Google N-Gram Viewer does not Include Arabic Corpus! Towards N-Gram Viewer for Arabic Corpus

Izzat Alsmadi[1] and Mohammad Zarour[2]
[1]Department of Computing and Cyber Security, Texas A&M University, USA
[2]Information Systems Department, Prince Sultan University, KSA

Abstract: *Google N-gram viewer is one of those newly published Google services. Google archived or digitized a large number of books in different languages. Google populated the corpora from over 5 million books published up to 2008. This Google service allows users to enter queries of words. The tool then charts time-based data that show the frequency of usage of query words. Although Arabic is one of the top spoken language in the world, Arabic language is not included as one of the corpora indexed by the Google n-gram viewer. This research work discusses the development of large Arabic corpus and indexing it using N-grams to be included in Google N-gram viewer. A showcase 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 number of people who speak Arabic. We believe that adopting Arabic language by Google N-gram viewer can significantly benefit researchers in different fields related to Arabic language and social sciences.*

Keywords: *Arabic language processing, corpus, google N-gram viewer.*

Received May 7, 2015; accepted September 20, 2015

1. Introduction

The size of data available on digital format continues to increase enormously. The dissemination and use social networks, mobile devices and Internet content played the main role in data explosion we see nowadays. As stated by [13], big data is about to significantly change everything, in research domains and industries such as linguistics, genomics, e-commerce, Internet search, automated translation and social networking. To be able to index and analyse text, this text is parsed into documents, sentences, words, N-grams, etc.,

Google N-gram viewer is one of those newly published Google services. Google archived or digitized a large number 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, the service has 22 classifications based on several 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). Arabic language and literature has gained widespread popularity for hundreds of years among not only Muslims but also Christians and Jews [9]. One of the estimates stated that by the year of 2000, there were around 220 million native Arabic speakers, 450 million Arabic speakers and around billion and a half Muslims who used Arabic as their religious language [9]. Arabic language is one of the top spoken languages in the world with speakers of more than French, German, Russian and Italian languages. Unfortunately, and despite of these facts, 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 significantly if the necessary resources and time are available to scan the huge amount of Arabic books available in the Arabic library over a 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.

2. 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 [19], commercial Google 1T 5-gram corpus with Web data. Different researchers discussed how to use and manage this corpus, see for example [10, 12], Microsoft Web N-gram Language Models. The use and application of Microsoft web N-gram is presented in [30] and English Giga-word corpus of newswire. An example of the usage of this corpus for language identification is presented in [29].

Google N-gram viewer is a text mining visual tool that uses Google N-gram data and helps researchers to explore trends of terms and phrases over time.

Figure 1 shows a simple example of using Google N-gram viewer for English corpus with a query of three terms: Islam, Arab, Saudi Arabia. In addition to selecting the corpus, the user can select the year range, and the smoothing level. For example, a smoothing selection of 1 means that the data shown for 1950 year will be an average of the raw count for 1950 with 1 margin on either side (i.e., 1950 plus and minus one year). For an event that occurs in a specific year, the smoothing level can be minimized. On the other hand, if this is a long time subject, smoothing level can be maximized. Zero means no smoothing just raw data.

The example in Figure 1 shows how the time information can be interpreted.

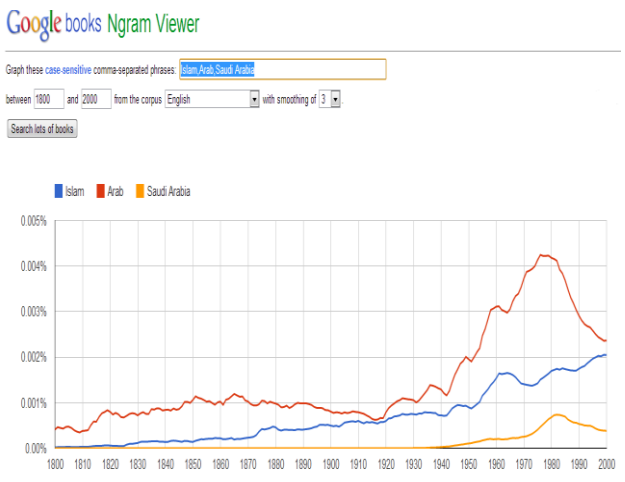

Figure 1. An example of using google n-gram viewer.

