Cross Language Information Retrieval Model For Discovering WSDL Documents Using Arabic Language Query

Prof. Dr.Torkey I.Sultan
Information Systems Department,
Faculty of Computers & Information
Helwan University,
Cairo,Egypt

Dr. Ayman E. Khedr
Information Systems Department,
Faculty of Computers & Information
Helwan University,
Cairo,Egypt

Fahad Kamal Alsheref
Management Information Systems
department, Modern Academy,
Cairo,Egypt

***Abstract*—Web service discovery is the process of finding a suitable Web service for a given user's query through analyzing the web service's WSDL content and finding the best match for the user's query. The service query should be written in the same language of the WSDL, for example English. Cross Language Information Retrieval techniques does not exist in the web service discovery process. The absence of CLIR methods limits the search language to the English language keywords only, which raises the following question "How do people that do not know the English Language find a web service, This paper proposes the application of CLIR techniques and IR methods to support Bilingual Web service discovery process the second language that proposed here is Arabic. Text mining techniques were applied on WSDL content and user's query to be ready for CLIR methods. The proposed model was tested on a curated catalogue of Life Science Web Services http://www.biocatalogue.org/ and used for solving the research problem with 99.87 % accuracy and 95.06 precision**

Keywords—Web services discovery; Cross Language Information Retrieval; WSDL; Text Mining

I. INTRODUCTION

Web service discovery aims at finding services whose description matches that of a desired service. The description of service contains a functional and a non-functional part. The former provides information about what the service does and how it works. This is basically expressed in terms of the required inputs and generated outputs, as well as any pre-conditions that need to be satisfied in order for the service to be executed and any effects that result from its execution. [9]

The discovery process is done through applying information retrieval techniques on the WSDL content of the web service. Large number of studies used various Information Retrieval (IR) techniques to search textual service metadata for service discovery. Wang and Stroulia [17] employed the inverted file [11] to index and search natural language description of desired services. Also Platzer and Dustdar [3] used the Vector Space Model (VSM) to implement a search engine for a Web service, and Sajjanhar et al. [1] have attempted to leverage Latent Semantic Analysis (LSA), a variant of VSM, for Web services discovery.

All of the above work index service metadata found either in a WSDL file (i.e., the <documentation/> tag) or from a Universal, Description, Discovery, and Integration (UDDI) registry entry. In both cases, service metadata written in English are manually created by service providers.

When user wants to search for a web service for a specific purpose he should write his query in English language because the service metadata is written in English language. The motivation for addressing this idea came from our experience in developing the web service and studying the Cross-Language Information Retrieval (CLIR) methods and data mining techniques. We deal with the web services as collections of documents that should be prepared for IR techniques, then we applied CLIR techniques to find the suitable service that matches the user query that written in other language. The process that is responsible for finding the suitable service is matchmaking process.it is the process of finding suitable services given by the providers for the service requests of consumers. The current service discovery mechanism of WSs is based on WSDL [8] and UDDI [2]. WSDL is an XML based language to describe properties of services that written in English language. UDDI is a registry where service providers can advertise their services and service consumers can search for services. The specific objective of our research is therefore to apply CLIR in the web services discovery; this is done by modifying the Match Maker process by adding CLIR components to support the Cross language web service discovery.

Applying this approach leads to enable different stakeholders to search for web services using their natural language, especially in the new operating systems like Android , Windows 8 metro applications and Apple IOS. The developers of applications that working on these platforms are interacting with web services to expand the application capabilities like connecting to database server, or generating reports from multiple database resources. Searching and finding such a service to perform the developer required function are the heart of our research, and as we mentioned before the search process was being performing by English language only, but in our proposed model we expanding the research process to let the service searchers to find their required service but with any language not only with English language. Here we proposed the Arabic language because in Arab countries we have a lot of good developers but without a good awareness with English language, so our proposed model

should support developers to find and understand the web service description file which written in other languages. [12].

The rest of this paper is organized as follows. Section 2 provides the technical background that is used in the rest of the paper. Section 3 provides the related works that used in our research. Section 4 defines our proposed model and the modifications that had been done on the original one. Section 5 studies our methods in a case study. Finally, Section 6 contains the conclusion and future work.

II. Background And Motivation

In this section we provide a succinct background of the used techniques in our research. These techniques are: 1) Cross Language Information Retrieval, 2) Inverted file indexes, 3) WSDL term extraction, 4) WSDL term tokenization. And 5) Bilingual dictionary.

A. CLIR

Cross-language information retrieval (CLIR) systems allow users to find documents written in different languages from that of their query. The goal of a CLIR system is to help searchers find documents that are written in languages that are different from the language in which their query is expressed. This can be done by constructing a mapping between the query and document languages, or by mapping both the query and document representations into some third feature space. The first approach is often referred to as ‘‘query translation’’ if done at query time and as ‘‘document translation’’ if done at indexing time, but in practice both approaches require that document-language evidence be used to compute query-language term weights that can then be combined as if the documents had been written in the query language.[13]

In all cases, however, a key element is the mechanism to map between languages. This translation knowledge can be encoded in different forms as a data structure of query and document-language term correspondences in a machine-readable dictionary or as an algorithm, such as a machine translation or machine transliteration system. Gina-Anne Levow (2004) proposed CLIR reference architecture that focuses on the query translation architecture that illustrates the full range of opportunities to improve retrieval effectiveness. Fig. 1 present the Gina CLIR model that illustrates the data flow between the key components in her reference architecture. The dictionary based query translation architecture consists of two streams of processing, for the query and documents. Specifically, she exploits methods for suitable term extraction and pseudo-relevance feedback expansion at three main points in the retrieval architecture: before document indexing, before query translation, and after query translation. [10]

