Unified Quranic Annotations and Ontologies

Luluh Aldhubayi

Msc Artificial Intelligence

2012/2013

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student):________________________

Summary

Recently, researchers have shown an increased interest in finding linguistic features of religious text. Consequently, different linguistics annotations and datasets have been constructed.

This project conducted an approach to unifying different Quranic datasets. The datasets merged in the project are: Quran Annotations Corpus (Dukes, 2012), Pronoun reference (QurAna) (Sharaf, 2012) and Qurany concept project (Abbas,2008).

The project started by unifying the dataset formats to XML and then merging them in one XML file. The merged Quranic dataset has been ported to the Sketch Engine tool to enhance the usability of the dataset and to allow it to be explored online by typical users.

The second phase of the project was to map the XML file to OWL ontology so as to enhance the semantic relationships between XML elements. An ontology design proposal was conducted initially, but then a facility in Protégé editor converting XML to OWL was used.

The final output created a Quranic dataset in which a user can find relationships between text in three dimensions; word morphology analysis, pronoun reference analysis and verse semantic analysis.

Acknowledgments

First and foremost, I would like to express my gratitude to God Allah. Thanks for giving the will and the strength to carry on.

I would like to express my sincerest thanks to my supervisor, Dr. Eric Atwell for his guidance and support. I feel truly privileged to have had the chance to know him and work with him. He is an exceptional and brilliant advisor and human being.

I would like to thank the project's assessor Dr. Lydia Lau for her great feedback and guidance during the project's interim report and progress meeting.

I would like to sincerely thank my parents, husband and daughter; for their understanding and endless love and supporting through the duration of my study.

Table of Contents:

Chapter 1: Introduction

Understanding the problem:

The Holy Quran is the Muslims' sacred scripture, and the Quranic text is considered the most perfect example of classical Arabic (Sha'rāwī, 1993). Therefore, linguists use computing techniques to analyse Quranic texts in order to accomplish many goals. In fact, people who are interested in the Holy Quran have both religious and linguistic motivations, such as studying the behaviour of a word in every occurrence in the Quran to find a pattern or a Quranic language rule. Usually, to study a Quranic word, three different levels of language analysis are considered. The first analysis is the syntactic analysis, which involves mainly studying the relations between words in a sentence or verse. The second analysis is the morphological analysis, which involves studying the word's grammatical features, while the third analysis is the semantic analysis, which mainly describes the word's meaning.

In the Quran there are many verses in which the pronouns referring to Allah are plural while in other verses the pronouns are singular. However, in fact there is only one Allah, as many verses indicate (e.g. verse, chapter). The reason for having different morphological features is becoming a debate among Quranic researchers. However, it is agreed that there is a relationship between the verse concepts and the morphological features, because in some verses the pronoun referring to Allah is in the singular form-while others are in the plural. Thus, a comprehensive dataset showing the pronoun reference with its morphological features and verse concepts is needed to resolve Quranic debates.

Another debate is about the synonyms and antonyms of Quranic words. For instance, the word (rain) is mentioned in the Quran using three Arabic words (gyv, mA' and mtr). Each has a different root, but all have the same meaning. By considering the context/verse concept related to that word, we can infer the purpose of using each word. For example, the words (gyv) and (mA') have been used in verses with award and goodness concepts while the word (mtr) has been mentioned in verses with scourge and bad deeds concepts. Thus, we cannot consider mtr as a synonym for gyv and mA' because the Quran never uses the word (gyv) in relation to bad deed concepts.

However, taking into account the Quranic researchers' motivations, there is an obvious demand for a system which makes it possible to search and analyse Quranic words. This search aims to have three levels of language analysis to produce a comprehensive word analysis.

In the second chapter, I investigate the current and online available Quran-related projects, and conclude that these Quranic search tools have some limitations. These limitations can be categorized in three points. First, some Quranic systems do not use all the Quranic chapters, but instead, restrict their system to a few chapters because of issues related to the project team. Second, the search tool used in some projects lacks capabilities in retrieving the Quranic words, for example, it is not possible to retrieve feminine nouns using the Quranic Arabic Corpus search tools. Third, not all systems use a common language; some use the Arabic vowelled script and others use the Buckwalter script. However, it is argued here that unifying the declarations in each dataset will help get more results from different datasets.

Furthermore, instead of having many systems and datasets sharing the same text but having different functions and scripts, I argue that merging datasets' annotations will overcome the keyword retrieval limitations and solve the difference in scripts by using a common script. In addition, putting the unified formatted datasets in one dataset helps achieve data integration, where the unified dataset is independent of other datasets. (Lenzerini et al, 2002). Nonetheless, there is another challenge as each dataset has its own file format and structure. So, to have a unified Quranic annotations dataset, I need to unify the datasets' formats as well.

