Incorporating Arabic Corpus into Google N-gram Viewer: Initial Results

Conference Paper · November 2014

CITATION
1

READS
906

2 authors:

Mohammad I. Zarour
Prince Sultan University
64 PUBLICATIONS **608** CITATIONS
SEE PROFILE

Izzat Alsmadi
Texas A&M University San Antonio
370 PUBLICATIONS **2,774** CITATIONS
SEE PROFILE

Incorporating Arabic Corpus into Google N-gram Viewer: Initial Results

Mohammad Zarour
College of Computer Science and Information Technologies
Prince Sultan University, P.O.Box 66833
Rafha Street, Riyadh 11586, Saudi Arabia
mzarour@psu.edu.sa

Izzat Alsmadi
Computer Science Department
Boise State University,
1910 University Drive
Boise, ID 83725
izzatalsmadi@boisestate.edu

Abstract: Google N-gram Viewer is a graphing tool that charts time-based data to show the frequency of usage of words or phrases. The tool is based on a large dataset from books collected by Google from open sources. Although Arabic is the fifth spoken language in the world with speakers more than French, German, Russian and Italian languages, unfortunately, Arabic language is not included as one of the corpora indexed by the Google n-gram viewer. This paper illustrates the possibility of building a big Arabic corpus and indexing it to be included in Google N-gram viewer. A case study is presented to build a dataset to initiate the process of digitizing the Arabic content and prepare it to be incorporated in Google N-gram viewer. One of the major goals of including Arabic content in Google N-gram is to enrich Arabic public content which has been very limited in comparison with the size of Arabic speakers. We believe that adopting Arabic language by Google N-gram viewer can significantly benefit researchers in different fields related to Arabic language and its speakers' history.

Keywords: Arabic, Corpus, Natural Language Processing, Google N-gram Viewer.

I. INTRODUCTION

The size of data available on digital format is increasing enormously day after another. The dissemination and use of social networks, mobile devices and internet content play the main role in data explosion we see nowadays. big data is about to change everything, in research domains and industries such as linguistics, genomics, e-commerce, Internet search, automated translation and social networking.[1]. To be able to index and analyze big data, a method is needed to divide the text into chunks or pieces. One of the well-known corpora that includes such big data is Google Books N-gram Corpus [2]. This corpus has "enabled the quantitative analysis of linguistic and cultural trends as reflected in millions of books written over the past five centuries" [3]. one of the tools that used Google N-gram Corpus to draw online phrase-usage graphs is Google N-gram viewer [2].

Google N-gram viewer is one of those newly published Google services. Google N-gram Viewer is a graphing tool that charts time-based data to show the frequency of usage of words or phrases. Google archived digitally a large amount of books in different languages. Google populated the corpora from over 5 million books published up to the year 2008. By the end of 2013, Google N-gram viewer uses the Google Books' word-search database which is classified into 22 classifications based on some languages and years. Those 22 classifications are: English (American English, British English, English, English Fiction, British English (2009), English (2009), English Fiction (2009), English One Million (2009)), Chinese (Chinese (Simplified), Chinese Simplified (2009)), French, French (2009), German, German (2009), Hebrew, Hebrew (2009), Italian, Italian (2009), Spanish, Spanish (2009), Russian, and Russian (2009). The huge amount of the collected data in such corpora can have a great value and can be used to predict events to happen in the future. For instance, claims were made that mining such huge amount of data can predict very important future events such as Arab rising or Arabic spring [4].

Although Arabic is the fifth spoken language in the world with speakers of more than French, German, Russian and Italian languages, unfortunately, Arabic language is not included as one of the corpora indexed by the Google n-gram viewer. Accordingly, proposing a Google N-gram based viewer for Arabic corpus, would help in adopting Arabic as one of the approved corpora used by Google N-gram viewer.

The work presented in this paper aims to show that Arabic corpus can be developed and indexed, hence, can be incorporated in Google N-gram viewer. A dataset from books collected from open sources is used to demonstrate the showcase. The dataset is then

indexed based on different sizes of N-grams. The dataset used in this research consists of around 3000 books. The size of this database can be enlarged tremendously if the necessary resources and time are available to scan the huge amount of Arabic books available in the Arabic library over the long history.

The rest of the paper is organized as follows: section 2 presents the Google Books N-gram viewer. Section 3 presents the challenges of indexing and querying Arabic language. Section 4 discusses the new proposed N-gram viewer for Arabic Corpus and the initial results. Section 5 introduces experiments and analysis and section 6 presents the conclusion and future work.

II. LITERATURE REVIEW

English language corpus is one the largest corpora available on the web. Several English corpora exist that include Google Books N-grams data [2], commercial Google 1T 5-gram corpus with Web data. Different researchers discussed how to use and manage this corpus, see for example [5, 6], Microsoft Web N-gram Language Models. The use and application of Microsoft web N-gram is presented in [7] and English Giga-word corpus of newswire. An example of the usage of this corpus for language identification is presented in [8].