However, these terms are getting recently more hits starting from 1950s (as we started seeing books about Arabic countries after occupation, oil discovery and current Middle East conflicts). On the other hand, the term (Saudi Arabia) clearly has no single hit before 1944, twelve years after the birth of the current Saudi Arabia state and hence the term. Such simple example can show how a large corpus of data could be utilized for useful knowledge or information extraction. This N-gram dataset contains English word N-grams and their observed frequency counts.

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 [17, 18] 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 will facilitate the study of linguistic trends. Another example on the use of N-gram statistics and Google N-gram viewer is presented in [21] which is a culturomics study on a leading Bengali newspaper corpus that gave interesting insights into word usage and cultural shift in Bengali community. Studying the cultural evolution by analysing the content of Millions of books, using the Google Ngram Viewer has been studied in [10]. Another interesting work that uses google English books corpus to study how to improve paper's citations is discussed in [17]. A procedure to build language corpora by crawling and post-processing Web data is presented in [2]; 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 [18, 19] 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 [7] where the whole corpus of "Mining Software Repositories workshop (MSR)" papers is analysed 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 [23].

Google N-Gram datasets are used to study some computation problems, using Hadoop and MapReduce, related to calculating the total number of words used per year and the average length of words used per year [4].

A similar study that used the Google Books corpus as a tool to study various linguistic trends of the English language is given in [20].

A similar work of what is proposed in this paper is presented in [5] where the authors discussed a project in the National Library of Norway, to develop a service comparable to the Google N-gram Viewer called (NB N-gram). It is based on all books and newspapers digitized at the National Library of Norway. Uni-, bi- and trigrams have been generated based on this text corpus containing some 34 billion words. Another similar work creates N-gram collection based on a large-scale corpus of all Polish sites on the Internet for the benefit of many applications in machine learning [24].

This paper is not about utilizing Google N-gram Viewer data or conducting textual analysis. Rather, it is about extending Google N-gram viewer to cover Arabic language following Google approach.by using words N-gram. Initial results of this research work have been presented in [31].

3. Challenges of Indexing and Querying Arabic Language

Building a large library of Arabic books and documents is a challenging task. Arabic language has some unique features. As such, most universally developed or available tools do not have support or handle such unique features. Here we will describe some examples of those challenges.

3.1. 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.

3.2. Tashkeel, Harakat, or Diacritic Marks

A unique aspect in Arabic is the addition of Diacritic marks or Tashkeel symbols (small و, small أ, small ي, Tanween: ٍ ٌ ً, stress or shaddah). Those 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 an initial training set are used to match subject word for one of the morphological rules. Elshafei *et al.* [8] (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 [22].

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.

3.3. 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: (أ ,إ , آ) are all forms of the same letter. Figure 2 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 equivalent forms of similar letters. As mentioned earlier such issues are solved technically once the right encoding scheme is used.

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 2. A sample of arabic unicode letter equivalency table.

3.4. 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.

3.5. Arabic Optical Character Recognition

Arabic Optical Character Recognition (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 image or Acrobat PDF formats. Special processing is needed to correctly parse content from those books. Many Arabic books may also mix their content with English words or statements.

There have been several trials to handle or improve Arabic OCR process.OCR system for Arabic and English based on HMM and Omni is proposed in [3] or based on Markov models, Touj *et al*. [28] 2005, or Dynamic Time Warping (DTW) algorithm, Khemakhem and Belghith [16], 2009. Authors reported 3.3 % error rate for Arabic dataset. A comparison that compared between two OCRs used for Arabic: OmniPage and Sakhr OCRs is given in [15], 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.

One challenge with Arabic writing is related to connecting letters with each other which makes it difficult for an OCR to segment or detect words letter by letter as discussed in [32].

In our case, we used several open source OCR implementations. However, these implementations are not optimized for Arabic language. The implication of this is that there is a need to build an Arabic OCR or extend existing OCRs to include libraries dedicated to deal with Arabic language and some of its unique features such as Tashkeel or the large number of different fonts. This should be an important objective when we build the large Arabic corpus.

