Information Extraction from Arabic Law Documents

Samah Abu Shamma, Aseel Ayasa, Wala' Sleem and Adnan Yahya
Department of Electrical and Computer Engineering,
Birzeit University,
Birzeit, Palestine
yahya@birzeit.edu

***Abstract*- Information hidden in unstructured or semi-structured law documents can be very useful but may not be readily accessible. To get this information, an information extraction (IE) system is needed. Making extracted information available in structured form enables answering complex queries that may go well beyond simple keyword search and thus may be of interest to law professionals. In this paper we address the issue of Arabic information extraction from law documents. We describe a system we developed to extract important information, that may be of interest to potential users of these documents, with minimal human intervention. We employs a hybrid approach that utilizes machine learning and rule-based methods and Arabic NLP to facilitate the extraction of needed information. The approach was applied to a limited class of Arabic law documents and we are working on extending it to other document types and to other fields.**

Keywords-Information Extraction, IE, Arabic IE, Named Entity Recognition, Arabic NER, Arabic NLP.

I. INTRODUCTION AND BACKGROUND:

A. Motivation

The last years witnessed a sustained rise in the volume of textual information available online, including in Arabic. This spans a spectrum of online news, government documents, corporate reports, legal acts, medical alerts and social media communication, which are mainly transmitted as free text and thus are hard to search and digest. This led to a growing need for efficient and effective systems to analyze free-text data and obtain the hidden valuable knowledge through Information Extraction (IE).

The growing interest in Information Extraction is due to its high utility and conceptual simplicity. However, IE has challenges that make it an exciting research area[5,8,12]. IE-based systems are being increasingly deployed in the financial, medical, legal, and security domains. Moreover, IE may benefit NLP systems such as Machine Translation, Question Answering, Text Summarization and Opinion Mining systems. Arabic IE (AIE) has not reached the same level of maturity achieved for languages like English, due to the Arabic language specific challenges[10] and the inadequate research effort in the field.

In this work we seek to apply IE methods in the legal field. Our choice of this field was motivated by several factors, the most important of which was our assumption that law documents have highly similar format and structure that may simplify IE[2]. The documents usually begin with basic metadata like date, type, location and number. Then, additional sections are added with a good degree of uniformity across documents. We assumed that this may facilitate writing unified code for larger classes of documents and also simplify machine learning for AIE.

Machine Learning (ML) approaches are based on dividing the annotated dataset into two parts: the training set for building the model and the test set for assessing the model success. One needs a source for these datasets required for ML. The online legal repository, Al-Muqtafi System, with its extensive human annotated collection of law documents meets this requirement. The system was developed by the Institute of Law (IoL) at a local uuniversity. It is basically a repository of major pieces of legislation, in Arabic, enacted in Palestine since the mid-1900s. It has two components: the Legislation database and Judgments database. The legislation database has about 13,000 pieces of legislation with more than 50,000 pages accessible to users, while the Judgments database has about 23,000 judicial judgments. All this was captured and annotated manually, a task that required a lot of human effort, but also made a good basis for applying machine learning methods. The system provides users with search facilities to retrieve a specific document but has no tools for the extraction of information from the retrieved documents. The system leaves that task to humans.

Our IE-based system accepts input in the form of raw text files and passes them through its three stages: The first is the *preprocessing* stage, in which the text is tokenized, normalized, split into paragraphs (Judges, Parties, Facts, Reasoning, Verdict and Conclusion) and manually labeled in preparation for the next stage. The main tool used in this stage is MADAMIRA which takes care of normalization, tokenization and provides POS tagging that is used in the next processing stage. The second stage is *feature selection* where multiple features are extracted from documents and used to build the Named Entity Recognition (NER) model to extract the entities needed in the following stage. We used Weka ML tool to build our NER model. The third stage is *Relation Extraction*, where we use a combination of Weka (ML) models and manually crafted rule-based approach to extract