Different researchers in information technology used this tool to conduct research to answer different questions related to data retrieval, information extraction, text categorization and identifying trends related to certain topics. For instance, a new edition of the Google Books N-gram Corpus is presented in [3] which described how often words and phrases were used over a period of five centuries, in eight languages; it reflects 6% of all books ever published. The new corpus can facilitate the study of linguistic trends. Another example on the use of N-gram statistics and Google N-gram viewer is presented in [9] which is a culturomic study on a leading Bengali newspaper corpus that gave interesting insights into word usage and cultural shift in Bengali community, similar work is presented in [10] . A procedure to build language corpora by crawling and post-processing Web data is presented in [11]. The paper described the construction of a very large Italian general-purpose Web corpus (almost 2 billion words) and a specialized Japanese "blogs" corpus (about 62 million words) and discussed potential linguistics' applications. A tool is developed in [12] to work with enhanced web-scale N-gram corpora that include rich levels of source annotation, such as part-of-speech tags. A text mining approach based on N-Gram analysis is presented in [13] where the whole corpus of "Mining Software Repositories workshop (MSR)" papers is analyzed to identify combinations of keywords which frequently appear together in a paper. Another work to specify the frequency of the use of English personality adjectives is discussed in [14].

It is noted for a long time that Arabic NLP lacks resources such as corpora, lexicons, machine-readable dictionaries in addition to fully automated fundamental NLP tools [15]. Nowadays, more research and initiatives have been lunched to enrich Arabic corpora, e.g. [16], enhance the Arabic handwritten recognition, e.g. [17]. use of morphological decomposition strategies for Arabic Automatic Speech Recognition [18].

This paper is not about utilizing Google N-gram corpus or conducting textual analysis. Rather, it is about extending Google N-gram viewer to cover Arabic language. Subjects in this paper discuss the overall steps required to include Arabic N-gram in Google public datasets.

III. CHALLENGES OF INDEXING AND QUERYING ARABIC LANGUAGE

Building a large library of Arabic books and documents is a challenging task. Arabic language has some features that are somewhat unique. As such, most universally developed or available tools do not have support or handle such unique features. Arabic natural language processing challenges and solutions has been studied in [19]. Here we will describe some examples of those challenges.

A. *Unicode of characters*

UTF-8 encoding scheme that came to replace ASCII 128 characters in principle can handle non-Latin languages such as: Arabic and Hebrew. However in many programs Arabic fonts may not be recognized, hence language interpreter and index may retrieve unreadable letters or words. More recently, this issue is solved in new encoding schemes. Hence users or developers need just to make sure that they are using the right encoding scheme.

B. *Tashkeel, Harakat, or Diacritic marks.*

A unique aspect in Arabic is the addition of Diacritic marks or Tashkeel symbols (small وٍ, small أ, small ي , Tanween : ًٌٍ, stress or shaddah ّ). Those marks are added to letters to give different meanings or ways of spelling letters. They may cause problems to indexing and querying of data in cases whether those Diacritics are added or ignored. We will present few of the proposed solutions to handle Tashkeel in Arabic.

Papers which tried to detect Diacritics proposed morphological rules to do that. Known patterns and initial training sets are used to match subject word for the morphological rules. Elshafei et al (2006) proposed a statistical pattern recognition approach, using Arabic corpus of King Abdulaziz City for Science and Technology - KACST, to automatically detect or guess Diacritic marks from unmarked text. Hidden Markov

Model (HMM) is used where the hidden states are the possible diacritized selections of the words. In Arabic text, Diacritic marks can be challenging whether they exist or not in the Arabic text. If they exist, the challenge will be to read or parse them correctly especially as they need extra resources in addition to typical letters' parsing in all other languages. On the other hand, Diacritic marks are necessary to Arabic words to clarify words and statements from ambiguity. HMM algorithm is used in Gal in 2002 for Arabic and Hebrew languages. Accuracy reported was 94.5%. HMM is also used for Arabic Diacritic detection in some other research papers, see for example [20].

The case under consideration in this paper is a little bit different and related to the OCR or scanning process. Books that need to be scanned and transformed to soft versions come with different fonts, font styles, sizes and even different Diacritic styles.

C. *Different forms of letters based on their location or based on Tashkeel.*

Letters in Arabic have different forms based on their location, accordingly, indexer needs to understand that: (ا, أ,إ) are all forms of the same letter. Figure 1 shows a sample of a Sphinx table of Unicode character set that can be used as part of an indexer to guide the indexer for letters of different forms. The goal here is to acknowledge that many Arabic letters have different forms and hence a Unicode character table needs to have equivalents forms of similar letters. As mentioned earlier such issues are solved technically once the right encoding scheme is used.

