Discovering Qur’anic Knowledge through AQD: Arabic Qur’anic Database, a Multiple Resources Annotation-level Search

Sameer M. Alrehaili
College of Science and Computer Engineering
Taibah University
Yanbu, Saudi Arabia
Srehaili@taibahu.edu.sa

Eric Atwell
School of Computing
University of Leeds
Leeds, United Kingdom
E.S.Atwell@leeds.ac.uk

***Abstract*—This paper introduces a novel resource for Arabic Qur’anic textual annotations: AQD, Arabic Qur’anic Database, providing an annotation-level search that draws on a number of available resources in a single query. In addition, it allows implementing a set of queries as rewrite rules, which is performed in a recursive way. The experiments show that our AQD is able to discover knowledge from very simple to very complex queries.**

Keywords—extraction, Quran, query language

I. Introduction

One of the main reasons for the lack of Arabic representation in the field of semantic web is the lack of Arabic information extraction tools, also one of the main problems for Arabic ontology [1]. For over 1,000 years, historical traditional Arabic grammars have documented in detail many Arabic linguistic or grammatical constructs. These books explained the mechanism for recognising these phenomena as sets of rules.

This paper presents a novel resource for available Qur’anic textual annotations: AQD, Arabic Qur’anic Database. It merges several annotations from different formats and resources, including morphological, structural, chronological and ontological. In addition, it provides an annotation-based search that draws on all available resources. This combination of different resources and search tool has not been achieved before for the domain of the Qur’an. Creating such an environment will enable researchers in Arabic computational linguistics and Qur’anic research to investigate new directions and new features for NLP learning tasks. The aim is to make use of these available annotations and integrate them in one single query. This will help researchers in Islamic studies discover hidden knowledge and mine Qur’anic information. It is also a helpful resource for researchers of classical Arabic grammar in that they can search for features from different resources for certain grammatical relations or patterns. We show statistical analysis of the combined annotations and examples of grammatical relations and linguistic phenomena that occur in Arabic as well as how to make queries to extract them.

The contributions presented in this paper are two-fold. First, different Qur’anic resources collected from different annotations and formats are combined. Second, an annotation-based search for making a query that integrates these different annotations in a single query is designed and implemented. In addition, this paper offers a new environment for discovering and mining hidden Qur’anic knowledge.

II. Combined Resources

Four different types of annotations have been combined into the MySQL Database, which is a Relational Database Management System (RDBMS). We included only annotations that cover Arabic text, cover all the entire Qur’an and are available for free. Therefore, the morphological layer is based on annotations from [2], which includes segments, the basic units that comprise the text of the Qur’an. For the topics, Qurany, a set of annotation by [3], is also available for the topic of verses. Another dataset collected from QAC includes verses and lists of concepts. The semantic layer [4] represents semantic tags for personal pronouns. In addition, chronological information and orders are also included based on [5], [6].

A. Morphological

The morphological annotations contain lexical information, which includes the diacriticised and undiacriticised formats of Arabic, Buckwalter transliteration, POS tag, root, lemma, and clitic information. Inflectional information includes person, gender, number, voice, aspect, mood and case. The derivational information includes the form and whether each segment is active or passive.

B. Structural

The structural annotations are the relations between the divisions and text parts of the Qur’an. For example, the relationship between chapters and verses, verses and words, or words and segments. In addition, this includes the relations between Juz and Hizb divisions. All these relations and entities are included in the database. These kind of annotations have been used for research in chronological analysis as well as being useful for ontology research based on the Qur’anic structure.

C. Chronological Annotations

Five types of orderings are included in the AQD database. Firstly, the order of verses according to their locations in the 114 chapters. This order starts from 1 and ranges to 114. The second order is the order of verses according to their revelation place, either Makkah or Madinah. It contains only two notations, 1 for verses revealed in Makkah and 2 for verses revealed in Madinah. Third order contains the 194 blocks of verses according to their revelation times; for example, block 1 contains the verses from 1 to 5 of chapter 96. Fourth order is an order proposed by Mehdi Bazargan (d. HS 1373/1995) and divides the verses into 22 groups. Fifth order is called the modified Bazargan order, and was obtained by combining some consecutive phases of the Bazargan to produce seven phases [5].