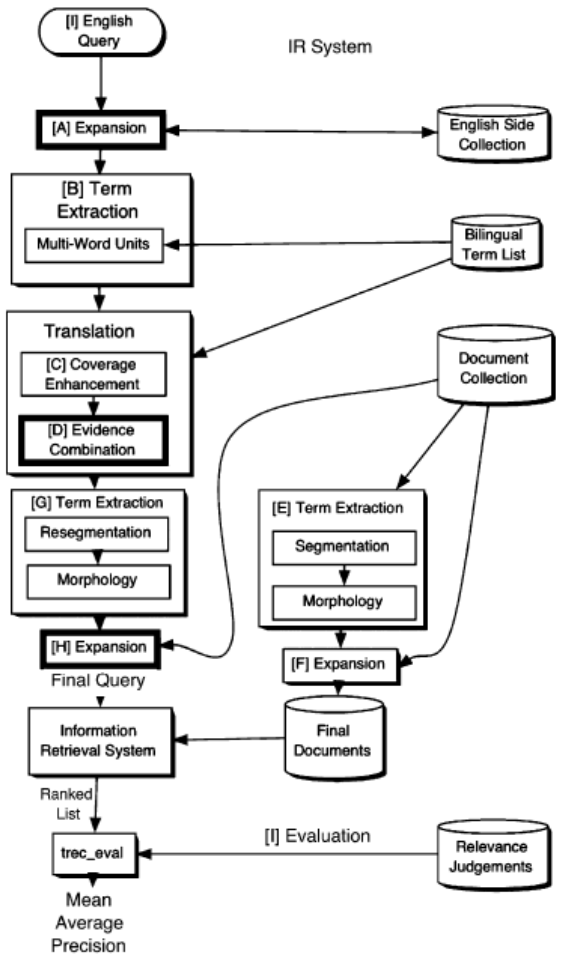


Fig. 1. CLIR architecture [10]

The previous architecture would be modified to accept queries written with other language like Arabic Indian etc. and the bilingual dictionary would be modified based the required languages for example Arabic vs. English Term list. The previous steps will be modified and explained in details in section 4.

B. Inverted file indexes

Inverted file is widely used for indexing text database. To support efficient information retrieval, the words of interest in the text are sorted alphabetically. For each word, the inverted file records a list of identifiers of the documents containing that particular term. Consider a sample text database consists of five documents. The indexer parses these five documents, and produces a set of distinct words for constructing the inverted file. The inverted file has two components a vocabulary and a set of inverted lists. The vocabulary comprises a collection of distinct words extracted from the text database. For each word t, the vocabulary also records: (1) the number (ft) of documents that contain t, and (2) the pointer to the corresponding inverted list. [15]

Each one of the word-specific inverted lists records: (1) a sequence of documents that contain t (notice that each document is represented as a document number d), and (2) for

each document d, the frequency (fd,t) of t appearing in d. Thus, the inverted list is a list of < d, fd,t > pairs.[15]

Applying the inverted file in WSDL discovery process will be done through dealing with WSDL files as a documents that containing the terms that needed to calculate the (fd,t) pairs the frequency (fd,t) of t appearing in WSDL document.

C. WSDL term extraction

WSDL document is an XML document validated against the WSDL schema. A lot of studies [16] in the XML retrieval literature suggested extract content from XML markups before applying IR techniques. Because the name attribute of an XML element represents the semantics for that particular element. Therefore, the value (i.e. nmtoken) of attribute name depicted in Fig. 2 is a good candidate for content extraction. [5]

Fig 3 presents a WSDL document instance. The value (e.g. GetLastTradePrice) of the name attribute for element <operation /> carries useful information implying the purpose of this operation at the lexical level. Similarly, values (i.e. nmtoken) of attributes ‘‘name’’ are extracted from a number of important WSDL elements (marked as bold faces in Fig. 3) definitions, message, part, portType, operation, input, output, service, and port. The value (i.e. qname) of attribute element for element part is also extracted for capturing the data structure of the parameters sent to/from the service operations. This forms recursive extraction of underlying data types for this element and/or type. In this example, the value (i.e. body) of attribute name for element part gives little useful information representing the real meaning of the input message part. Nevertheless, values (i.e. TradePriceRequest and TradePrice) of attribute element provide very valuable data in understanding the meaning of two message parts. The data structure within the WSDL element types and schema reveals more important lexical information about these two message parts through extracting the value of attribute name for element. Thus, in the case of TradePriceRequest, the element value is tickerSymbol. For TradePrice, price is extracted. Thus, by exploring solely lexical information, one can speculate that thisWebservice takes as input the stock ‘‘ticker symbol’’, and returns as output the ‘‘price’’ of the corresponding stock. The related operation name GetLastTradePrice also supports this proposition.[5]