D. *Text direction*

Arabic writing starts from right to left unlike Latin languages. This may also cause problems to typical indexing and retrieving processes especially if we want to support multi-word queries.

E. *Arabic Optical Character Recognition (OCR)*

OCR is the process of recognizing language letters from images. Actual books are scanned through scanners to produce images of those books in electronic formats. For those images to be recognized by information retrieval systems for searching and querying, OCRs are used to recognize letters and words from original images. Many of the evaluated open source and free tools seem to have problems supporting Arabic language and their accuracy is not very high.

Moreover, most programming languages stream readers show problems when reading some particular Arabic fonts especially when the original text is saved in Acrobat PDF format. Special processing is needed to correctly parse content from those books.

U+FBAF→U+06D2, U+FBB0→U+06D3, U+FBB1→U+06D3, U+FBD3→U+06AD, U+FBD4→U+06AD, U+FBD5→U+06AD, U+FBD7→U+06C7, U+FBD8→U+06C7, U+FBD9→U+06C6, U+FBDA→U+06C6, U+FBDB→U+06C8, U+FBDC→U+06C8, U+FBDD→U+0677, U+FBDE→U+06CB, U+FBDF→U+06CB, U+FBE0→U+06C5, U+FBE1→U+06C5, U+FBE2→U+06C9, U+FBE3→U+06C9, U+FBE4→U+06D0, U+FBE5→U+06D0, U+FBE6→U+06D0, U+FBE8→U+0649, U+FBFC→U+06CC, U+FBFD→U+06CC, U+FBFE→U+06CC, U+0621, U+0627..U+063A, U+0641..U+064A, U+0660..U+0669, U+066E, U+066F, U+0671..U+06BF, U+06C1, U+06C3..U+06D2, U+06D5, U+06EE..U+06FC, U+06FF, U+0750..U+076D, U+FB55, U+FB59, U+FB5D, U+FB61, U+FB65, U+FB69, U+FB6D, U+FB71, U+FB75, U+FB79, U+FB7D, U+FB81, U+FB91, U+FB95, U+FB99, U+FB9D, U+FBA3, U+FBA9, U+FBAD, U+FBD6, U+FBE7, U+FBE9,U+FBFF

Figure 1: A sample of Arabic Unicode letter equivalency table.

Many Arabic books may also mix their content with English words or statements.

There have been several trials to handle or improve Arabic OCR. An OCR system for Arabic and English based on HMM and Omni is proposed in [21]. Authors reported 3.3 % error rate for Arabic dataset. [22] has performed a comparison between two OCRs used for Arabic: OmniPage and Sakhr OCRs where several metrics related to accuracy (i.e. ability to predict correct letters) and performance are used. Sakhr product showed a better accuracy. However, recently Sakhr Company and their OCR are not available or supported.

[23] discussed one challenge with Arabic writing related to connecting letters with each other which makes it difficult for an OCR to segment or detect words letter by letter.

In our case, we used some open source OCR implementations. However, these implementations are not optimized for Arabic language. Accuracy and OCR performance are the main two criteria to evaluate in the OCR process. The main challenge we noticed in this regard is that when using a large set of books coming from different text sources with different fonts, OCR process optimization is a challenge as a single optimized OCR process may work well for some books and not for others. This is especially true where Arabic books are usually written in non-standard fonts.

IV. New N-gram based indexer for Arabic Corpus: A showcase

In the following section, we will present the development of an N-gram based Arabic indexer. Ultimately, the goal is to collect and index a large amount of Arabic books from early and medieval time to present time. This is considered a distinguished contribution where major research initiatives focused largely on Arabic e-content. The second ambitious goal is to include Arabic language as one of the

adopted languages by Google N-gram viewer and be able to expand the content frequently. Specifically, we intended to present a showcase on how such process can be incorporated in Google N-gram Viewer.

A. Building the sample of Arabic corpus

There have been some trials to build online corpora for Arabic content that can be useful for research and not only for data downloading and browsing. Few of these corpora can be considered in comparison with proposed Arabic Google N-gram viewer where the corpus contains a large amount of Arabic books, indexed in a format that can be integrated easily with N-gram based viewer. The corpus is expected also to give users the ability to make rich customized queries. Those queries should accept more than one word and can retrieve content and metadata related to date, authors and other related information.

A list of free and commercial Arabic datasets is available at [24]. However none of them is shown to be up-to-date or active. Other practical trials to build interactive Arabic corpora include: King AbdulAziz City for Science and Technology Arabic Corpus [25] Saudi Digital Library [26], and Arabic Corpus [27].