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. Optimizing Arabic OCR implementations however is out of this research paper scope.

4. A 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.

The steps adopted to develop the Arabic Ngram indexer and information retrieval system are not unique in terms of what components should be included as those exist in all similar systems. These include: Data pre-processing and indexing. In the indexing section, we describe how to implement two popular methods of indexing, Inverted index and Ngram. In another step, we show how a sample dataset of Arabic corpus can be collected and indexed. Information in each step is what unique and not those steps as they are typical to most IR systems.

4.1. 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. These queries should accept more than one word and can retrieve content and metadata related to date, authors and other related information.

Online sources include different attempts with different sizes to build Arabic corpuses. Some of the significant attempts to build interactive Arabic corpora include: king Abdelaziz city for science and technology Arabic corpus [14] Saudi digital library [25], and Arabic corpus [20].

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

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	No	No

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 and are available online. The books 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.

The books format is either PDF or Word as they are the two popular formats used to save text files on the web. For each one of these type formats, 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 as they are available freely on the web. The main application is developed using C# and .NET technology as the authors are expert in it. Books may also include irrelevant text in the cover pages, e.g., logos, which may need to be removed. Special characters that are not Arabic letters or numbers should be also removed. A list of stop words and characters is compiled to be eliminated from the parsed text. Nonetheless Ngram based index and retrieval is typically immune to stemming stop words where research showed that stemming stop words in N-gram based language processing can have a little impact.

4.2. Data Pre-Processing

Several pre-processing activities 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 scanned books that are 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 [26]. 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 will 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*. [11] 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. These include: Book title, author(s), year(s) of writing, publisher, publication year and number of pages. Not all these information may 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.

4.3. Data Indexing

"Indexing is the process of converting text data into a format that facilitates rapid searching" [27]. 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. Lucene open source indexer includes several indexing algorithms, from which we can specifically use inverted tree and N-gram indexing algorithms (discussed in section 4.3.1. and 4.3.2.), accordingly we decided to use Lucene indexer to complete data indexing. The overall data flow activities that occur in Lucene search and indexing system includes three major tasks: Data indexing, searching and results retrieval and then display results.

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 and accuracy 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 [27].

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 [1, 2]. The inverted index tree contains list of words and their occurrences or locations in documents. Figure 3 illustrates the pseudo code for the developed inverted index.

```
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 3. Pseudo code for the developed inverted index.

Figure 4 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.

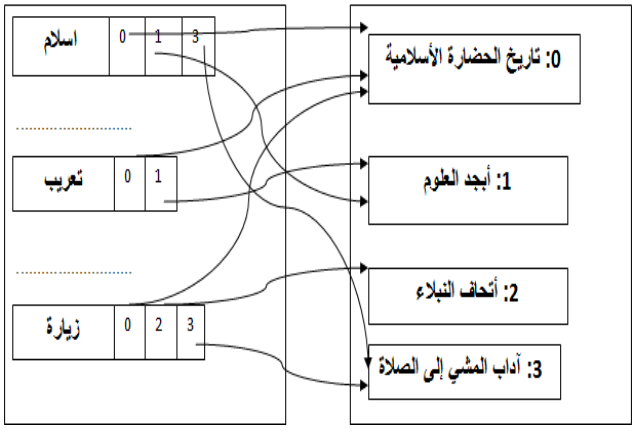

Figure 4. A simple example of Inverted tree content.

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.

5. 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 pre-processing, 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 implications of the results presented in this section showed that Arabic, is not different from other languages already indexed in Google N-gram data-set and that given the collection of Arabic contents specially from the large number of books, it is possible for Google to add a large and rich data-set representing Arabic language and content.

The 3000 books collected in our experiment are transformed correctly with the right encoding. Retrieved results from the search index include fields generic or specified by searcher. For example, Figure 5 shows for each document that contains the query: Number of pages/occurrences, matching score to the query, document ID and location. (The developed program shown in Figure 6 is developed for testing our system only; the design of actual system should imitate Google N-gram viewer architecture).

The score column represents the matching score which is calculated with a value between 1 (perfect match) and zero (no match). The retrieved results represent the top search results based on the matching score.

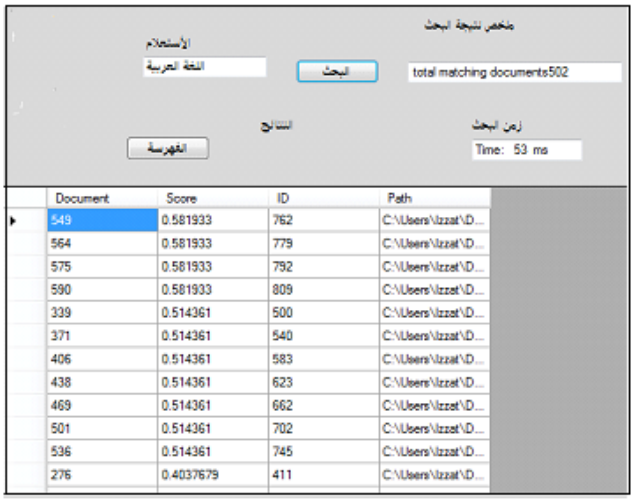

Figure 5. Examples of retrieved results.

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 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 (i.e., due to the normalization process).

The similarity 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 Lucene index 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.CultureInfo.InvariantCulture),
doc.Add(new Field("modified",
DateTools.TimeToString(f.LastWriteTime-
.Millisecond, 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));
```