The <documentation/> elements contain natural language information that could be used for constructing the text database.

```
<wsdl:definitions name="nmtoken"? targetNamespace="uri"?>
  <import namespace="uri" location="uri"/>*
  <wsdl:documentation .... /> ?
  <wsdl:types> ?
    <wsdl:documentation .... />?
    <xsd:schema .... />*
  </wsdl:types>
  <wsdl:message name="nmtoken"> *
    <wsdl:documentation .... />?
    <part name="nmtoken" element="qname"? type="qname"?/> *
  </wsdl:message>
  <wsdl:portType name="nmtoken">*
    <wsdl:operation name="nmtoken">*
      <wsdl:input name="nmtoken"? message="qname">?
      </wsdl:input>
      <wsdl:output name="nmtoken"? message="qname">?
      </wsdl:output>
      <wsdl:fault name="nmtoken" message="qname"> *
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="nmtoken" type="qname">*
    <wsdl:documentation .... />?
    <!– more elements omitted here –> *
  </wsdl:binding>
  <wsdl:service name="nmtoken"> *
    <wsdl:documentation .... />?
    <wsdl:port name="nmtoken" binding="qname"> *
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Fig. 2. WSDL document structure [9][5].

However, the reliability of comments or documentation written by humans is concerning as they may be either misleading or obsolete from the actual WSDL interface. So the WSDL parser is responsible for extracting the textual content directly from the interface definition to support information retrieval tasks.

D. WSDL term tokenization

An example of WSDL document in fig 3 according to the WSDL document structure in fig 2.This document uses untokenized words and sentences to define WSDL element attributes such as names (bold face fonts). This is due in part to the rules defined in the WSDL standard: all WSDL element names are associated with the W3C Schema attribute data type ‘NMTOKEN’, which is a mixture of name characters (including letters, digits, combining chars, etc.) but excludes the single white space (#x20) [5][4]. Furthermore, in practice, most WSDL documents are not created directly by humans but are automatically converted from other high level programming languages (e.g. Java, C#, etc.), in which white spaces are prohibited in variable names. Consider the operation

name GetLastTradePrice in Fig. 5. Under the normal text operation, the inverted file will create a new entry for the single word GetLastTradePrice in the vocabulary. [5]

As a result, the inverted file would be something looks like in Fig. 4, in which a partial vocabulary and its associated document frequencies ft and inverted lists (assuming the document id for this WSDL file is '1') are presented.[5]

```
……
<types>
    <schema targetNamespace="http://example.com/stockquote.xsd"
        xmlns="http://www.w3.org/2000/10/XMLSchema">
      <element name="TradePriceRequest">
        <complexType>
          <all>
            <element name="tickerSymbol" type="string"/>
          </all>
        </complexType>
      </element>
      <element name="TradePrice">
        <complexType>
          <all>
            <element name="price" type="float"/>
          </all>
        </complexType>
      </element>
    </schema>
  </types>
  <message name="GetLastTradePriceInput">
    <part name="body" element="xsd1:TradePriceRequest"/>
  </message>
  <message name="GetLastTradePriceOutput">
    <part name="body" element="xsd1:TradePrice"/>

