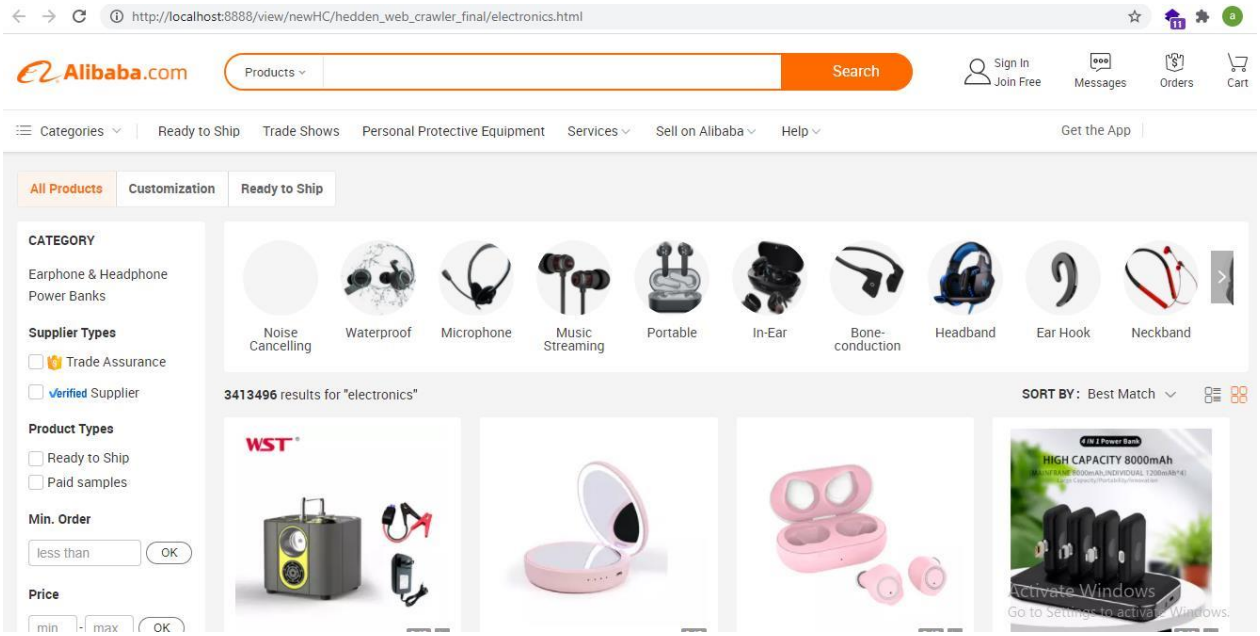

Figure 5-4 Result page of web crawler after submitting electronics keyword electronics html result

5.6 Evaluation

The objective of this research is to design a Hidden Web crawler which will be used to download relevant hidden web pages behind web form. However the relevancy of the downloaded web page depends on the extracted keyword to be submitted, so our evaluation is done on keyword extraction.

We use Alibaba e-commerce web page to see how our method performs in keyword extraction. For the performance analysis, we use precision, recall, and F-measure scores with different number of keyword to evaluate our methodology and decide the number of keyword extracted per page with 10 because our experiment shows that keyword extraction with word2vec good precision, recall and f- measure value.

We use TF to extract keywords from web page for comparison purpose with keyword extracted from web page using word2vec model. We use the same algorithm with the word2vec model for TF for extracting keywords.

	Number of extracted keywords								
	10			15			20		
Methods	P	R	F	P	R	F	P	R	F
W2V	0.8	0.8	0.8	0.73	0.73	0.73	0.6	0.6	0.6
TF	0.60	0.60	0.60	0.20	0.133	0.155	0.55	0.50	0.5166

Table 5-1 Evaluation of keyword extraction

Precision is the number proportion fraction of the correctly recognized keywords and measured as:

$$Precision=TP/TP+FP \quad [5.1]$$

A recall is the number proportion fraction of the keywords in the ground truth that are correctly recognized and can measure as:

$$Recall=TP/TP+FN \quad [5.2]$$

F-measure is a classical accuracy measure and is a harmonic mean of precision and recall. It is calculated using the formula:

F-measure < (2*Precision*Recall)……. [5.3]

Where TP is true positive, FP is false positive, and FN is false negative.