D. Ontological

Four ontological annotations have been included in this database. QAC ontology is composed of 294 concepts linked to each other hierarchically with Qur'anic words. Qurany datasets are made of topics that linked to each other and the verses of the Qur'an. QurAna has several referents linked to personal pronoun segments.

III. Related Work

For Qur'anic text, which is the focus of this paper, previous work has proposed different ways for applying annotation-level search. This section is focused on annotation-level search technique made for Arabic and in particular for Qur'anic textual resources.

A dictionary and morphological search tool were made freely available online by [7]. This feature allowed annotators to search over tag like POS and root in order to find the correct occurrences of the targeted meaning. However, they only allow searching over a certain group of morphological features such as POS, form, ROOT, Lemma and Stem-based on Buckwalter transliteration and does not allow searching using Arabic letters. [8] In her MSc dissertation unified a number of Qur'anic datasets and offered an annotation-based search using sketch engine. The author made a powerful tool that allowed the user to extracted knowledge based on the annotation of three different datasets (Qurany, QurAna and QAC).

Another related work in terms of applying annotation-level search includes [9], who applied JAPE rule using only one level of annotation which is the POS value to find the collocations among a set of defined POS tags.

[10] Introduced a framework for Arabic ontology learning from textual resources. They defined a number of lexico-syntactical patterns using JAPE to model extracting noun phrases, instances, concepts and different type of relations such as "is-a" and "has-a". The results were divided into two groups; in a first experiment by using Stanford syntactic parser on a limited number of sentences (34 sentences). Then, a second experiment covers 12779 words by using Arabic morphology analyser. The author could not parse the whole corpus due to it requiring all sentences to be well-formed in terms of punctuation.

AQD is the only database covering the entire Quran in a number of different levels, so it is not possible to comparatively evaluate the search method on a different database, unfortunately. However, we were not able to find any available tool that allows one to search documents at different levels with the ability of performing multiple queries as rewrite rules.

To solve that problem, we propose a query language for retrieving multiple resources annotations and able to perform a set of rules.

IV. The Search System

To make use of the combined annotations, we developed a sequential annotation-level system shown in Figure 1. This search allows users to extract concordance lists of a given complex query over multiple-level resources. The query is received by the system, comprising two components: parsing the query into nodes and transferring the nodes to SQL query format. The execution manages the implementation of the SQL query in a sequence as an array of linked lists. Finally, the relevant instances, which meet all given conditions in the query, are stored in text files.

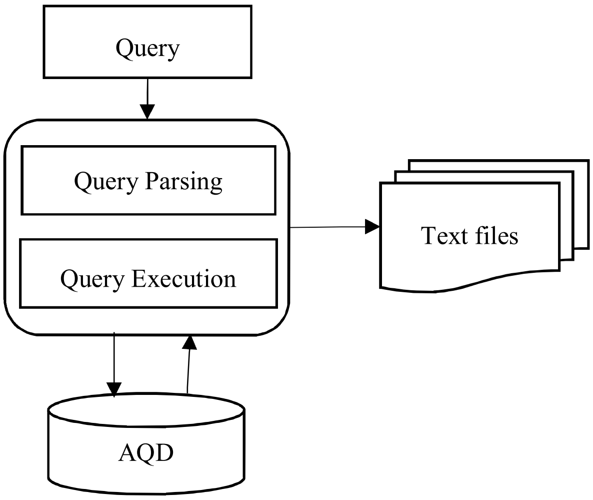


Figure 1 The Complex query implementation diagram

A. Query Syntax

Extended Backus-Naur forum (EBNF), one of most common context-free grammars for notation techniques, is used here to describe the query in detail. It is often used to describe how compilers and computer languages work. EBNF is used because we are presenting a language for a query that has different brackets and operators instead of words and letters.