  </message>
……
<portType name="StockQuotePortType">
    <operation name="GetLastTradePrice">
      <input message="tns:GetLastTradePriceInput"/>
      <output message="tns:GetLastTradePriceOutput"/>
    </operation>
</portType>
……
```

Fig. 3. A sample of a partial WSDL document, source [9][5].

Vocabulary	f_t	*Inverted lists*
…	…	…
body	4	< 1, 2 >, … , … , …
…	…	…
getlasttradeprice	1	< 1, 1 >, …
…	…	…
stockquoteporttype	1	< 1, 1 >, …
…	…	…
tradeprice	3	< 1, 2 >, … , …
tradepricerequest	2	< 1, 2 >, …
…	…	…

Fig. 4. A sample of a partial WSDL document, source [9][5].

This should affect the IR process so the operation name has to be tokenized into four separate terms "GetLastTradePrice" to Get, Last, Trade, and Price. One may argue that a trivial string pattern discovery (e.g. letter case patterns) would easily tokenize them into meaningful terms. This Tokenization problem was solved and the solution is presented in the related work section.

E. Bilingual dictionary

A bilingual dictionary or translation dictionary is a specialized dictionary used to translate words or phrases from one language to another. Bilingual dictionaries can be unidirectional, meaning that they list the meanings of words of one language in another, or can be bidirectional [disambiguation needed], allowing translation to and from both languages. Bidirectional bilingual dictionaries usually consist of two sections, each listing words and phrases of one language alphabetically along with their translation. In addition to the translation, a bilingual dictionary usually indicates the part of speech, gender, verb type, declension model and other grammatical clues to help a non-native speaker use the word. Other features sometimes present in bilingual dictionaries are lists of phrases, usage and style guides, verb tables, maps and grammar references. In contrast to the bilingual dictionary, a monolingual dictionary defines words and phrases instead of translating them. [6]

Bilingual term lists are extensively used as a resource for dictionary-based Cross-Language Information Retrieval (CLIR), in which the goal is to find documents written in one natural language based on queries that are expressed in another.

The translation component of dictionary-based CLIR techniques depend on a successful cascade of three processes: (1) selection of the terms to be translated, (2) generation of a set of candidate translations, and (3) use of that set of candidate translations in the retrieval process. For the first stage, the best results are typically obtained by translating multiword expressions when possible, backing off to individual words when necessary, and further backing off to morphological roots when the surface form cannot be found [6].

In the second stage, algorithms for choosing among alternative translations have been extensively studied, and older techniques based on averaging weights computed for each translation can benefit significantly from translation selection based on term co-occurrence within the target corpora. The focus on the third stage has been somewhat more recent, with the best presently known technique based on accumulating term frequency and document frequency evidence separately in the document language, then combining that evidence to create query-language term weights [6].

III. Related Work

A. The IR-style Web Services Discovery

Numerous recent efforts have been reported in applying IR for Web services discovery. Here we proposed Chen's IR-style web service discovery that uses inverted file indexes for Web services discovery that will be merged in CLIR methods in our proposed approach.

Chen Wu proposed IR-style Web services discovery approach that was illustrated in Fig. 5, in which term tokenization constitutes an important step for WSDL term processing. Initially, service providers deploy their Web services accessible to the public via the Web. In doing so, they also publish a service description, i.e., the WSDL documents, which captures the functional capabilities and technical details (e.g., transport bindings) of a Web service.

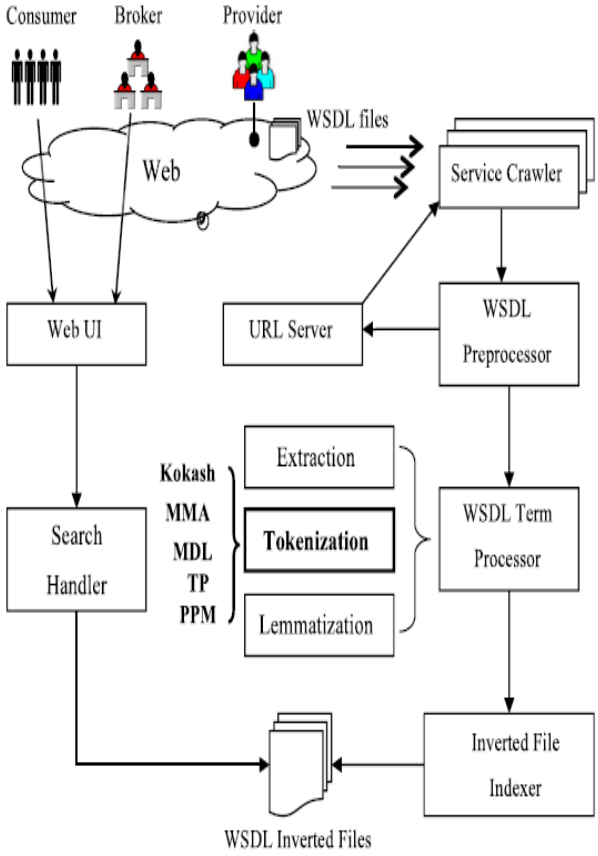


Fig. 5. IR-style service discovery approach. [5]

These service descriptions can be collected by a number of Service Crawlers, which fetch WSDL files from the Internet. Alternatively, they can also be collected from some well-known service datasets such as one of the WS curated catalogue .Crawlers hand over retrieved WSDL files and associated HTML files to the WSDL Preprocessor for link analysis. This yields a list of new URLs that may point to some new WSDL files. These URLs are assigned to an idle crawler by the URL server such as Pica-Pica Web Service Description Crawler. [5]

All retrieved WSDL files are then passed to the WSDL Term Processor, which (1) parses WSDL files and extracts important data (e.g., operations, messages, data types, etc.), (2) tokenizes extracted content into separate terms (the focus of this paper), and (3) carries out other linguistic tasks such as lemmatization and stop-word elimination, etc. Five tokenization methods (two baselines – Kokash and MMA – and three statistical methods MDL, TP, and PPM) are used in (2).

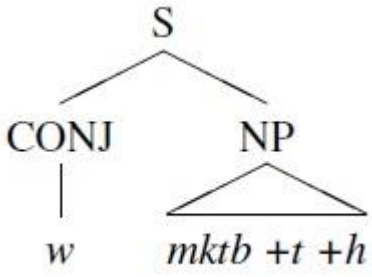
The WSDL Processor generates the 'term document', which contains separated words in a flat structure. The term document is transferred to the Inverted File Indexer. The indexer takes as inputs tokenized and lemmatized terms with their associated occurrences information in each document and generates as outputs the compiled data arrangement with pre-aggregated information optimized for fast searching. The data structure of inverted index is consistent with the notion of term-document matrix, which consists of term vectors as matrix rows and document vectors as matrix columns. The term vector is sorted that allows fast lookup operation. After finishing the document tokenization and indexing processes, the search handler component will extract the key terms from the query then it will search via extracted terms in the inverted file index , after that the most matched WSDL documents is returned based on terms frequency in the WSDL documents [5].

But Chen's model is based on the document retrieval where the query and the documents are written in the same language ,but if the query was written in different language it will not retrieve the documents or the services, for this limitation the need for applying the CLIR techniques was arising to support searching in documents with queries with different languages, so we modified Chen's model to accept queries with different languages "Arabic in our model" and retrieve the suitable service and WSDL document with a translated WSDL version to query writer's language.

B. Arabic Treebank (ATB) segmentation

The Arabic language has a very rich morphology where a word is composed of zero or more prefixes, a stem and zero or more suffixes. This makes Arabic data sparse compared to other languages, such as English, and consequently word segmentation becomes very important for many Natural Language Processing tasks that deal with the Arabic language

The ATB is used in most of Arabic language segmentation cases, this is a light segmentation adopted to build parse trees in the Arabic TreeBank (ATB) corpus. This type of segmentation considers splitting the word into affixes if and only if it projects an independent phrasal constituent in the parse tree. As an example, in the word ومكتبته (wmktbth — and his library) mentioned earlier, the phrasal independent constituents are: (i) conjunction و (w — and); (ii) nounand the head of a Noun Phrase (NP) مكتبة (mktbt library); and (iii) a pronoun (PRON) ه (h—his). This would lead to the following parse tree: [14]



A full segmentation (i.e., morphological segmentation) will separate the suffix ة(t feminine marker) from the word **مكتبة** (mktbt -library).Since the ة (and generally all the suffixes which are gender marks) are not independent constituents as shown in the previous parse tree, they are not considered for ATB segmentation. Thus, the ATB segmentation scheme considers splitting only a subset of prefixes and suffixes from the stem. When using ATB segmentation, the number of words is similar to its counterpart in English. This is one reason why