A summary of these three Arabic corpora is given in Table-1. This simple evaluation study for those Arabic corpora showed that some of the sought characteristics similar to that of Google N-gram viewer are not available in the investigated corpora.

Unfortunately, the currently available Arabic corpora do not contain N-gram based indexers for customized search queries. They may only contain simple string search based on single keywords. However such search is not meant for researchers in data mining or machine learning. Unlike public library researchers who are looking for particular information in one or more books, linguistic data miners are looking for aggregated words and results based on rich or complex queries.

The dataset built in this research work that contains around 3000 Books written in Arabic language have been collected from the following Arabic electronic libraries' websites: http://www.alwaraq.net, http://www.arab-book.com/ebook/, http://www.aljlees.com/, http://abooks.tipsclub.com/, http://www.arabicebook.com/, http://4kitab.com/, http://arablib.com/, http://www.wdl.org/ar/, http://al-mostafa.com, http://www.almeshkat.net, http://arabicorpus.byu.edu and http://www.irtipms.org.

Table-1 Summary of the Arabic Corpora

Criteria \ Corpus	KACST Arabic Corpus	Saudi Digital Library	Arabic Corpus
Public	Yes	No	Yes, needs login
Size	700 Million Words	242,000 References	173,600,000 Words
Type of contents	Books, newspaper, reports, Academic Theses	Books, Academic Theses	Largely from Egyptian newspapers, some pre modern, modern and nonfiction books
Contents language	Arabic	Arabic and English	Arabic
Searchable	Yes	Yes	Yes
Search speed	Fast	Fast	Slow
Statistics available	Yes, Readymade (fixed)	No	No

The books format is either PDF or Word. For each one of these types, special stream readers are developed to make sure that Arabic text is completely parsed correctly. For books in PDF format, several open source PDF parsing public libraries such as: ItextSharp, PDFBox, IKVM are used. The main application is developed using C# and .NET environment. Books may also include irrelevant text in the cover pages, e.g. logos, which may need to be trimmed. Special characters that are not Arabic letters or numbers should be also trimmed. A list of stop words and characters is compiled to be eliminated from the parsed text.

B. Data Preprocessing

There are several pre-processing activities that are necessary to properly store the content of the collected books. Typically this may include: Text parsing, stemming, tokenizing, filtering and eliminating noise data. The following activities are executed in order for each new Arabic-book to be added to the library:

- OCR stage: This is particularly for books in PDF, or any other image, format. Arabic books to be scanned and indexed may exist in

different formats. These include hard copies, or manuscripts. These also include soft copies of: Acrobat PDF, Word documents, images and texts. In principle, each type of these formats requires different initial preprocessing activities. For example, hard copies and manuscripts should be scanned first. Based on the OCR type and process, scanned books or materials can be converted to: Image, Acrobat, text or rich text files.

- Text Parsing: Text from scanned books should be parsed word by word. Lucene open source library information system is used [28]. The package includes components to perform the different processes such as parsing, stemming and tokenization. Typically, the process of stemming includes finding the roots for words. However, in this N-gram collection, words will be stemmed into their N-grams. Following Google approach, we stem words based on word N-grams. For example a three-gram directory will include stored text in the index based on the occurrence and repetition of three-words batches. Moreover, we have evaluated our work using other indexers such as: RavenDB and algorithms described in Huston et al (2011) paper to efficiently find repeated N-grams in a corpus.

- Meta data: In addition to the content of the corpus, the index system collects Meta data related to the content and the process. This includes: Book title, author(s), year (s) of writing, publisher, publication year and number of pages. All these information may not be necessarily retrieved with search query results. However, they may be used for customized queries or reports.

These two major processes are required to include new books or content to the corpus. Other language processing activities are required when performing users' query search.

C. Data Indexing

"Indexing is the process of converting text data into a format that facilitates rapid searching" [29]. The performance and accuracy of retrieved query results depend largely on the quality of the Indexer module. Typical to traditional library indexes, the electronic library index stores, in one or more files, pointers to the actual data in the documents or books' dataset. These pointers can be letters, words, or terms of several words. Indexers built originally for other languages, especially Latin languages, need to be customized to accommodate features unique in Arabic language such as those mentioned previously. We indexed the collected documents using Lucene open source indexer which includes several indexing algorithms, from which we specifically used inverted tree and N-gram indexing algorithms (discussed in section 4.3.1 and 4.3.2).

Prefix and Suffix trees are also used to store data index. The goal is to be able to store and retrieve queries from that data in quick and reliable manners. In principle, the structure of the tree depends on having letters in nodes at the different levels to point to the actual words in the leaves. Documents or books are stored in the index as objects.