Natural language can be described using another language that has regular expression and that deals with letters. However, this is not a natural language; it is made of different brackets and operators.

```
Patterns           =   “{”, {sequences}, “}”
Sequences          =   “[”, [Node] , [logical_operator], “]”,
                       [Wild_card]
Node               =   {Attribute, Match_operator, Value}
Logical_operator   =   “&” | ”|”
Wild_card          =   “*” | “+” “?”
Match_operator     =   “==” | “!=” | “re” | “not re” | “like”
Attribute          =   A tag in annotation
Value              =   “”, Value, “”
```

FIGURE 2 EBNF GRAMMAR OF THE QUERY

The previous code is called production rules and is written for describing the syntax of the complex query. The left-hand side shows the list of non-terminal symbols of the language, and the right-hand side shows the terminal as well as non-terminal symbols in a regular expression.

The first line shows how this language supports many patterns for extraction; the EBNF curly brackets mean that the content of the brackets is repeatable. For example, two patterns for extracting a sequence of one segment composed of nouns and another sequence of adjectives can be written as {[pos==“N”]} {[pos==“ADJ”]}. In this case, the execution component will run the first pattern and add its results to the file, then run the next pattern and append it to the file. A pattern starts with a square bracket followed by the details of the node and an optional logical operator as shown in line two. It ends with an optional wild card. There are differences between the brackets in the EBNF above; for example, the bracket inside the quotation marks means that it is an actual part of the query while the one without quotation marks is for describing the query. The square bracket in EBNF indicates that an element is optional; for example, the wildcard after the node is optional and the logical operator is optional, but the match operator is not. However, the whole node can be optional. This syntax is similar to the Poliqarp syntax in [11], but it does not support the same operators as are shown in the following section.

B. *Query Syntax*

A parsed query is received as a set of nodes, and the quantifiers for the nodes are received as an array. For each node, the function finds all instances in which the given conditions are meet (the pair of attributes and values). The node also has the current position of the found instance. The second run visits the last elements for each linked list and checks whether they are null or not. If a value is null, this means that the linked list is not relevant. If it has a position number, then the second node conditions are applied for the position after the current node.

No	pos	No	pos
1	N	7	V
2	N	8	N
3	V	9	N
4	P	10	V
5	V	11	V
6	N		

TABLE I SAMPLE DATA WITH ONE TAG

TABLE I shows an example of the sample of rows in a table composed of many tags. The example in TABLE II shows the transition based on sample data with only one tag.

Listing 1 {[pos==“N”] [pos==“N”][pos==“V”](2)}

Runs	Conditions	Quantifier	Output
1	[pos==“N”]	1	1 → 1 2 → 2 3 → 6 4 → 8 5 → 9
2	[pos==“N”]	1	1 → 1 2 2 → 2 N 3 → 6 N 4 → 8 9 5 → 8 N
3	[pos=“V”]	2	1 → 1 2 3 2 → 2 N 3 → 6 N 4 → 8 9 10 5 → 8 N 1 → 1 2 3 2 → 2 N 3 → 6 N 4 → 8 9 10 5 → 8 N

TABLE II AN IMPLEMENTATION FOR A ANNOTATION-BASED SEARCH

TABLE II presents an example of how search runs the annotation-based search over several resources base on the query Listing 1. The first column indicates the nodes of the query. Each run adds the instances that satisfy the condition of the current node. The second column shows the current node and the conditions that must be met in order to be added. The third column represents the repetition information of the node. As shown in the third row, there are two repetitions because this node was followed by the quantifier symbol (n). The third column shows an array of linked lists, which represent the retrieved instances. Every node generates an addition to this array depending on the given conditions and the linked lists, and no null values will be considered as the relevant value retrieved at the final step. Therefore, there is only one instance that satisfies the given query for this example, segments 8 to 11 in TABLE I.

V. Experiments

In this section, the usage of AQD and the annotation-based search will be described and evaluated by different examples of linguistic phenomena occurring in the Arabic text, starting with a simple search and ending with a very complex extraction task.