ATB segmentation is widely used in building machine translation systems for the English-Arabic language pair. For the word **ومكتبته** (wmktbth - and his library), the ATB segmentation would be **ه+ مكتبت + و** (w+ mktbt +h). Prefixes that are considered for possible segmentation are:

1. "**ل**", (l — to);
2. "**ب**" (b — in);
3. "**و**" (w —and); and
4. "**ك**" (k — as).

Possible segmented suffixes are the possessive personal pronouns such as:

1. "**ي**" (y— my);
2. "**هم**" (hm — their);

*F. "**كم**" (km— yours); etc.[14]*

C. Approaches to CLIR

The current CLIR research into three categories [7]:

- Document Translation.

Document translation comprises approaches to CLIR which require that all documents in the collection be translated into the language of the original user request. User requests or derived queries are then dealt with by a monolingual IR system.

The principal translation method reported in the literature is commercial off the shelf MT. The rationale behind this approach is that whereas user requests are often viewed as being too short to provide sufficient contextual information for traditional MT to perform well, documents may be translated as normal texts. This approach may be implemented without developing any CLIR-specific software. Nothing is needed other than a commercial MT product and a standard retrieval engine. Problems such as translation ambiguity and coverage are dealt with in a "black box" manner by the MT software.

There are several problems with document translation for CLIR. The first is that the cost in terms of time and money of translating a large document collection using traditional MT technology can be prohibitive. It took Oard and Dorr two months to translate the 550MB TREC German collection.

- Non-Translation Based Methods.

There is a small number of approaches to CLIR which translate neither the requests/queries nor the documents, opting instead to convert both to a language-independent representation where they can be searched directly .The only such system to be entered in a large-scale evaluation such as TREC was the CINDOR system at Textwise Corporation which used Wordnet synsets as a multilingual thesaurus to mediate between requests and documents. However, despite the existence of projects like EuroWordNet which aim to translate Wordnet into languages other than English by hand, Wordnet is still limited in its coverage, and it is difficult to see how it could be expanded without considerable work. Considerable improvements in performance were recorded at TREC-8 for the CINDOR system by switching to using MT software for request translation.

- Request or Query Translation

We have seen that it is not usually feasible to translate each document in the collection into every language represented in it, and that existing techniques which map both documents and queries or requests to an Interlingua representation require as much hand-crafted knowledge as document MT but do not perform as well. In this section we examine the obvious alternative - translating the requests or queries into the languages of the document collection. There are three main query translation methods:

- Request MT. This is where a commercially-available MT engine is used as a "black box" to translate the user request as-is.
- Corpus-Based Query Translation. This is where techniques from the domain of corpus linguistics are applied to map the terms in the bag of words query derived from the user request to a semantically equivalent representation in the target language.
- Dictionary-based Query Translation. This is where a simple machine-readable bilingual dictionary is employed to map the terms in the bag of words query to an equivalent representation in the target language.

IV. ARABIC CLIR WEB SERVICE DISCOVERY MODEL

A. The Arabic CLIR Web services discovery approach is illustrated in fig 6.

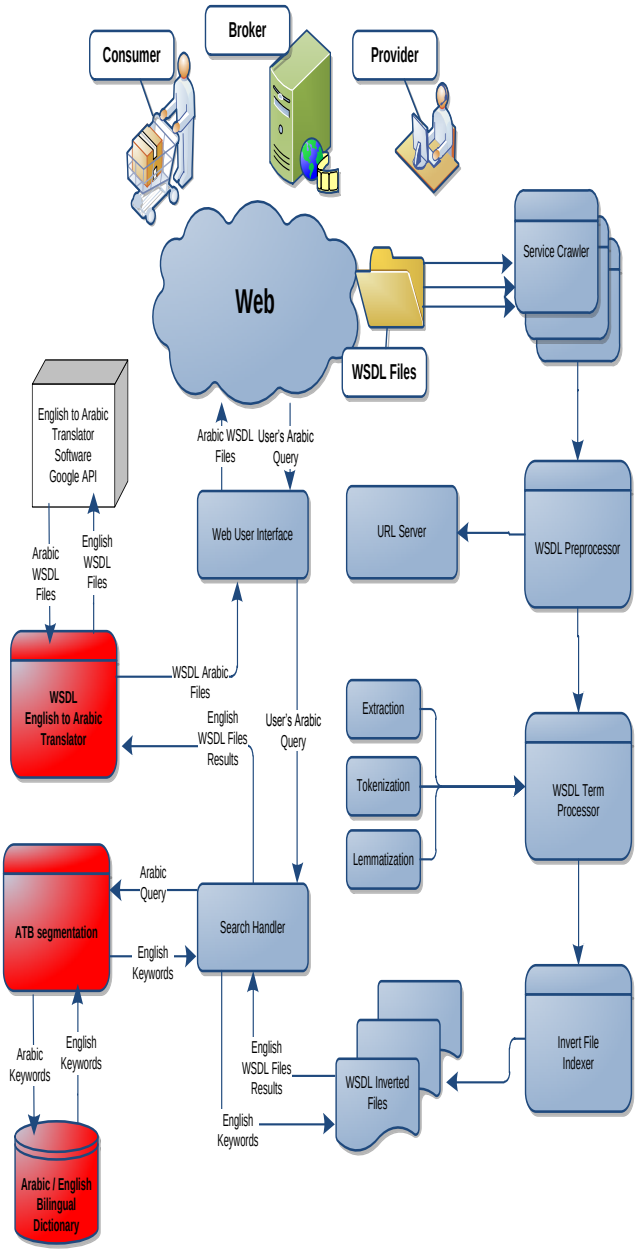


Fig. 6. Cross Language Information Retrieval service discovery approach.

As we mentioned in the IR-web service discovery in the related work part, the Service Crawler is responsible for collecting the WSDL document for the services, each service has its own WSDL file that was written by the service provider.

The WSDL preprocessor and WSDL are responsible for preparing the WSDL files and extract nmtoken values that mentioned in the WSDL tokenization part in this paper. All WSDL files are transferred to WSDL inverted files by extracting the WSDL nmtokens values and put them into the same form as fig4.

Our contribution is presented in adding the CLIR components to IR-web service discovery approach. This components will expand the WS discovery to other languages like Arabic and etc.in our approach we are working in Arabic language only.

B. CLIR Components

The user will send his query through the web user interface to the search handler component, this query may be written in English language or in Arabic language, and if the query was written in English then the search handler will search inside WSDL inverted files without any change in the IR-web service discovery.

But if user writes his query in Arabic then the search handler will use our CLIR components the detailed in the following sections.

C. ATB Segmentation

We chose the Query Translation approach to apply CLIR. The query translation approach is cost efficient especially if it was compared with the translation of each document in the collection into every language represented in it.

The first step in the Arabic Query Translator is query segmentation. We used The ATB segmentation algorithm for divide the user's Arabic query into several tokens .as we mentioned before number of Arabic tokens is similar to the English one which makes the translation process is easier. The following example shows the Arabic query as an input and the segmented Arabic tokens as an output of the process.

Arabic query:

تحويل من PDF الى نص

After applying the ATB:

(S (NP <tahweel **تحويل**
(PP <min **من**
(NP< PDF **PDF**
(PP <ilaY **إلى**
(NP< nas **نص**
)))))

After applying the ATB segmentation algorithm the output is a several tokens which each one could be an entry in bilingual dictionary in the next step.

D. Arabic / English Bilingual Dictionary

As we mentioned before the bilingual dictionary is a specialized dictionary used to translate words or phrases from one language to another. For this purpose we searched for Arabic / English open source dictionary. We found ARABEYES, it is a Meta project that is aimed at fully supporting the Arabic. It is designed to be a central location to standardize the Arabization process.

The extracted tokens from the ATB are sent it to the ARABEYES to get the translated keywords as shown in the following example:

Arabic Tokens:

- تحويل
- من
- **PDF**
- إلى
- نص

ARABEYES output:

- **Conversion** تحويل
- **From** من
- **PDF** **PDF**
- **To** إلى
- **Text** نص

The Translated keywords are sent to the search handler components. Search handler components are using the inverted file techniques to find the WSDL documents that matches the user's query keywords. Each one of the translated keywords is compared to inverted lists records where each WSDL documents that contain t (notice that each document is represented as pairs < d, fd,t > d is a document number and the frequency (fd,t) of t appeared in d. then the arranged WSDL document list according the term frequency in the document is returned to the search handler component.

E. WSDL English to Arabic Translator

The search handler component gets the WSDL Documents that matching the user's query. All these documents are written in English language. These document should be translated to Arabic, before the translation process the information inside the WSDL document should be extracted. As we mentioned in section 2.3 the important information is exists in the attribute names that contain a semantic information about the selected web services and its values contain the information that is related to the previous attributes, the other important part in the WSDL document is the <documentation> element. This part is written in natural language as a description for the web service, which describe the purpose of the service and other information provided by the service provider, the following example in fig7 shows some of this information:

Attributes and its content:

```
.............
<service name="PdfToTextService">
  <port name="PdfToTextPort" binding="tns:PdfToTextPortBinding">
 <WSDL:address location="http://gnode1.mib.man.ac.uk:8080/
  FullTextWebServices/PdfToTextService"></soap:address>
  </port>
</service>
<documentation>
  This service will extract the text content from a PDF file.
  It uses the pdftotext executable from Xpdf
  (http://www.foolabs.com/xpdf/).
  The text returned from this service
  often contains characters which are XML-invalid,
  therefore the text is returned in i
</documentation>
<portType name="PdfToText">
<operation name="pdfToTextBase64">
<input message="tns:pdfToTextBase64"></input>
<output message="tns:pdfToTextBase64Response"></output>
</operation>
<operation name="pdfToText">
<input message="tns:pdfToText"></input>
<output message="tns:pdfToTextResponse"></output>
</operation>
</portType>
.................................
```

Fig. 7. Example of WSDL document returned form the search handler.

The extracted WSDL documents are sent to the WSDL English to Arabic Translator component. The attributes names and its values are translated to Arabic. For this purpose they will be sent to WSDL translator component as a pairs like in the following example:

< (Attribute Name) service name=
(Attribute Value)"PdfToTextService">

And the output will be like:

<اسم الخدمه="نص الى pdf ">

After translating all WSDL attributes and their values the WSDL translating components will collect all translating process output and put them in the form like figure 8.

The two WSDL documents the English and the translated are sent to the user or to the service requester, which will give him a better understanding of the WSDL documents, and let him to decide which service is the best match for his query.