In section 5.3, we evaluate the developed system's performance which includes evaluating the developed index. The index should further allow users to perform different customized search queries in comparison with simple string searches. Due to the fact that many words have common sub parts or terms, the indexer does not need to store all dataset content word by word. Encoding and compression can be also used to reduce index size and optimize storage

4.3.1 Inverted index: Binary or Balanced trees (B-tree) or hash tables can be used to store or build indexes. They can be also utilized to optimize index tree size [30]. The inverted index tree contains a list of words and their occurrences or locations in documents. Figure 2 illustrates the pseudo code for the developed inverted index.

Figure 3 shows a simple example or a snapshot of the content of an inverted tree index. Words and N-grams are sorted in the index, with number and location of occurrences in corpus documents.

```
For each document d in the collection {
 For each term t in document d {
Find term t in the term dictionary
If term t exists, add a node to its posting list
Otherwise, –Add term t to the term dictionary
Add a node to the posting list
    }}
After all documents have been processed, write the inverted index to disk
```

Figure 2: Pseudo code for the developed inverted index

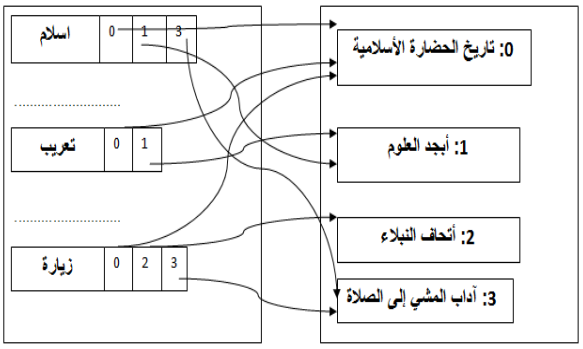

Figure 3: A simple example of Inverted tree content

4.3.2 N-gram Index: N-gram can be used for queries, information retrieval and text mining activities. N-grams of 2 (Bi-grams, 3-grams ,4-grams, etc.) can be applied for either letters or words.

Major goal of N-gram based indexing and search is to get fast search results as you type the query. N-gram can be used for letters or for words. Google N-gram indexing uses grams of: 2,3,4,5, and so on.

N-gram can be also classified within inverted index types. The difference is that typically words are indexed while in N-gram indexing grams of words or letters are the roots for the indexing process.

4.3.3 Document term index: This index can simply index words or terms, their frequency and occurrences. We noticed that Google N-gram viewer focuses on indexing words, their frequency and location along with the data attributes.

A vector space model (VSM) is developed to match between terms and documents. Values in this model represent the frequency of the terms in the document.

V. Experiments and Analysis

In the previous section we have discussed the development of a system to perform the tasks described earlier: Arabic books' dataset collection, and preprocessing, building a library or directory index, building an application including a user interface to perform simple and complex query search. Further the application includes generic and customized statistical reports.

The 3000 books are transformed correctly with the right encoding. Retrieved results from the search index include fields generic or specified by searcher. For example, Figure 4 shows for each document that contains the query: Number of pages, matching score to the query, document ID and location.

The score column represents the matching score which is calculated with a value between 1 (perfect match) and zero (no match). The retrieved results display the top search results based on the matching score. When Lucene library calculates the score, it finds each individual score for term hits within field contents and gets their total. If the highest ranked hit has a total that is greater than 1, all of the document scores are re-normalized to be between 0 and 1, with the highest ranked document having a score of 1. On the other hand, if no document's total was greater than 1, no normalization occurs and the scores are returned as-is. This justifies why the matching score of the same document may vary slightly based on the normalization process.

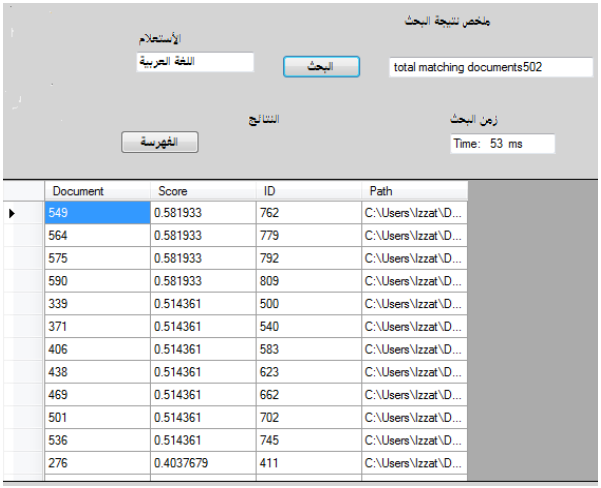

Figure 4: Examples of retrieved results

The score value is calculated by Lucene search and indexing library, using inverted index algorithm. Such similarity is calculated based on the percentage of the match between query and retrieved results. Such score value needs extensive evaluation and investigation in future.