A. *Simple Annotation Based Search*

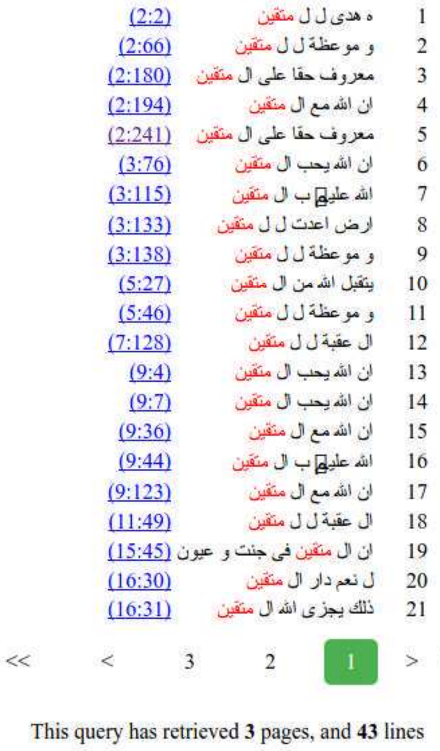
Each annotation that was included in the AQD can be used as an attribute, as is shown in the following query. The simple query is composed of one node and does a blank tag, such as {[]}. This query refers to any segment stored in the database. The curly bracket, '{}' denotes the node, while square brackets '[]' represents a value of a tag for that node. A node can be retrieved using multiple tags as well as written like a regular expression. Furthermore, a node can be retrieved as a subtask using rewrite rules. These functionalities allow interested researchers to build a block of rules for retrieving sophisticated NLP tasks.

Another simple example can be shown in Listing 2, which extracts all occurrences of the word 'متقين.' This query output is shown in Figure 3. The attribute diacritics represents the diacriticised Qur'anic text. Several different features can be used as attributes.

Listing 2 {[diacritics==”متقين”]}

The retrieved instances are matched to the given word. This will only search for one node matched by the given text. If there are other words with the same meaning that we wish to extract, this query will not be able to retrieve them.

Searching over lemma instead of the exact text of words will extract all occurrences of a word that belongs to the given lemma. The example shown in Listing 3 retrieves more instances because it is based on the lemma and not the word.



(2:2)	ه هدى ل ل متقين	1
(2:66)	و موعظة ل ل متقين	2
(2:180)	معروف حقا على ال متقين	3
(2:194)	ان الله مع ال متقين	4
(2:241)	معروف حقا على ال متقين	5
(3:76)	ان الله يحب ال متقين	6
(3:115)	الله عليم ب ال متقين	7
(3:133)	ارض اعدت ل ل متقين	8
(3:138)	و موعظة ل ل متقين	9
(5:27)	يتقبل الله من ال متقين	10
(5:46)	و موعظة ل ل متقين	11
(7:128)	ال عقبة ل ل متقين	12
(9:4)	ان الله يحب ال متقين	13
(9:7)	ان الله يحب ال متقين	14
(9:36)	ان الله مع ال متقين	15
(9:44)	الله عليم ب ال متقين	16
(9:123)	ان الله مع ال متقين	17
(11:49)	ال عقبة ل ل متقين	18
(15:45)	ان ال متقين فى جنت و عيون	19
(16:30)	ل نعم دار ال متقين	20
(16:31)	ذلك يجزى الله ال متقين	21

<< < 3 2 1 > >>

This query has retrieved **3** pages, and **43** lines

FIGURE 3 THE OUTPUT OF A SIMPLE QUERY

Listing 3 {[LemmaD==”متقين”]}

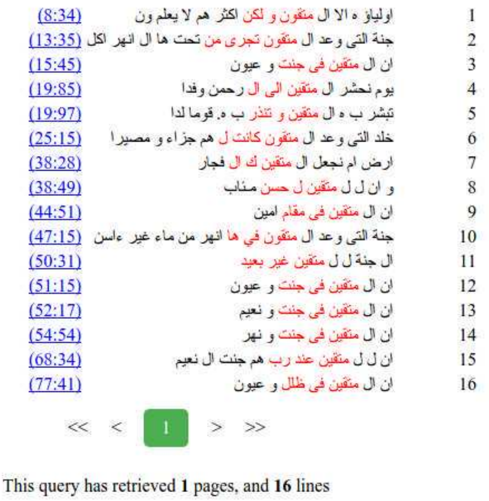
Query in Listing 3 retrieves any node that has a value "متقين" for the tag LemmaD. In this query, the output will be larger because words like 'متقون' will be considered as well.