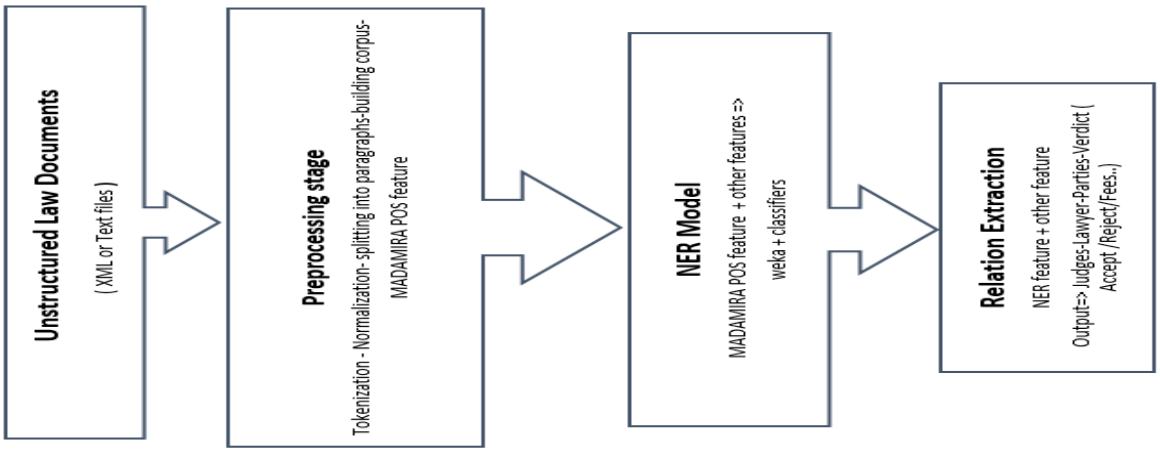

Figure 1.1 Flowchart of Information Extraction Work Steps

the relations. The relations we are after for each case are: Judges, Plaintiffs, Lawyers, Defendants and Verdict which may be thought of as *has-a* relations for the case. Figure 1.1 shows the flow of these stages.

B. Arabic Information Extraction Work Tools

Most current work in IE has an English focus[8], resulting in many tools and projects on issues like NER and Relation Extraction with high accuracy available freely for researchers and programmers. Unfortunately, Arabic IE is still in its infancy[11]. Here, we review some tools we found useful for our work on AIE, their potential and limitations and our selections.

1) Standard Arabic Morphological Analyzer (SAMA)

SAMA is a morphological analyzer that is an improved version of the Buckwalter Arabic Morphological Analyzer (BAMA) which was developed by researchers at Linguistic Data Consortium (LDC). SAMA takes an Arabic word and produces all possible morphological analyses regardless of context[6]. It accepts input files with .txt extension using UTF-8 encoding and outputs an .xml file that has the analyses for all input words. SAMA is used in MADAMIRA morphological analyzer and disambiguation system described later.

2) Al Khalil Morphological Analyzer

Al Khalil morphological analyzer can process non-vocalized as well as vocalized texts. The analysis process passes through the following steps: *preprocessing* which cleans and divides text into words, then eliminates diacritics and non-Arabic words. The next step is *segmentation*, where words are divided into proclitics + stem + enclitics. Then the word proceeds through three phase analysis, first as a non-derived word, second as a derived word and third as a verb to get morphological features[7,3]. Compared with BAMA, SAMA and its early version, AL Khalil Morpho Sys 1, Al Khalil Morpho Sys 2 has shown the best performance and SAMA came second in our testing. Since Al Khalil Morpho sys 2 was not available to us as open source tool, we decided to use SAMA analyzer for the NER task: first, because we could have a copy of the software from LDC and second, it's used by MADAMIRA tool, which we used in our work.