Typically when adding a new document or book to the indexer certain Meta data should be added during the insertion process. A typical code to do this may look like the following:

```
Document doc = new Document();
doc.Add(new Field("path", f.FullName,
Field.Store.YES, Field.Index.NOT_ANALYZED));
doc.Add(new
Field("id",id.ToString(System.Globalization.CultureI
nfo.InvariantCulture),

doc.Add(new Field("modified",
DateTools.TimeToString(f.LastWriteTime.Milliseco
nd,
DateTools.Resolution.MINUTE), Field.Store.YES,
Field.Index.NOT_ANALYZED));
Treader = new FilterReader(fileNameSource);
NumericField frequencyField = new
```

```
NumericField("frequency", Field.Store.YES, true);
doc.Add(new Field("contents", Treader,
Field.Store.YES,Field.Index.ANALYZED
));
```

After defining a new document to be added to the index, lines of Meta data can be added once at a time. As mentioned earlier this may include: File name, book name, ID, insertion date and publication date. File name here is different than Book name as collected books have file names that are not necessary the same as the book title. Some of these Meta data are related to the insertion process while others are related to the document or the book itself.

The line (Treader = new FilterReader(fileNameSource);) includes defining a reader for the book. We used different parsing libraries to read book files based on their text format and type (e.g. .pdf, .doc. .txt, etc.). We also used special libraries to be able to read or encode Arabic text. Note also that some fields can be used in the indexes and others cannot. Certain attributes such as the book title can be used to index data. Usually such selection depends on the uniqueness of the attribute vales along with some other selection criteria. This can be user defined based on what kind of queries to optimize. This line will then be called in a loop to go through the document, word by word and each word can be then inserted in the inverted index. Frequency field represents the number of times the term exists in the document.

The above index pseudo code adopts inverted index algorithm that is presented as part of Lucene libraries. The following line shows that the subject word is to be indexed in the index. As mentioned earlier, this is user defined based on what should be indexed or not.

```
document.add(new Field("ngram", ngram,
Field.Store.YES, Field.Index.ANALYZED));
```

NGramTokenizer class should be called first to stem or parse each document into N-grams. The same indexing process can be repeated for the search or the query. It is usually better to use the same indexing approach for data indexing and search or query.
The process to perform query search includes the following steps:

1. Call indexer to the memory (read only)
2. Call query reader to read query and prepare it for the search process
3. Call index searcher class to carry query to the search index
4. Retrieve results and present them to the user properly

For queries with more than one word, an algorithm should be defined to rank retrieved matches. In our experiments, the following is used:

1. First retrieve documents that contain all the words occurring together in the same sequence as the query.
2. Second retrieve documents that contain maximum number of words in the same sentence
3. Third retrieve documents that contain all the words but not necessary in the same sentence or sequence.
4. Last, retrieve documents that contain at least one or two of the query words. However, this may not always be called if retrieved result is large

N-gram indexing gives then more preference to the same sequence of words as in the query. This is especially true if query size is less than the N-gram size. For N-gram search, frequency of occurrences can be used also in ranking the retrieved results. For example, an N-gram with 200 frequencies should get a higher score than an N-gram with 100 frequencies

Indexers or lexicons are stored in memory. Size of the indexer can be changed based on the indexing algorithm. Such size may have direct impact on performance or speed of query retrieval. To build N-gram-based indexer, we followed the tasks described in Coetzee (2008) and shown in Figure 5.

A. Arabic OCR

A good percentage of Arabic books available online is in an image PDF format. This format is not easily readable by IR systems. We used several libraries for OCR in Arabic such as: Tesseract. However we noticed that most evaluated OCRs for Arabic have serious issues with both accuracy and performance. Those are two criteria that are usually used to evaluate the quality of the OCR process:

```
1  function ConstructIndexSimple(D, n_min, n_max, t
2    for n from n_min to n_max:
3      for d from 1 to |D|:
4        for i from 1 to |D_d|:
5          s ← D_d[i..i + n − 1]
6          Add d to P(s)
7      for each s where |s| = n and |P(s)| > 0:
8        Q(s) ← ⋂ P(s') over substrings s' of s in
9        if |Q(s)| − |P(s)| ≤ t
10         Discard P(s)
11       else
12         L ← L ∪ {s}
13   return {L, P}
```

Figure 5: General N-gram index construction process [31]

B. Reports

As part of the library system, users can utilize online queries (i.e. real time queries) for top letters and words. Inspired by Google N-gram, Top 10 grams for letters and words can be retrieved (see Figure 6). This includes grams of words and letters from 1 to 5. N-gram indexing is considered language independent.

Figure 6: Arabic Library reports

General reports include statistics related to attributes collected about books. This includes reports related to Books: Authors, years of writing, years of publications, and possibly subjects' classifications.

Customized reports include similar information to that of general reports with the exception that users can define report elements (e.g. search for authors from a specific period, authors, book and titles. with searched for keywords, etc.).