```
.....
">pdf الى نمى "=اسم الخدمه>
">  المنفذ الملزم الى نمى:tns"=الملزم " pdf الى نمى "=اسم المنفذ>
<WSDL:عنوان المكان="http://gnode1.mib.man.ac.uk:8080/
FullTextWebServices/PdfToTextService"></soap:العنوان>
<المنفذ/>
<الخدمه/>
<التوثيق>
وهذه الخدمة استخراج مضمون النص من ملف .PDF فإنه يستخدم PDF لتنفيذ النص منPDF
(http://www.foolabs.com/xpdf/).
عاد النص من هذه الخدمة
غالبا ما تحتوي على أحرف والتي هي غير صالحة-XML ،
ولذلك يتم إرجاع النص
< التوثيق/>
">pdf الى نمى "=اسم نوع المنفذ>
"64>pdf الى نص"=اسم العملية>
<المدخلات/>"64>pdf الى نص :tns"=مدخلات الرساله>
<المخرجات/>"استجابة 64 pdf الى نص :tns"=مخرجات الرساله>
<العمليه/>
">pdf الى نص"=اسم العملية>
<المدخلات/>" pdf الى نص :tns"=مدخلات الرساله>
<المخرجات/>"استجابة pdf الى نص :tns"=مخرجات الرساله>
<العمليه/>
نوع المنفذ/>
<binding name="PdfToTextPortBinding" type="tns:PdfToText">
<operation name="pdfToTextBase64">
.....
```

Fig. 8. Translated WSDL document

G. English To Arabic Translator Software API

As we mentioned in section 4.1.3 the retrieved WSDL documents were translated to Arabic. This process is done through a specialized software that we mentioned it in figure 6 as a black box.in this part we used a readymade software because the WSDL document contains some attributes an elements that contain a long description of the service which like:

```
<documentation>
  This service will extract the text content from a PDF file.
  It uses the pdftotext executable from Xpdf
  (http://www.foolabs.com/xpdf/).
  The text returned from this service
  often contains characters which are XML-invalid,
  therefore the text is returned in i
</documentation>
```

That needs a full translation with applying the grammars rules and machine learning techniques. For this purpose we used the Google translation APIs. Google Translate is a free statistical multilingual machine-translation service provided by Google Inc. to translate written text from one language into another. Google API client libraries are available in a number of popular programming languages like .NET and Java. The API provides access to a service and provides a single URI that acts as the service endpoint.

The following figure show a sample of the programming code using c# for accessing the Google API, the following functions has two input 1- the WSDL attribute and its value as a pair ,2- the language pair which in our case is English to Arabic. And the output is the translated text:

```
public string TranslateText(string input, string languagePair)
{
 string url = String.Format("http://www.google.com/translate_t?hl=en&ie=UTF8&text={0}&langpair={1}"
, input, languagePair);
 WebClient webClient = new WebClient();
 webClient.Encoding = System.Text.Encoding.UTF8;
 string result = webClient.DownloadString(url);
 result = result.Substring(result.IndexOf("<span title=\"") + "<span title=\"".Length);
 result = result.Substring(result.IndexOf(">") + 1);
 result = result.Substring(0, result.IndexOf("</span>"));
 return result.Trim();
}
```

Fig. 9. C# code for accessing Google API

V. Evaluation

For testing our proposed approach we searched for a registry of curated Web Services that contains a huge numbers of Web services. And we found the BioCatalogue, it is a centralized registry of curated Life Science Web Services and it has the following functions:

1) Search (by Keywords):
User or agent could search using any part of service name.
2) Search (by Data):
User or agent could search using a specific input data or output data.
3) View Full Details of a Service:
Also they could view the full details of a service.

The previous functions are available on the BioCatalogue.com website through a defined web interface, also there are an available API's to access all BioCatalogue.com searching, filtering, browsing data and WSDL documents. The BioCatalogue provides a set of public RESTful endpoints that allow user and agent to query the registry programmatically and integrate the data and functionality into their own scripts, workflows, apps, tools and mashups.

We selected a 20 English queries for a web services, we chose them based on BioCatalogue Latest Activity log, which contains the recent used web services based on users' queries. The 20 queries and their returned results were listed in the Table 1.

TABLE I. English Queries and Their results

No	English Query	No WS
1	Convert PDF to Text	13
2	Cleaning Text from unwanted classes	3
3	Text search and retrieval from large databanks	3
4	Protein Sequence Analysis on pfam	12
5	Named Entity Recognition	16

6	Biomedical Named Entity Recognizer	2
7	Document Discovery	3
8	Document Similarity	1
9	Document Clustering	1
10	Classify text	1
11	Text Mining	38
12	Sentence split	2
13	Structure Retrieval	70
14	Similarity sequence databases	1
15	Chemical text mining	1
16	Image Retrieve	5
17	Genome Information Broker	6
18	Protein sequence database query	56
19	Protein Sequence Similarity service	4
20	Protein Tertiary Structure	15

The selected English queries were translated manually to Arabic language and entered as input of our proposed model and their results were tested against the original English queries results. The Arabic queries and their correct and incorrect returned results numbers were listed in the Table 2, for each query we calculated the accuracy and precision by using the binary classification like:

	True	False
Positive	True positive	False positive
Negative	False negative	True negative

That is, the accuracy is the proportion of true results (both true positives and true negatives) in the population.

$$\text{accuracy} = \frac{\text{number of true positives} + \text{number of true negatives}}{\text{number of true positives} + \text{false positives} + \text{false negatives} + \text{true negativ}}$$

On the other hand, precision or positive predictive value is defined as the proportion of the true positives against all the positive results (both true positives and false positives)

$$\text{precision} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{false positive}}$$

Note: The total number of available Services for searching is **2485 services**

We supposed that the returned number of services in English language is the ideal case which represents the total number of services that query written in Arabic language should return, which exists in the third column in table 2.

Then we calculated the average accuracy and average precision for the 20Arabic queries:

AvgAccuracy = (Total accuracy/No of queries) =99.87 %

AvgPrecision = (Total precision /No of queries) =95.07%

The WSDL documents for the selected web services are translated as we mentioned before in 4.1.3 and figure 8.

And the following chart shows the similarity between the results of English query and Arabic Query:

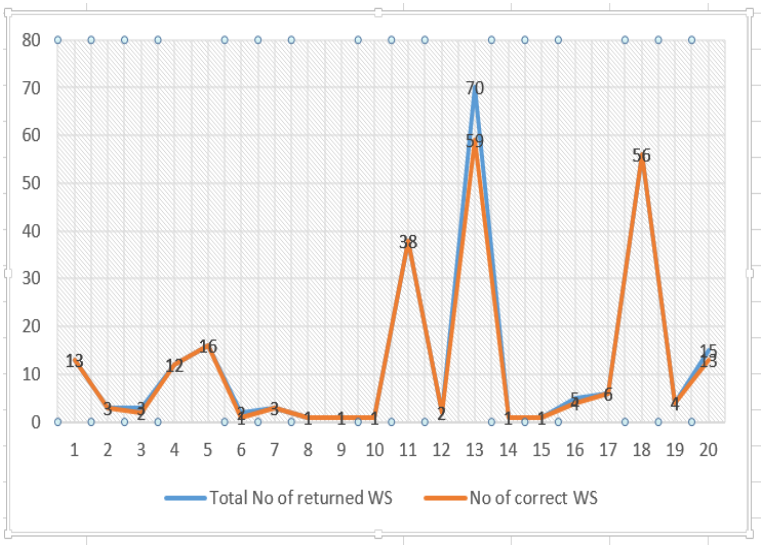


Fig. 10. English vs. Arabic Queries results

TABLE II. Arabic Oueries and Their results

No	Arabic Query	Number of returned WS from English Query	Total No of returned WS	No of True Positive	No of True Negative	No of False Negative	No of False Positive	**Accuracy %**	**Precision %**
1	تحويل PDF إلى نص	13	13	13	2472	0	0	100.00	100.00
2	تنظيف النص من الطبقات غير المرغوب فيها	3	3	3	2482	0	0	100.00	100.00
3	البحث عن النص واسترجاع من قواعد البيانات الكبير	3	2	2	2482	1	0	99.96	100.00
4	تحليل تسلسل البروتين على pfam	12	12	12	2473	0	0	100.00	100.00
5	التعرف على الكيان المسمى	16	16	16	2469	0	0	100.00	100.00
6	التعرف على الطب الحيوي للكيان المسمى	2	1	1	2483	1	0	99.96	100.00
7	اكتشاف الوثيقه	3	7	2	2478	1	4	99.80	33.33
8	تشابه الوثيقه	1	1	1	2484	0	0	100.00	100.00
9	تجميع الوثيقه	1	1	1	2484	0	0	100.00	100.00
10	تصنيف النص	1	4	1	2481	3	0	99.88	100.00
11	التعدين النص	38	38	38	2447	0	0	100.00	100.00
12	انقسام الجملة	2	2	2	2483	0	0	100.00	100.00
13	هيكل استرجاع	70	59	40	2396	30	19	98.03	67.80
14	قواعد البيانات تشابه تسلسل	1	1	1	2484	0	0	100.00	100.00
15	النص الكيميائية التعدين	1	1	1	2484	0	0	100.00	100.00
16	استرجاع الصور	5	4	4	2480	1	0	99.96	100.00
17	سمسار معلومات الجينوم	6	9	6	2476	3	0	99.88	100.00
18	الاستعلام عن البروتين تسلسل قاعدة البيانات	56	56	56	2429	0	0	100.00	100.00
19	خدمه تسلسل التشابه البروتين	4	4	4	2481	0	0	100.00	100.00
20	بنية البروتين العالي	15	13	13	2470	2	0	99.92	100.00

VI. EVALUATION

Web service discovery is a very important process special after applied IR techniques which leads to IR web service discovery approach, but this limits it use to English language users only, our approach leads to expand the web service users to other languages, in our research we proposed the Arabic language approach which could be used in other languages like Indian, Chinese and so on. This may lead to further applications that could use multiple language web service and application to application different language data exchange.

Our future work to expand our approach to use the data mining techniques on the web service user's history to improve the selection mechanism.

REFERENCES

[1] A. Sajjanhar, J. Hou, Y. Zhang, Algorithm for web services matching, in: APWeb, 2004, pp. 665–670.

[2] A. Wombacher, P. Fankhauser, B. Mahleko, E. Neuhold, Matchmaking for business processes based on conjunctive finite state automata, International Journal of Business Process Integration and Management 1 (1) (2005) 3–11.

[3] C. Platzer, S. Dustdar, A vector space search engine for Web services, in: Third IEEE European Conference on Web Services, Sweden, 2005.

[4] C. Van Rijsbergen, Information Retrieval, Buttersworth, London, 1979.

[5] Chen Wu,WSDL term tokenization methods for IR-style Web services discovery,Science of Computer Programming,2011.

[6] Dina Demner-Fushman,Douglas W. Oard,The Effect of Bilingual Term List Size on Dictionary-Based Cross-Language Information Retrieval,2003.

[7] Donnla Nic Gearailt,Dictionary characteristics in cross-language information retrieval,2005.

[8] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, Web services description language (WSDL) 1.1, W3C, http://www.w3.org/TR/2001/NOTEwsdl20010315/ (March 2001).

[9] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, Web Services Description Language (WSDL) 1.1., 2001, 18/05/2007. Available: http://www.w3.org/TR/wsdl.

[10] Gina-Anne Levow , Douglas W. Oard , Philip Resnik ,Dictionary-based techniques for cross-language information retrieval,Information Processing and Management,2005.

[11] J. Zobel, A. Moffat, Inverted files for text search engines, ACM Computing Surveys 38 (2006) 1–55.

[12] Jianguo Lu, Yijun Yu,Web Service Search: Who, When, What, and How,Proceedings of the 2007 international conference,2007.

[13] Levow G-A, Oard D, Resnik P,Dictionary-based techniques for cross-language information retrieval,Information Processing & Management,2005.

[14] Mohamed Maamouri, Ann Bies, Tim Buckwalter, Wigdan Mekki,The Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus,2004.

[15] R. Baeza-Yates, B. Ribeiro-Neto, Modern Information Retrieval, Addison Wesley, 1999.

[16] R. Baeza-Yates, N. Fuhr, Y.S. Maarek (Eds.), Proceedings of the SIGIR Workshop on XML and Information Retrieval, 2002.

[17] Y. Wang, E. Stroulia, Flexible interface matching for web-service discovery, in: Fourth International Conference on Web Information Systems Engineering, 2003.