3) MADAMIRA Tool

MADAMIRA is a morphological analyzer and disambiguation system that supports MSA and Egyptian (EGY) dialect. It is a combination of the systems MADA and AMIRA and provides many morphological features taken from MADA such as POS tagging, lemmatization, tokenization and gender plus two features taken from AMIRA: base phrase chunking (BPC) and NER[1].

The preprocessor in MADAMIRA cleans the input text then converts it to Buckwalter transliteration[13] for passing to SAMA module which accepts data in that format. SAMA morphological analyzer produces all possible analyses of the words out of context. Then the text is passed to a feature modeling component which derives prediction for each analysis of the word. Next, Analysis ranking component is used to score each word analysis list. The highest scored analysis is then passed to tokenization which produces many schemes of tokenization based on the request of the user. For BPC the AMIRA style component divides the words into chunks using SVM models. Finally, the named entity recognition (NER) component uses SVM models to mark the *Person*, *Location*, *Organization* (PER, LOC, ORG) entities in the text[9]. What makes MADAMIRA important is that it is a disambiguation system not only an analyzer. SAMA and Al Khalil don't support NER task, while MADAMIRA does that with good accuracy. Since we need to extract the names of people in legal documents such as judges, lawyers and plaintiffs, NER is a core task for us and we settled for MADAMIRA. Another option that we are currently exploring is Farasa[4].

II. Methodology and Results

We worked on two tasks: NER and Relation Extraction, with the second task using the output of the first to extract the needed relations. This division

was based on the assumption that relations may be present between widely varying entities like persons (single or group), organizations and maybe more. Skipping NER may reduce efficiency. Next we give the details of this approach.

A. System Specifications

1) The System Input: The system accepts text (.txt) files. If given an xml (.xml) file, that needs to be converted to a text file in the preprocessing stage. Our selected documents are of the Appeal law cases category (قضايا استئناف).

2) The System Output: The output of the system is the extracted relations for each case as follows:

- The names of the case judges (أسماء القضاة).
- The names of the case lawyers (أسماء المحامين).
- The names of the case plaintiffs and defendants (الجهات المستأنفة والمستأنف عليها).
- The verdict of the case and the verdict type: with or without penalties (نتيجة الحكم / قرار المحكمة).

The choice of these relations was based on them being of interest to the current users of Al-Muqtafi system and was decided upon in consultation with the system users. Since our extracted relations are associated with cases, it is possible to extract more complex relations between entities, for example *Shared_a_Case*, through the case links. We believe that the approach can also be extended to extract other types of relations like the reasoning behind the verdict and the laws referenced in the given case.

3) The Programming Language and Tools Used: We used Python as our main programming language since it allows rapid prototyping and its powerful and elaborate set of standard libraries make it fit for many large-scale software projects. It also can run on multiple platforms.

4) The Type of input Law Documents: We studied many types of cases and decided to work with the appeal cases because they have common patterns, structure and high similarity across lawsuit types (theft, murder, … etc.). In this class we further limited our work to leased premises cases, mostly for privacy considerations. So our data set is in the appeal cases for leased premises lawsuits.

B. Preprocessing Stage

In this stage we performed the following steps:

1) Separate Raw Data into Paragraphs:

Instead of dealing with document text as one unit we split it into paragraphs in order to do the learning process on comparable paragraphs from different documents. That is, we used individual paragraphs as the learning documents for different relations. This was based on experiments we conducted which gave poor performance when we worked with full documents as a single unit for each of the extracted relations. The paragraphs we identified are: Judges, Parties, Facts, Reasoning, Verdict and Conclusion.

The split was done automatically and was based on the presence of keywords specific to certain paragraphs. So our splitting algorithm was based on some frequent patterns of (possibly normalized) keywords and phrases in each paragraph. Table 2.1 list some of the expressions used to distinguish paragraphs in the case text. Clearly, these paragraphs have the relations to be extracted, but the mapping between the paragraph text the relations is not straightforward.