C. System Evaluation

Although the collected set of books forms a small sample to make a general evaluation for aspects such as: performance, the system is evaluated for performance. The performance is measured by calculating the time it takes to retrieve results (see Table 2). The experiments showed that indexing and retrieval time is fast which is expected for a mature indexer such as Lucene. . However, we acknowledge that we still have a relatively small size dataset. Hence such values need to be re-evaluated for a dataset that includes larger number of Arabic Books.

In testing the performance, focus was given to complex queries (i.e. queries that include more than one word). This is because almost all of the evaluated Arabic online libraries were missing the ability to conduct complex query search. Table 2 below shows evaluated search queries with number of retrieved results and processing time. Accuracy evaluation is conducted manually to compare retrieved results with actual files or books. Manual evaluation showed some differences in queries results between manual investigation through books and retrieved search queries. However, this needs to be thoroughly investigated to be able to quantify system accuracy.

Table 2: Search Queries Evaluation

Query	NO of retrieved results	Processing time (ms)	Query	NO of retrieved results	Processing time (ms)
الله	1177	53	عليه السلام	1029	140
مكة	374	123	اللغة العربية	792	94
العراق	226	77	علوم النحو والصرف	477	193
الأعراب	124	61	صلى الله عليه وسلم	1192	236
الفاعل	269	128	عليه الصلاة والسلام	1019	250
الجمل	227	178	ابيات الشعر	384	603

VI. LIMITATIONS

This study aims to show the applicability of Arabic language to be added as a corpus to Google N-gram viewer. The following issues are considered as

limitations to this study that need further study and research:

1. The study did not explore all possible Arabic corpora. A separate study is needed to evaluate and compare among all Arabic corpora.
2. As the proposed solutions aims to examine the applicability of the Arabic language to be used by Google N-gram viewer, no claim is made about the performance of our solution. Where the performance and optimization issues are subject of other studies.
3. More quality tests are needed to test the accuracy of the retrieved results. Such tests are not discussed in this paper due the limited number of allowed pages.
4. Increasing the corpus size requires enough funding which is not available right now. Funding such research work is essential to be able to digitize the huge amount books available in the Arabic literature.

VII. Conclusion and Future Work

In this paper, we discussed developing and indexing Arabic corpus to show that Arabic language can be easily incorporated in Google N-Gram viewer that includes a large number of books of different languages (which is currently excluding Arabic). A major goal is to disseminate Arabic content, specially the large amount of ancient books, making them publically available for researchers in different fields. A dataset of Arabic books is assembled. Open source Lucene indexer and information retrieval system is used. Indexing and retrieval processes are evaluated using the collected dataset of 3000 Arabic books. Evaluation includes: Evaluating performance of the indexing and retrieval processes. It also includes manual verification for the correctness of some selected search queries. In order to reach a large significant size of collected Arabic books similar to the size archived by Google for English and other languages, we should scan a large amount of Arabic books. Further, OCR processes should be conducted while making sure that converted books are correct in their electronic format.

References

[1] S. Grabowski and J. Swacha, "Google Books Ngrams Recompressed and Searchable," *Foundations of Computing and Decision Sciences December 2012,* vol. 37, pp. 271-281, 2012.

[2] J. B. Michel, Yuan K. Shen, Aviva P. Aiden, Adrian Veres, Matthew K. Gray, and J. P. P. The Google Books Team, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, Erez L. Aiden, , "Quantitative Analysis of Culture Using Millions of Digitized Books " *Science,* vol. 331, pp. 176-182, 2011.

[3] Y. Lin, J.-B. Michel, E. L. Aiden, J. Orwant, W. Brockman, and S. Petrov, "Syntactic annotations for the Google Books Ngram Corpus," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics* Jeju Island, Korea: Association for Computational Linguistics, 2012, pp. 169-174.

[4] P. Ball, "News mining might have predicted Arab Spring," in *Nature* Published online 13 September 2011, 2011.

[5] S. Evert, "Google Web 1T 5-Grams Made Easy (but not for the computer)," in *Proceedings of the NAACL HLT 2010 Sixth Web as Corpus Workshop*, Los Angeles, California, 2010, pp. 32-40.

[6] A. ISLAM and D. INKPEN, "Managing the Google Web 1T 5-gram Data Set," in *International Conference on Natural Language Processing and Knowledge Engineering, 2009. NLP-KE 2009* Dalian , China, 2009, pp. 1-5.

[7] K. Wang, C. Thrasher, E. Viegas, X. Li, and B.-j. Hsu, "An overview of Microsoft web N-gram corpus and applications," in *Proceedings of the NAACL HLT 2010: Demonstration Session* Los Angeles, California: Association for Computational Linguistics, 2010, pp. 45-48.