Figure 6. Part of the developed program.

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 can typically be different than Book name as collected books have logical 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.

The line (Treader = new Filter Reader (file Name Source);) 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 users defined based on what should be indexed or not.

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.

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

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 an N-gram-based indexer, we followed the steps described in [6] and shown in Figure 7.

5.1. 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 some issues with both accuracy and performance.

```
function ConstructIndexSimple(D, n_min, n_max, t):
  for n from n_min to n_max:
    for d from 1 to |D|:
      for I from 1 to |Dd|:
        s ← Dd[i..i + n-1]
        Add d to P(s)
      for each s where |s| = n and |P(s)| > 0:
        Q(s) ← ∩ P (S') over substring S' of S in L
        if | Q(s) | - P(S) ≤ t
          Discard P(s)
        else
          L ← L ∪ {S}
  return {L, P}
```

Figure 7. General N-gram index construction process [6].

5.2. Reports

As part of our developed 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 8). This includes grams of words and letters from 1 to 5.

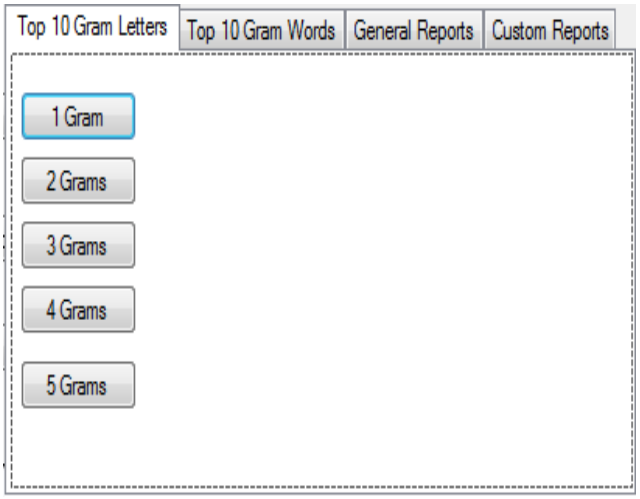

Figure 8. Arabic library reports.

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.,). This reporting system is evaluated in our developed library system although Google N-gram Viewer is not including such option.

5.3. 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 and accuracy based on the collected set to give indications of the quality of the developed system. 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.

Search results retrieval can be compared or evaluated from different perspectives. The first perspective is accuracy of retrieved results. Perfect accuracy of the indexer means that the indexer should firstly: retrieve search terms positively, where all retrieved terms correctly exist in the dataset and in the right retrieved location. Secondly: it should not miss any word that is supposed to be retrieved. We conducted this accuracy evaluation manually on a small scale size of the dataset since doing this for a large dataset can be time consuming. Accuracy evaluation showed that system accurately can index and retrieve text without significant problems.