Table 2.1. Expressions to Identify Paragraphs

Paragraph	Sample words/Phrases
Judges	الهيئة الحاكمة
Parties	طاعن، مستانف
Facts	الإجراءات، محكمة بداية
Reasoning	تدقيق مداولة ،تدقيق الحكم
Verdict	لهذه ،بناء على ما تقدم الأسباب
Conclusion	صدر تدقيقا ،صدر وتلي

2) Parsing Text Files Using MADAMIRA:

We used MADAMIRA tool to get the POS tags for the input text.

3) Stop Word Removal and Manual Labeling:

We used standard lists for Arabic stop word removal. The tags (classes) we used for labeling named entities are the ones used in CoNLL: Person names (PER), Organizations (ORG), Locations (LOC) and miscellaneous (MISC) which we replaced by Others (O). We used the (B-Begin and I-Inside) scheme for text chunking (Table 2.2).

The first part of the entity/relation is (B) and any following parts of the same entity are annotated with(I). For example:

سامي B-PER

صرصور I-PER

For the NER of type Person "سامي صرصور"

For the relations, the classes we used for the relations we want to extract from the text are:

- JUDGE (القاضي)
- LAWYER (المحامي)
- PLAINTIFF (الطاعن او المستأنف)

Table 2-2 NER Tags

NER TAG	NER Definition
B-PER	Beginning of person
I-PER	Internal of person
B-LOC	Beginning of location
I-LOC	Internal of location
B-ORG	Beginning of organization
I-ORG	Internal of organization
O	Other

- DEFENDANT (المطعون او المستأنف عليه)
- VERDICT (حكم) statements which are
  - Accept Verdict (قبول الحكم) ,
  - Reject Verdict (رد الحكم) ,
  - Revoke Verdict (فسخ القرار) ,
  - Ratification (تصديق القرار),
  - Restoration (إعادة الارواق),
  - Fee Payment (إلزام المستأنف بالرسوم والمصاريف

C. Named Entity Recognition (NER) Process

1) Feature Selection:

To build a NER model we processed our text to generate the input file for Weka. Table 2-3 describes the 6 features selected for the NER learning process, based on several experiments we conducted.

Table 2-3 NER Features

Attribute	Description
Position (#)	The word position in the paragraph
Word Length (#)	The number of characters of a word.
Paragraph ID	The paragraph to which the instance (entity) belongs.
POS Tag (Tag)	The morphological type of the word: Noun, Verb or something else.
Previous Word Tag	The type of the previous word to help in determining full names: (First, middle and last name).
PER Weight Tag (#)	A numeric value representing how likely the current word is a PER according to text patterns. Described below in little more detail.

Note that *PER Weight Tag* is a number representing the likelihood that the word has a PER tag. For example, if the previous phrase was " هيئة حاكمة" or "محامي" it is highly possible that the current word is a judge's name so a high weight is given to the word. The final attribute is the **class** (output) which can be one of the tags in Table 2-2.

2) Building the NER Model

We ran the learning data from the 120 documents training set, using various classifiers: we experimented with Maximum Entropy (Logistic), Support Vector Machine (SMO), Decision Tree (j48) and K-nearest neighbor (IBM) as the ones used frequently in the information extraction field. We studied evaluation metrics (Recall, Precision, and F-measure) and produced Table 2.4. Decision tree was the best classifier and is the one selected for building our final model. The best accuracy we got for NER learning was 87.2% using decision tree classifier with 13 fold testing mode.

Table 2.4. ML Classifiers and their Metrics

Algorithm	Precision (P)	Recall (R)	F-Measure (F)
Maximum Entropy (Logistic)	61.7 %	63.5 %	61.4 %
Support Vector Machine (SVM)	65.4 %	66.2 %	58.5 %
Decision Tree (13-fold testing mode)	**86.4 %**	**87.2 %**	**86.8 %**
K-nearest neighbor(IBM)	84.3 %	84.9 %	84.6 %

3) Feature Effect on NER Classification:

The 6 features used in predicting the correct NER tag were: Paragraph Id, Position, PER-weight, POS tag, Previous POS tag**,** Word Length. We studied the effect of each of these features on the system accuracy. Table 2-5 shows how each feature affects the performance when zeroing that feature alone for the 120 documents in the training set. The table shows that the paragraph ID is the most discriminating feature as ignoring it decreases the accuracy by 8.2%.