CHAPTER 6: CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

Hidden web has a very large volume of quality information compared to surface web. Extraction of hidden web information can be highly fruitful for a general user or a specific user. Traditional search engines deal with the Surface Web which is a set of Web pages directly accessible through hyperlinks and ignores a large part of the Web called hidden Web which is a great amount of valuable information of online database which is "hidden" behind the query forms.

There are works done so far in the area of hidden web crawling. However most of them require human effort to start crawling which means they are not fully automatic and they do not consider semantic relatedness between keywords extracted from web page which will be used as query to extract relevant hidden web page.

In this work, literature review and analysis of some of the important hidden web crawlers was carried out to find their advantages and limitations. A comparative analysis of existing hidden web crawler was carried out on the basis of database structure and query formulation. Hidden web crawler approach is proposed using word2vec model, to automatically extract relevant keywords from hidden web page itself, which will be used as a query for generic text box.

We have used different tool and development environment while developing the prototype. Evaluation of the system is carried out using different evaluation criteria, such as precision, recall and f-measure.

6.2 CONTRIBUTION OF THIS WORK

The system designed in this thesis contributes to the search engine to access hidden web as a Web crawler is main component of search engine. The main accomplishments of this work are: a Hidden Web Crawler Architecture Using word2vec Model, E-commerce word2vec model, E-commerce product review dataset and prototype of Hidden Web Crawler Using word2vec Model.

6.3 FUTURE WORK

This research work explores different areas that can be further improved as well as some components that should be implemented and integrated for better functioning of the system. Some of the future works include: a complete implementation, evaluation and analysis of this approach, compare the performance with different hidden web crawler, design to include hidden web behind multi attribute and design and implement hidden web search engine including indexing and ranking

Reference

1. Hasan K S, Ng V, “Automatic Key phrase Extraction”, A Survey of the State of the Art[C] Meeting of the Association for Computational Linguistics. 2014:1262-1273
2. A. Onan S. Korukoğlu and H. Bulut "Ensemble of keyword extraction methods and classifiers in text classification" Expert Systems with Applications vol. 57 pp. 232-247 2016
3. S. Rose D. Engel N. Cramer and W. Cowley "Automatic keyword extraction from individual documents" Text Mining: Applications and Theory pp. 1-20 2010.
4. K. Sparck Jones "A statistical interpretation of term specificity and its application in retrieval" Journal of Documentation vol. 28 pp. 11-21 1972.]
5. G. Salton A. Wong and C.-S. Yang "A vector space model for automatic indexing" Communications of the ACM vol. 18 pp. 613-620 1975.
6. Daudaravicius, V.: Automatic identification of lexical units. Informatica: An International Journal of Computing and Informatics 34(1), 85–91 (2010)
7. Jean-Louis, L., Gagnon, M., Charton, E.: A knowledge-base oriented approach for automatic keyword extraction. Computaci´on y Sistemas 17(2), 187–196 (2013)
8. Jiang, X., Hu, Y., Li, H., “A ranking approach to key phrase extraction”. In: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 756–757. ACM (2009)
9. Frank, E., Paynter, G.W., Witten, I.H., Gutwin, C., et al.”Domain-specific key phrase extraction”, In: Proc. Sixteenth International Joint Conference on Artificial Intelligence, pp. 668–673. Morgan Kaufmann Publishers (1999)
10. Turney, P.D.”Learning algorithms for keyphrase extraction”, Information Retrieval 2(4), 303–336 (2000)
11. A. Hulth "Improved automatic keyword extraction has given more linguistic knowledge" in Proceedings of the 2003 conference on Empirical methods in Natural Language Processing 2003 pp. 216-223
12. G. K. Palshikar "Keyword extraction from a single document using centrality measures" in International Conference on Pattern Recognition and Machine Intelligence 2007 pp. 503-510.
13. D. B. Bracewell F. Ren and S. Kuriowa "Multilingual single document keyword extraction for information retrieval" in Natural Language Processing and Knowledge Engineering