The second accuracy measure related to the query, which is measured by Lucene library index, is the similarity index. This similarity index is a measure between searched for queries and retrieved results. We also evaluated this library value on a small dataset manually and results showed to be always correct.

The last metric of measurement for the quality of the indexing process is 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 a larger number of Arabic Books.

Table 2. Search queries evaluation.

Query	Retrieved results	Processing time (ms)	Query	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

6. Conclusions and Future Work

In this paper, we have illustrated 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, especially 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 future and to reach a large significant size of collected Arabic books similar to the size archived by Google for English and other languages, we should include more electronic versions of Arabic books.

For the many Arabic books that have no electronic versions we need to scan them suing OCR tools then index them as discussed in this research. More work to validate the new corpus and its performance is also needed.

References

[1] Al-Jedady A., Al-Kabi M., and Alsmadi I., "Fast Arabic Query Matching for Compressed Arabic Inverted Indices," *International Journal of Database Theory and Application*, vol. 5, no. 4, pp. 81-94, 2012.

[2] Baroni M. and Ueyama M., "Building General- and Special-Purpose Corpora by Web Crawling," *in Proceedings of the 13th NIJL International Symposium*, Tokyo, pp. 31-40, 2006.

[3] Bazzi I., Schwartz R., and Makhoul J., "An Omnifont Open-Vocabulary OCR System for English and Arabic," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 6, pp. 495-504, 1999.

[4] Bhowmick S., Chakraborty S., and Agrawal D., "Study of Hadoop-MapReduce on Google N-Gram Datasets," *in Proceedings of IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems*, Dallas, pp. 488-490, 2015.

[5] Birkenes M. and Johnsen L., "From Digital Library to N-Grams: NB N-gram," *in Proceedings of the 20th Nordic Conference of Computational Linguistics*, Vilnius, pp. 293-295, 2015.

[6] Coetzee D., "TinyLex: Static N-Gram Index Pruning with Perfect Recall," *in Proceedings of the in ACM 17th Conference on Information and Knowledge Management*, California, pp. 409-418, 2008.

[7] Demeyer S., Murgia A., Wyckmans K., and Lamkanfi A., "Happy birthday! A Trend Analysis on Past MSR Papers," *in Proceedings of the 10th Working Conference on Mining Software Repositories*, San Francisco, pp. 353-362, 2013.

[8] Elshafei M., Al-Muhtaseb H., and Al-Ghamdi M., "Machine Generation of Arabic Diacritical Marks," *in Proceedings of International Conference on Machine Learning; Models, Technologies and Applications*, LasVegas, pp. 128-133, 2006.

[9] Ernst C., *The Global Significance of Arabic Language and Literature. Religion Compass*, John Wiley and Sons Ltd, 2013.

[10] Evert S., "Google Web 1T 5-Grams Made Easy (but not for the Computer), " *in Proceedings of the NAACL HLT 6th Web as Corpus Workshop*, Los Angeles, pp. 32-40, 2010.

[11] Huston S., Moffat A., and Croft W., "Efficient Indexing of Repeated N-Grams," *in Proceedings of the 4th ACM International Conference on Web Search and Data Mining*, Hong Kong, pp. 127-136, 2011.

[12] Islam A. and Inkpen D., "Managing the Google Web 1T 5-Gram Data Set," *in Proceedings of International Conference on Natural Language Processing and Knowledge Engineering*, Dalian, pp. 1-5, 2009.

[13] Grabowski S. and Swacha J., "Google Books Ngrams Recompressed and Searchable," *Foundations of Computing and Decision Sciences*, vol. 37, no. 4, pp. 271-281, 2012.

[14] KACST, "King AbdulAziz City for Science and Technology Arabic Corpus," URL: www.kacstac.org.sa/. Last Visited, 2015.

[15] Kanungo T., Marton G., and Bulbul O., "OmniPage vs. Sakhr: Paired Model Evaluation of Two Arabic OCR Products," *in Proceedings of SPIE Conference on Document Recognition and Retrieval*, San Jose, pp. 109-120, 1999.

[16] Khemakhem M. and Belghith A., "Towards a distributed Arabic OCR Based on the DTW Algorithm: Performance Analysis," *The International Arab Journal of Information Technology*, vol. 6, no. 2, pp. 153-161, 2009.