The query language also supports all morphological annotations produced by [2]. To measure the quality of the extracted results, the QAC morphological search was used, and the results included all the given word form.

Another feature is the blank nodes it shown in Listing 4, which lets the user add as many as needed n-grams. The blank node just retrieves the next segment without checking content.

Listing 4 {[LemmaD==”متقين”][][]}

This is an example of how to use blank nodes. Two blank nodes were added to retrieve the two segments, followed the given lemma. Figure 4 shows the output of this query.



1	اولياؤ ه الا ال متقون و لكن اكثر هم لا يعلم ون	(8:34)
2	جنة التى وعد ال متقون تجرى من تحت ها ال انهر اكل	(13:35)
3	ان ال متقين فى جنت و عيون	(15:45)
4	يوم نحشر ال متقين الى ال رحمن وفدا	(19:85)
5	تبشر ب ه ال متقين و تنذر ب ه. قوما لدا	(19:97)
6	خلد التى وعد ال متقون كانت ل هم جزاء و مصيرا	(25:15)
7	ارض ام نجعل ال متقين ك ال فجار	(38:28)
8	و ان ل ل متقين ل حسن مئاب	(38:49)
9	ان ال متقين فى مقام امين	(44:51)
10	جنة التى وعد ال متقون في ها انهر من ماء غير ءاسن	(47:15)
11	ال جنة ل ل متقين غير بعيد	(50:31)
12	ان ال متقين فى جنت و عيون	(51:15)
13	ان ال متقين فى جنت و نعيم	(52:17)
14	ان ال متقين فى جنت و نهر	(54:54)
15	ان ل ل متقين عند رب هم جنت ال نعيم	(68:34)
16	ان ال متقين فى ظلل و عيون	(77:41)

This query has retrieved **1** pages, and **16** lines

FIGURE 4 ADD TWO NODE AS A SEARCH WINDOW

The query language also supports regular expression inside the values of the given attribute. Retrieval is accomplished by providing part of the text using a re-operator, as seen in Listing 5

Listing 5

```
{[diacritics re “%متقين”]}
```

B. *Construct State (iDAfa)*

iDAfa is a construction of two nouns, or what is known in English as a compound or possessive construction. iDAfa is important for dependency parsing and is also known as an annexation construct. It makes up more than 75% of Arabic sentences.

The following example in Listing 6 shows two things. First, it shows how to enables multiple rules for a single task. Second, it shows how to extract the construct of an iDAfa. The following example shows three rules that were defined to extract iDAfa.

To prepare rules for extraction, an expert of Arabic language must define them. We used the same rules applied in [12]. A construct-state in Arabic is composed of two parts, a noun and either a definite noun or a pronoun in genitive case. We found 4,348 iDAfa constructs in the Qur’an.

Listing 6

```
{[pos=="N" & case=="gen"][pos=="PRON"]}
{[ pos=="N"][pos == "PN" & case=="gen"]}
{[ pos=="N"][pos == "DET"][pos == "N" & case=="gen"]}
```

The same task can be represented using context-free grammar, as in Listing 7.

Listing 7

1. GDN->{[pos == "PN" & case=="gen"]}{[pos==”DET”][pos=="N" & case=="gen"]}
2. GIN->{[pos=="N" & case=="gen"]
3. IN->{[pos=="N"]}
4. iDAfa ->{[GIN] [pos=="PRON"]} {[GIN][GDN]}{[IN][GDN]}

This query allows users to implement a set tasks that depend on each other. For example the rule GDN is implemented as a terminal rule, while the iDAfa is implemented as non-terminal which is require to implement the GIN as a sub query of this rule. Listing 7.