[8] E. Tromp and M. Pechenizkiy, "Graphbased n-gram language identification on short texts," in *In Proc. 20th Machine Learning conference of Belgium and The Netherlands*, 2011, pp. 27-34.

[9] S. Phani, S. Lahiri, and A. Biswas, "Culturomics on Bengali Newspaper Corpus," in *International Conference on Asian Language Processing* Hanoi, Vietnam, 2012, pp. 237-240.

[10] K. Leetaru, "Culturomics 2.0: Forecasting Large-Scale Human Behavior Using Global News Media Tone In Time And Space," *First Monday,* vol. 16, 2011.

[11] M. Baroni and M. Ueyama, "Building general- and special-purpose corpora byWeb crawling," in *Proceedings of the 13th NIJL*

International Symposium Tokyo, Japan, 2006, pp. 31-40.

[12] Y. Lin, J.-B. Michel, E. L. Aiden, J. Orwant, W. Brockman, and S. Petrov, "Syntactic Annotations for the Google Books Ngram Corpus," in *the 50th Annual Meeting of the Association for Computational Linguistics*, Jeju, Republic of Korea, 2012, pp. 169–174.

[13] S. Demeyer, A. Murgia, K. Wyckmans, and A. Lamkanfi, "Happy birthday! a trend analysis on past MSR papers," in *Proceedings of the 10th Working Conference on Mining Software Repositories* San Francisco, CA, USA: IEEE Press, 2013.

[14] E. Roivainen, "Frequency of the use of English personality adjectives: Implications for personality theory," *Journal of Research in Personality,* vol. 47, pp. 417–420, 2013.

[15] Diab M, Hacioglu K, and J. D, "Automatic processing of modern standard Arabic text. In: Soudi A, van den Bosch A, Neumann G (eds) Arabic Computational Morphology: Knowledge-based and Empirical Methods, Text, Speech and Language Technology," *Springer Netherlands,* pp. 159-179, 2007.

[16] Alghamdi M, Chafic M, and M. M, "Arabic language resources and tools for speech and natural language: KACST and Balamand," in *2nd International Conference on Arabic Language Resources and Tools* Cairo, Egypt, 2009.

[17] Habash N and Roth R, "Using deep morphology to improve automatic error detection in Arabic handwriting recognition.," in *49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* Stroudsburg, PA, USA, 2011, pp. 875-884.

[18] Diehl F, Gales M, and T. M, "Morphological decomposition in Arabic ASR systems," *Computer Speech and Language,* vol. 26, pp. 229-243, 2012.

[19] A. Farghaly and K. Shaalan, "Arabic natural language processing: Challenges and solutions," *ACM Transactions on Asian Language Information Processing (TALIP),* vol. 8, 2009.

[20] R. A.-H. Mohamad, C. Mokbel, and L. Likforman-Sulem, "Combination of HMM-Based Classifiers for the Recognition of Arabic Handwritten Words," in *9th International Conference on Document Analysis and Recognition, ICDAR 2007*, Curitiba, Brazil, 2007, pp. 959-963.

[21] I. Bazzi, R. Schwartz, and J. Makhoul, "An omnifont open-vocabulary OCR system for English and Arabic," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 21, pp. 495-504, 1999.

[22] T. Kanungo, G. Marton, and O. Bulbul, "OmniPage vs. Sakhr: Paired Model Evaluation of Two Arabic OCR Products," in *SPIE Conference on Document Recognition and Retrieval*, San Jose, CA, 1999.

[23] A. Zidouri, M. Sarfraz, S. A. Shahab, and S. M. Jafri, "Adaptive Dissection Based Subword Segmentation of Printed Arabic Text. IV 2005: 239-243," in *Ninth International Conference on Information Visualisation*, London, England, pp. 239 - 243

[24] L. Al-Sulaiti, "Arabic Corpora." vol. 2014, 2010.

[25] KACST, "King AbdulAziz City for Science and Technology Arabic Corpus ", 2013.

[26] SDL, "Saudi Digital Library," 2014.

[27] D. Parkinson, "Arabic Corpus."

[28] Y. Seeley, "Solr Architecture." vol. 2014, 2009.

[29] A. Sonawane, "Using Apache Lucene to search text," IBM technical library, http://www.ibm.com/developerworks/library/os-apache-lucenesearch/18 August 2009 2009.

[30] A. Al-Jedady, M. Al-Kabi, and I. Alsmadi, "Fast Arabic Query Matching for Compressed Arabic Inverted Indices," *International Journal of Database Theory and Application,* vol. 5, pp. 81-94, 2012.

[31] *D. Coetzee*, *"TinyLex: static n-gram index pruning with perfect recall," in ACM 17th Conference on Information and Knowledge Management* California, USA, 2008, pp. 409-418.