[17] Letchford A., Preis T., and Moat H., "The

Advantage of Simple Paper Abstracts," *Journal of Informetrics*, vol. 10, no. 1, pp. 1-8, 2016.

[18] LinY., Michel J., Aiden E., Orwant J., Brockman W., and Petrov S., "Syntactic Annotations for the Google Books Ngram Corpus," *in Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, Jeju Island, pp. 169-174, 2012.

[19] Michel J., Shen Y., Aiden A., Veres A., Gray M., The Google Books Team, Pickett J., Hoiberg D., Clancy D., Norvig P., Orwant J., Pinker S., Nowak M., and Aiden E., "Quantitative Analysis of Culture Using Millions of Digitized Books," *Science*, vol. 331, no. 6014, pp. 176-182, 2011.

[20] Parkinson, D. "Arabic Corpus: Arabic Corpus Search Tool," URL: http://arabicorpus.byu.edu/. Last Visited, 2015.

[21] Pechenick E., Danforth C., and Dodds P., "Characterizing The Google Books Corpus: Strong Limits to Inferences of Socio-Cultural And Linguistic Evolution," *PloS One*, vol. 10, no. 10, pp. 1-24, 2015.

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

[23] Roivainen E., "Frequency of the use of English Personality Adjectives: Implications for Personality Theory," *Journal of Research in Personality*, vol. 47, no. 4, pp. 417-420, 2013.

[24] Roziewski S., Stokowiec W., and Sobkowicz A., *Machine Intelligence and Big Data in Industry*, Springer Cham, 2016.

[25] SDL, "Saudi Digital Library," Last Visited, 2015.

[26] Seeley Y., "Apache Solr," presentation, 29 June 2006, Dublin, Ireland, http://people.apache.org/~yonik/presentations/Solr.pdf, Last Visited, 2015.

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

[28] Touj S., Amara N., and Amiri H., "Arabic Handwritten Words Recognition Based on a Planar Hidden Markov Model," *The International Arab Journal of Information Technology*, vol. 2, no. 4, pp. 318-325, 2005.

[29] Tromp E. and Pechenizkiy M., "Graphbased N-Gram Language Identification on Short Texts," *in Proceedings of the 20th Machine Learning Conference*, Belgium, pp. 27-34, 2011.

[30] Wang K., Thrasher C., Viegas E., Li X., and Hsu B-j., "An Overview of Microsoft Web N-Gram Corpus and Applications," *in Proceedings of the NAACL HLT* 2010*: Demonstration Session*, Los Angeles, pp. 45-48, 2010.

[31] Zarour M. and Alsmadi I., "Incorporating Arabic Corpus into Google N-gram Viewer: Initial Results," *in Proceedings of the 5th International Conference on Arabic Language Processing*, Oujda, pp. 190-199, 2014.

[32] Zidouri A., Sarfraz M., Shahab S., and Jafri S., "Adaptive Dissection Based Subword Segmentation of Printed Arabic Text," *in Proceedings of the 9th International Conference on Information Visualisation*, London, pp. 239-243, 2005.

Izzat Alsmadi is an assistant professor in the department of computing and cyber security at Texas A&M, San Antonio. He obtained his Ph.D. degree in software engineering from NDSU (USA), his second master in software engineering from NDSU (USA) and his first master in CIS from University of Phoenix (USA). He had a B.sc degree in telecommunication engineering from Mutah university in Jordan. He has several published books, journals and conference articles largely in software engineering, security, data mining, IR and NLP.

Mohammad Zarour holds a Ph.D. in Software Engineering since June 2009 from École de Technologie Supérieure (ETS) – Université du Québec (Montréal, Canada). A master degree in Computer Science in 1998 from University of Jordan. He is currently an assistant professor at Prince Sultan University. He has over 6 years of experience in teaching in a university environment. Dr. Zarour worked as a chief Technical Advisor at one of the UNDP programmes in King AbdulAziz City of Science and Technology in Riyadh for 3 years. He also has several years of industry experience in information systems development and process improvement. His research interests include software process quality, software product quality, cost estimation, data mining and Arabic related natural language processing. He has more than 25 peer-reviewed publications.