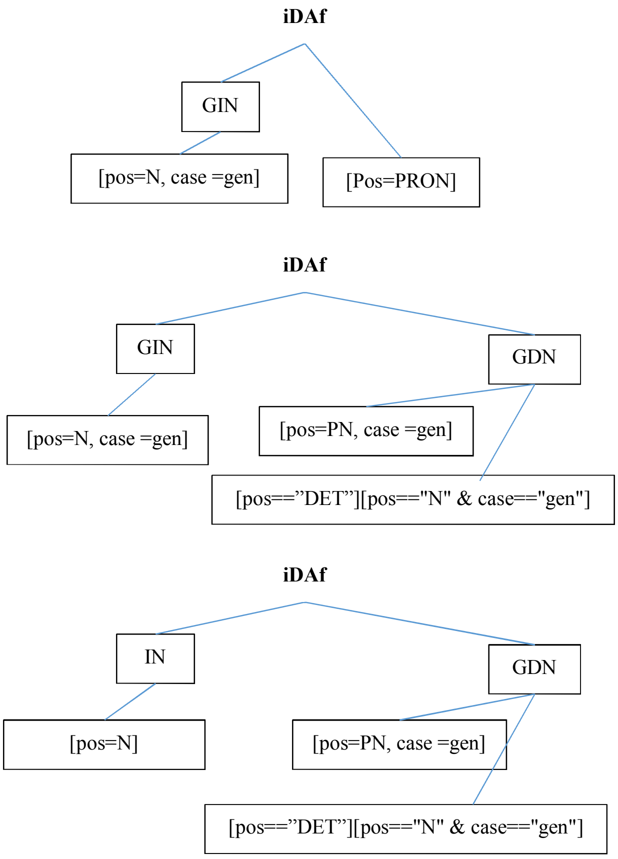
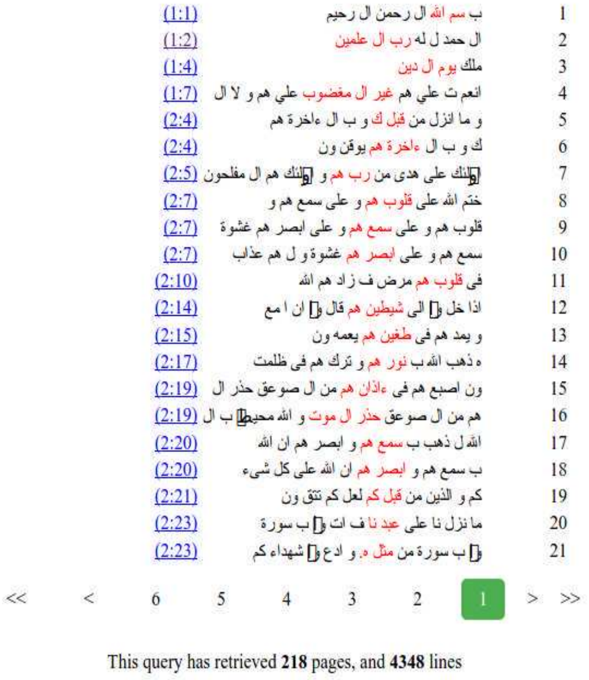