Table 2-5. NER Feature Accuracy Effects

Feature	Accuracy after setting feature to 0	Accuracy decrease
Paragraph Id	79.0 %	8.2 %
Position	81.6 %	5.6 %
PER-weight	82.4 %	4.8 %
POS tag	83.8 %	3.4 %
Previous POS tag	85.6 %	1.6 %
Word Length	85.0 %	2.0 %

D. Relation Extraction Process

We separated our work into NER then Relation Extraction (RE) on the assumption that it may work better for cases when the relation entities can be quite diverse in terms of the participating entities.

1) Relation Extraction Flow

Relation extraction is the main goal of our work. The diagram in Figure 3.1 shows the three steps of this stage when using the ML tool Weka.

The input is the corpus we intend to extract relations from its documents. The NER results for the elements in that corpus are the features for building the relation models. The Relation Extraction system is built using a combination of two approaches: ML and Rule-Based. Case *Judge, Lawyers* and *Verdict* relations are extracted using ML techniques while Case *Parties: Plaintiffs* and *Defendants* relations are extracted using Rule-Based approach.

The reason is that the parties section structure, which includes plaintiffs and defendants, is more complicated and nonuniform than Judges and Verdict sections to the degree that ML models couldn't predict plaintiffs and defendants with sufficient accuracy. The diversity in Parties which can be one person, two persons or even more. Also, the name of the person may consist of 3 parts (first, second and family name) but occasionally it consists of 2 or even 4 names and sometimes the Parties are not even persons. Another reason, place of residency of the party may be included but not always. All this makes learning difficult.

2) Machine Learning (ML) Relation Extraction Subsystem for relations ***Judge, Lawyers, Verdict****:*

The first four features in Table 3.1 are common to the three models. The feature *Paragraph Section* is used in building the *Parties* model and the feature *Verdict Weight* to build *Verdict* model. These features are generated using python code that outputs an arff file including the entities recognized from the corpus at the NER stage, and their generated features.

3) Rule Based Relation Extraction Subsystem for Relation ***Parties****:*

In this subsystem, *Parties* relation, viewed as *Case_Has_a* (B-PLAINTIFF, I-PLAINTIFF, B-DEFAENDANT, I-DEFENDANT) relation, is extracted using rules written according to the pattern detected for writing the *Parties* names in the training set. We used

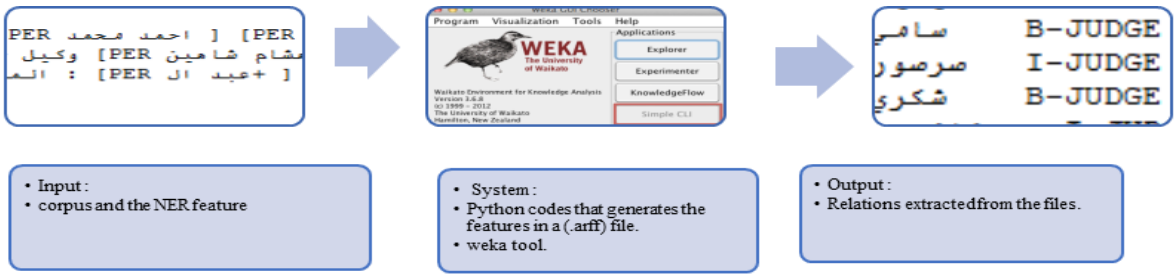

Figure 3.1. Relation Extraction Flow

In this subsystem, three models were constructed; one predicts the case Judge (B-JUDGES, I-JUDGES), the second predicts the case Lawyers (B-Lawyer, I-Lawyer) and the third predicts the Case_Verdict (B-Accept_Verdict, I-Accept_Verdict, B-Reject_Verdict, I-Reject_Verdict, B-Restoration, I-Restoration, B-Revoke_Verdict, I-Revoke_Verdict, B-Fee, I-Fee) relations. The classifier used for the three models is Decision Tree (j48) using Weka.