2005. IEEE NLP-KE 05.Proceedings of 2005 IEEE International Conference on 2005 pp. 517 522

14. Ecommerce product review dataset, https://www.kaggle.com/datasets**,** retrieved on April 1,2019
15. Product review dataset, https://jmcauley.ucsd.edu/data/amazon/**,** retrieved on April 1,2019
16. Dilp kumer sharma and A.K. Sharma, "A Novel Architecure of Deep web crawler" in international journal of information technology & web engineering USA, vol.6,No
17. Dilp kumer sharma and A.K. Sharma,"Search Engine: A backbone for information Extraction in ICT Scenario", In International Journal of ICTHD, USA, vol.3 , No. 2, pp.38-51,2011
18. Qingzhao Tan, "Designing New Crawling and Indexing Techniques for Web Search Engines", 2008.
19. Craswell and David Hawking ,Web Information Retrieval Author Preprint for Web Nick 18 April 2009
20. Web crawler, http://www.thinkpink.com/bp/WebCrawler/History.html, retrieved on October 12, 2019.
21. J. Hirai, S. Raghavan, H. Garcia-Molina, and A. Paepcke. WebBase : A repository of web pages. In Proc. of the 9th Int. World Wide Web Conference, May 2000.
22. S. Raghavan, H. Garcia-Molina. "Crawling the Hidden Web", In: the proceedings of the 27th International Conference on Very large databases VLDB'01, Morgan Kaufmann Publishers Inc., San Francisco, CA, p.p. 129-138,2001
23. Luciano Barbosa and Juliana Freire, "Searching for Hidden-Web Databases", Eighth International Workshop on the Web and Databases,2005. https://www.researchgate.net/publication/50520349
24. Dilp kumer sharma and A.K. Sharma," Deep Web Information Retrieval process: A Technical Survey", In International Journal of Information Technology & Web Engineering,USA, VOL5, No.1,pp.1-22,2010.
25. Jayant Madhavan, David Ko, Lucja Kot, "Google's Deep-Web Crawl", Auckland, New Zealand , August 23-28,2008
26. Luciano Barbosa, Juliana Freire , "Siphoning Hidden-Web Data through keyword-

Based Interfaces",Department of Computer Science & Engineering , School of Computing University of Utah,2004

27. Moumie Soulemane ,Mohammad Rafiuzzaman ,Hasan Mahmud * Crawling the Hidden Web: An Approach to Dynamic Web Indexing,International Journal of Computer Applications (0975 – 8887) Volume 55– No.1, October 2012 7
28. Manuel Álvarez, Juan Raposo, Alberto Pan, Fidel Cacheda, Fernando Bellas, Víctor Carneiro ,* DeepBot: A Focused Crawler for Accessing Hidden Web Content ,conference Paper ,https://www.researchgate.net/publication/221327368, January 2007
29. Qingzhao Tan, "Designing New Crawling and Indexing Techniques for Web Search Engines", 2008.
30. Web Information Retrieval Author Preprint for Web Nick Craswell and David Hawking 18 April 2009
31. M. Koster ,'The Web Robots Pages', 1994.
32. P. Bailey, et al. (2000). 'Dark Matter on the Web'. In WWW9 Poster Proceedings
33. E. Greengrass,"Informational Retrieval :A Survey",University of maryland, "Baltimore County, November 2000.
34. Daniel Adenew," Design of a Dark Web Crawler and Offline Language Identifier for Amharic Documents", Department of computer science, Addis Ababa University, February 2016.
35. John D.King ,'Search engine content analysis', Queensland University of Technology, Brisbane,Australia Submitted on December 15, 2008.
36. KwongBor Ng and Paul P. Kantor. An investigation of the preconditions for effective data fusion in information retrieval: A pilot study, 1998.
37. Sherman, C., & Price, G.: The Invisible Web. Medford, NJ: Information Today, 2003
38. JyriLehtonen ,'Characterizing the deep Web', Master's thesis UNIVERSITY OF TURKU Department of Information Technology Computer Science December 2011.
39. Baeza-Yates, Ricardo and Ribeiro-Neto, Bertheir. "Modern Information Retrieval". New York: ACM Press, 1999