FIGURE 5 SUBGRAPHS OF THE REWRITE RULES OF IDAFA



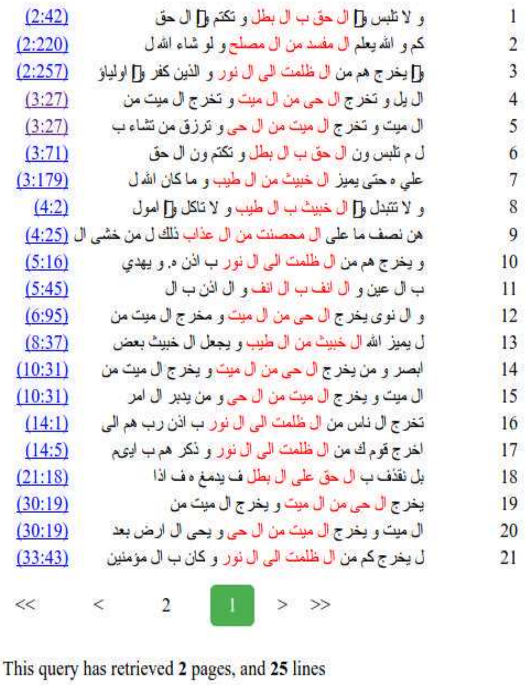
(1:1)	ب سم الله ال رحمن ال رحيم	1
(1:2)	ال حمد ل له رب ال علمين	2
(1:4)	ملك يوم ال دين	3
(1:7)	انعم ت علي هم غير ال مغضوب علي هم و لا ال	4
(2:4)	و ما انزل من قبل ك و ب ال ءاخرة هم	5
(2:4)	ك و ب ال ءاخرة هم يوقن ون	6
(2:5)	□لئك على هدى من رب هم و □لئك هم ال مفلحون	7
(2:7)	ختم الله على قلوب هم و على سمع هم و	8
(2:7)	قلوب هم و على سمع هم و على ابصر هم غشوة	9
(2:7)	سمع هم و على ابصر هم غشوة و ل هم عذاب	10
(2:10)	فى قلوب هم مرض ف زاد هم الله	11
(2:14)	اذا خل و□ الى شيطين هم قال و□ ان ا مع	12
(2:15)	و يمد هم فى طغين هم يعمه ون	13
(2:17)	ه ذهب الله ب نور هم و ترك هم فى ظلمت	14
(2:19)	ون اصبع هم فى ءاذان هم من ال صوعق حذر ال	15
(2:19)	هم من ال صوعق حذر ال موت و الله محيط□ ب ال	16
(2:20)	الله ل ذهب ب سمع هم و ابصر هم ان الله	17
(2:20)	ب سمع هم و ابصر هم ان الله على كل شيء	18
(2:21)	كم و الذين من قبل كم لعل كم تتق ون	19
(2:23)	ما نزل نا على عبد نا ف ات و□ ب سورة	20
(2:23)	و□ ب سورة من مثل ه. و ادع و□ شهداء كم	21

<< < 6 5 4 3 2 1 > >>

This query has retrieved **218** pages, and **4348** lines

FIGURE 6 THE GRAMMATICAL STRUCTURE (IDAFA) BASED ON THREE PATTERNS

C. Ontological relations between Definite Nouns



(2:42)	و لا تلبس و□ ال حق ب ال بطل و تكتم و□ ال حق	1
(2:220)	كم و الله يعلم ال مفسد من ال مصلح و لو شاء الله ل	2
(2:257)	و□ يخرج هم من ال ظلمت الى ال نور و الذين كفر و□ اولياؤ	3
(3:27)	ال يل و تخرج ال حى من ال ميت و تخرج ال ميت من	4
(3:27)	ال ميت و تخرج ال ميت من ال حى و ترزق من تشاء ب	5
(3:71)	ل م تلبس ون ال حق ب ال بطل و تكتم ون ال حق	6
(3:179)	علي ه حتى يميز ال خبيث من ال طيب و ما كان الله ل	7
(4:2)	و لا تتبدل و□ ال خبيث ب ال طيب و لا تاكل و□ امول	8
(4:25)	هن نصف ما على ال محصنت من ال عذاب ذلك ل من خشى ال	9
(5:16)	و يخرج هم من ال ظلمت الى ال نور ب اذن ه. و يهدي	10
(5:45)	ب ال عين و ال انف ب ال انف و ال اذن ب ال	11
(6:95)	و ال نوى يخرج ال حى من ال ميت و مخرج ال ميت من	12
(8:37)	ل يميز الله ال خبيث من ال طيب و يجعل ال خبيث بعض	13
(10:31)	ابصر و من يخرج ال حى من ال ميت و يخرج ال ميت من	14
(10:31)	ال ميت و يخرج ال ميت من ال حى و من يدبر ال امر	15
(14:1)	تخرج ال ناس من ال ظلمت الى ال نور ب اذن رب هم الى	16
(14:5)	اخرج قوم ك من ال ظلمت الى ال نور و ذكر هم ب ايم	17
(21:18)	بل نقذف ب ال حق على ال بطل ف يدمغ ه ف اذا	18
(30:19)	يخرج ال حى من ال ميت و يخرج ال ميت من	19
(30:19)	ال ميت و يخرج ال ميت من ال حى و يحى ال ارض بعد	20
(33:43)	ل يخرج كم من ال ظلمت الى ال نور و كان ب ال مؤمنين	21