dictionaries, gazetteers and a list of Arabic names in finding the full name of the party and its place of residence. Rule-based extraction gave much better results than ML: when experimenting using the ML approach only 60% of the parties were classified correctly while using rules gave better results: 90% of the *Parties* relation for the 120 case documents were classified correctly.

Table 3-1. Machine Learning Relation Extraction Features

#	**Attribute/feature**	**Description**
1	Position	The position of the word within the paragraph.
2	Word length	The number letters in the word.
3	NER tag	The NER tag for this word taken from the NER feature file. Possible values: [O, B-PER, I-PER, B-LOC, I-LOC, B-ORG, I-ORG].
4	Previous NER Tag	The NER tag for the previous word. Example given "I-PER محمد علي B-PER)" the previous NER tag for the word (علي) is B-PER. Possible Values: [O, B-PER, I-PER, B-LOC, I-LOC, B-ORG, I-ORG].
5	Paragraph Section	This feature is a numeric value that splits the Parties Paragraph into sections, section for Plaintiffs, section for lawyers and section for Defendants according to significant keywords. This feature is important in recognizing Lawyers' names.
6	Verdict Weight	Verdict Weight is an integer value given to some important words in the Verdict Paragraph such that they express the verdict like the sentence "رد الحكم"and the sentence "قبول الاستئناف موضوعا". E.g. The words that expresses Accept Verdict are given the value 10 and the words Reject Verdict are given the value 20 and so on. This feature is used in building verdict model.
7	Class	The output class that has the relations we need to extract. Possible Values: [B-JUDGE, I-JUDGE, B-AcceptVerdict, I-AcceptVerdict, B-RejectVerdict, I-RejectVerdict, B-Ratification, I-Ratification, B-RevokeVerdict, I-RevokeVerdict, B-Restoration, I-Restoration, B-Fee, I-Fee]

4) Relation Extraction Output:

The extraction results can be displayed as text in the system interface. The results are also saved to files (judges.txt, parties.txt and verdict.txt) for possible later usage, e.g. export to a database. The user can filter which relations to extract for a given document through a user interface.

5) Why Decision Tree Classifier?

We experimented with different models using different classifiers for relation extraction. The classifiers we tested were Maximum Entropy, SVM, Decision Tree and IBK since they showed the best results within the IE learning experiments in the surveyed literature. In our experiments we found the decision tree to be the best classifier for our dataset of 120 cases. The F-measure for the three Relation Extraction models using the Maximum Entropy classifier was 72.4%, 36.2% and 87.2%, for the SVM classifier the percentages were 71.4%, 33.1% and 81.3% and using Decision Tree classifier the values were 90%, 70% and 94%.

6) Why Three ML Relation Extraction Models?

One may ask why we built three models to predict the relations we sought and not a single model? Actually, we performed two trials: we started our work by building one model that extracts all relations and uses the entire text of each case document in the dataset, and then experimented with the paragraphs we need for a given relation in another. Correct model prediction results for full documents were 60% for *Judges*, 40% for *Parties* and 25% for *Verdic*t. Our study found that the document size varies greatly from case to case. One may find a case with 200 entities and another with many more. We decided not to take the whole text but only the paragraphs most relevant to the relation. This resulted in a significant improvement especially for *Verdict* relation. As a result, we decided to consider three models, with each model dedicated to predicting one relation using the automatically identified relevant paragraphs.