40. A. Shen, "Algorithm and Programming Problems and Solution", Springer , 2nd ed. 2010 , pp.135.
41. Christopher D. Manning, PrabhakarRaghavan, andHinrichSchütze. “An Introduction to Information Retrieval”. Cambridge University Press,England Online edition (c), 2009.
42. Bergman, Michael K. The Deep Web: Surfacing hidden value. Bright Planet, 2001.
43. Najork, Marc. Web Crawler Architecture. Mountain View: Microsoft Research, 2005.
44. Chang, C.-C.K., He, B., Patel, M., Zhang, Z.: Structured Databases on the Web: Observations and Implications. SIGMOD Record, 33(3), (2004)
45. Sonali Gupta , Komal Kumar Bhatia ,* HiCrawl: A Hidden Web Crawler for Medical Domain Department of Computer Engineering YMCA University of Science & Technology Faridabad, India,2013
46. L.Saoudi ,Mohammed Boudiaf Abderrahmane Mira University bejaia, algeria S.Mhamedi* A New Hidden Web Crawling Approach, Vol. 6, No. 10, 2015
47. Desai, Lovekeshkumar, "A Distributed Approach to Crawl Domain Specific Hidden Web." Thesis, Georgia State University, 2007
48. Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." arXiv:1301.3781v3 [cs.CL] 7 Sep 2013
49. Jon Ezeiza Alvarez. A review of word embedding and document similarity algorithms applied to academic text. October 22, 2017.
50. Radim Rehuek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, pages 45–50, Valletta, Malta, 5 2010. ELRA
51. Socher, Richard, and Richard Socher Mundra. "CS 224D: Deep Learning for NLP1." , lecture note part 2, (2016).
52. A. Mahdaouy, E. Gaussier, and S. Ouatik El Alaoui, Arabic Text Classification Based on Word and Document Embeddings.International Conference on Advanced Intelligent Systems and Informatics, 2016.
53. M. Naili, A. H. Chaibi, and H. H. Ben Ghezala, “Comparative study of word embedding methods in topic segmentation,” Procedia Computer Science, vol. 112,

pp. 340–349, 2017.

54. Edgar Altszyler, Mariano Sigman, Sidarta Ribeiro and Diego Fernández Slezak. Comparative study of LSA vs Word2vec embeddings in small corpora: a case study in dreams database. 11 Apr 2017.
55. Dilip Kumar Sharma. “Design of a framework for extraction of deep web information”, Faculty of electronics, informatics & computer engineering shobhit university, meerut (india), 2012
56. Alexandros Ntoulas, Petros Zerfos and Junghoo Cho, “Downloading Textual Hidden Web Content through Keyword Queries”, UCLA Computer Science, Denver, Colorado, USA 2005
57. Manvi , Ashutosh Dixit , Komal Kumar Bahatia and Joyti Yadav, “Design and Implmentation of Domain based Semantic Hidden Web Crawler”, Department of Computer Engineering YMCA University of Science and Technology Faridabad, India, IJIACS, ISSN 2347-8616, Volume 4, may 2015.

Declaration I, the undersigned, declare that this research is my original work and has not been presented for degree in any other university, and that all sources of materials used for the research have been acknowledged.

Declared by:

Name: Engdawerk Kebede

Signature: _____

Date: _____

Confirmed by advisor:

Name: Fekade Getahun (Ph.D.)

Signature: _____

Date: _____

Place and date of submission: Addis Ababa University, October 2020.