<< < 2 1 > >>

This query has retrieved **2** pages, and **25** lines

FIGURE 7 SOME OF THE ANTONYMIC RELATIONS IN PREPOSITIONAL PHRASES

Figure 7 shows the ontological relationships between Qur'anic nouns in prepositional phrases. These type of relations need very complex rules that combines several features to be included in the query to extract these instances.

VI. CONCLUSIONS

In this paper, a new database AQD Arabic Qur'anic Database has been combined from different annotation sources. AQD includes morphological, chronological, ontological and structural annotations. We conducted an implementation of an annotation-based search for different types of annotations for the Qur'an. The query supports differently-resourced annotations. This tool allows Qur'anic researchers to mine and discover new patterns within these annotations. The tool used in the experiments is available on our website, together with the source code of the annotation-based search. The package of the source code and the database are available for download. The experimental results show that the environment can extract a syntactic construct of Arabic called iDAfa, and antonym relationships occur in prepositional phrases.

This work supports more annotations for the annotation-based search of the Qur'an and allows users to make a complex query that integrates a number of annotations in one single query. In addition, it supports context-free grammar for complex extraction tasks like those shown in the extraction of iDAfa.

There is much room for further work here. We plan to investigate the possibility of adding more annotations and features for the AQD.

REFERENCES

[1] L. M. Bin Saleh and H. S. Al-Khalifa, "AraTation: An Arabic Semantic Annotation Tool," in Proceedings of the 11th International Conference on Information Integration and Web-based Applications, 2009, pp. 447–451.

[2] K. Dukes and N. Habash, "Morphological Annotation of Quranic Arabic.," in Lrec, 2010, pp. 1–3.

[3] N. Abbas and E. Atwell, "Qurany," 2009. [Online]. Available: http://quranytopics.appspot.com/.

[4] A. Sharaf and E. Atwell, "QurAna: Corpus of the Quran annotated with Pronominal Anaphora.," in LREC Language Resources and Evaluation Conference, 2012, pp. 130–137.

[5] B. Sadeghi, "The Chronology of the Qurʾān: A Stylometric Research Program," Arabica, vol. 58, pp. 210–299, 2011.

[6] S. M. Alrehaili, "The Chronology of the Texts in the Holy Quran According to NLP," Unpublished MSc Dissertation, School of Computing, University of Leeds, 2012.

[7] K. Dukes, "Statistical Parsing by Machine Learning from a Classical Arabic Treebank," PhD Thesis, School of Computing, University of Leeds, 2013.

[8] L. Aldhubayi, "Unified Quranic Annotations and Ontologies," University of Leeds, 2012.

[9] S. Zaidi, A. Abdelali, F. Sadat, and M.-T. Laskri, "Hybrid Approach for Extracting Collocations from Arabic Quran Texts," 2012.

[10] N. Ghneim, M. Al, and S. Ali, "Building a Framework for Arabic Ontology Learning," in Knowledge Man- agement and Innovation in Advancing Economicss: Analyses & Solutions, 2009, p. 1730–1 1735.

[11] D. Janus and A. Przepiórkowski, "Poliqarp An open source corpus indexer and search engine with syntactic extensions," pp. 85–88, 2007.

[12] T. Arts, Y. Belinkov, N. Habash, A. Kilgarriff, and V. Suchomel, "arTenTen: Arabic Corpus and Word Sketches," Journal of King Saud University Computer & Information Sciences, vol. 26, pp. 357–371, 2014.