III. Conclusions and Future work

The output of our system proves that automatic information extraction from Arabic law documents is feasible and can be achieved with good performance. This is a proof of concept work that needs further extension to make it more usable. One can think of several possible extensions such as dealing with different types of cases in a given legal system that may be more general and thus more challenging to process. One can also consider cases from other legal systems, where the document structures may differ substantially from what we worked with. Our initial work suggests that learning in one legal system may not work equally good when applied to a different legal system due to substantial variations in style and requirements. Also using other, more advanced, Arabic NLP tools[4] may be explored for possible effect on performance. One interesting research area to explore is using deep learning for AIE.

Another extension, is for more relations to be extracted so that the system becomes more powerful. For example, the Reasoning for the verdict, the amount of penalties the parties have to pay, the dates the lawsuit was held, the place where the lawsuit took place and much more. Another improvement aspect is the presentation of the results in more user friendly manner or using more natural language phrasing. Arabic Information Extraction from documents in other fields such as health and finance are other possible extensions.

References

[1] M. Al-Badrashiny, M. Diab, N. Habash, M. Pooleery, O. Rambow and R. Roth. (2013, Nov). "MADAMIRA v2.0 User Manual" [Online]. http://www.nizarhabash.com/publications/MADAMIRA-UserManual.pdf

[2] J. Andrew and X. Tannier. "Automatic Extraction of Entities and Relation from Legal Documents". Proceedings of the Seventh Named Entities Workshop, PP. 1–8. Melbourne, Australia, July 20, 2018.

[3] M. Boudchiche, A. Mazroui, M. Bebah, and A. Lakhouaja, "The Morphological Analyzer Alkhalil Morpho Sys II," Proceeding of the 1st National Doctoral Symposium in the field of Arabic Language Engineering, Rabat, Morocco, 2014.

[4] K. Darwish and H. Mubarak. "Farasa: A New Fast and Accurate Arabic Word Segmenter". Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016). Portorož, Slovenia. May 2016. European Language Resources Association (ELRA) PP. 1070-1074.

[5] P. Jackson, K. Al-Kofahi, A. Tyrrell and A. Vachher, "Information extraction from case law and retrieval of prior cases", Artificial Intelligence, Volume 150, Issues 1–2, 2003, Pp. 239-290,

[6] M. Maamouri, D. Graff, B. Bouziri, S. Krouna, A. Bies, S. Kulick. LDC Standard Arabic Morphological Analyzer (SAMA) Version 3.1[Online], https://catalog.ldc.upenn.edu/LDC2010L01 [April. 16, 2017].

[7] A. Mazroui, A. Lakhouaja, A. Meziane, M. Bebah, A. Boudlal, M. Shoul, "Alkhalil Morpho Sys 1". Proceeding of the International Arab Conference on Information Technology, Benghazi, Libya, 2010.

[8] C. Niklaus, M. Cetto, A. Freitas and S. Handschuh. "A Survey on Open Information Extraction". Proceedings of the 27th International Conference on Computational Linguistics, pages 3866–3878Santa Fe, New Mexico, USA, August 20-26, 2018

[9] A. Pasha, et al. "MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic". Proceedings of the 9th International Conference on Language Resources and Evaluation, Reykjavik, Iceland. 2014.

[10] J. Piskorski and R. Yangarber, "Information Extraction: Past, Present and Future". In: Multi-source, Multilingual Information Extraction and Summarization, Eds. Springer Berlin Heidelberg, 2013, pp. 23-49.

[11] I. Sarhan Y. El-Sonbaty and M. Abou El-Nasr. "Arabic Relation Extraction: A Survey". International Journal of Computer and Information Technology. Volume 05–Issue 05, September2016

[12] T. Wu, "Theory and Applications in Information Extraction from Unstructured Text," M.S. thesis, Dept. CS. Eng., Lehigh Univ, Pennsylvania, 2002.

[13] Wikipedia. (2016, Oct. 29). BuckWalter Transliteration [Online], Available: https://en.wikipedia.org/wiki/Buckwalter_transliteration [April. 16, 2017].