In addition, by investigating the current Quranic ontologies, I conclude that there is a demand for a Quranic ontology. Most ontologies are dedicated to describing one aspect of the Quran; for instance, the ontology that studies antonyms was dedicated to finding the opposite nouns sharing the Time concept only (Al-Khalifa et al., 2009). However, to date, there has been no research to produce an otology that describes a word semantically and grammatically. The project aims to develop a unified ontology describing the Quranic words in relation to their semantic and grammatical features. This ontology will improve web/text searches. So, if a user searches for a word that has a concept meaning or morphological features, other similar Quranic words sharing the same features can be shown in the search results.

The project aim:

The aims of this project are as follows:

1- create a unified annotations dataset by merging three Quranic annotations datasets
2- port the unified annotations dataset to a corpus tool in order to explore the Quranic text and enhance the dataset's usefulness by making it accessible online for researchers
3- convert the unified annotations dataset to the ontology format
4- examine the unified ontology by using an ontology editor.

Objectives:

The objectives of this project are as follows:

1- understand the problem by investigating the existing Quranic datasets and ontologies
2- collect the Quranic datasets and understand their annotations, formats and tools if available
3- design three software to convert the three datasets' formats into a unified format
4- design software to merge the three datasets into one unified dataset
5- find an appropriate corpus system to present the unified annotations dataset

6- test the corpus under the chosen system

7- evaluate the usage and usefulness of the unified corpus to Quranic researchers concerned with the Arabic language and Quranic researchers

8- map the unified annotations dataset to ontology format

9- find an ontology tool to examine the converted ontology.

Minimum requirements:

1- acquiring datasets from three different Quranic datasets

2- processing the acquired datasets and converting them into one unified format

3- combining datasets to form one unified dataset

4- uploading the dataset into a corpus exploring system

5- converting the unified dataset to an ontology format

Degree relevance:

This project is conducted based on the knowledge and skills achieved from two modules of my MSc Artificial Intelligence course. The COMP5410M Language module addresses many of the fundamental principles of computational linguistics, such as corpus annotation, tagging, and tag sets. In addition, the COMP5450M Knowledge Representation and Reasoning module focuses on the fundamental principles in knowledge representation, such as logical analyses, description logic, ontology in general, and AI knowledge bases. Therefore, the two modules have provided me with a strong background about the project.

Research methodology:

The aim of the project is to create a unified annotations dataset in order to fully understand the Quranic text and to increase the users' knowledge. For instance, finding text patterns and clustering the text with their grammatical and semantic features is a demand made by many Quranic scholars and Arabic language researchers. However, these goals are considered as data mining tasks as the aim is to increase the knowledge about the data. A suitable methodology for this project is the CRISP-DM approach. This approach has several stages, as follows.

Business understanding

This phase defines the main objectives of this project. The project aims to help Quranic researchers and Arabic language experts and learners. Its objective is mainly to understand clients' needs by investigating and analysing their requirements of having unified annotations datasets and ontologies. According to Abbas et al. (2013), the current text search is limited to the keyword itself and works by introducing the semantic web and tagging each word with its semantic and grammatical features. The search output might show other un-typed words sharing similar features.

Therefore, this project aims to address the requirements of Quranic researchers by investigating and merging three of the current Quranic annotations datasets and uploading the unified annotations dataset to an exploring and search tool. By having a unified annotations dataset, a user can perform a comprehensive analysis of words, verses and chapters as well. The unified annotations dataset aims to increase the users knowledge by showing different annotations per word.

Data understanding:

The aim is to acquire and understand a range of annotations datasets and decide whether each annotations dataset is appropriate to accomplish our goals. To understand a dataset, a comprehensive reading of the project is needed.

In addition, this study aims to investigate the datasets by inspecting the covered chapters. In fact, some projects have covered a few chapters of the Quran (Dror, 2004), while others have published a domain model without the instances (Al-Khalifa et al., 2009).

Data preparation

After understanding the datasets, it is important to inspect the dataset format to consider whether the data need to be prepared before implementing the merging process. I investigated the formats and concluded that I would need to prepare the dataset by converting each dataset to a standard format (XML file). This phase makes merging the prepared datasets in the modelling phase a much more straightforward task.

Modelling

The purpose of merging the three datasets is to increase the data knowledge by organising the way that clients can use them. The model phase involved designing an appropriate solution to merge the three datasets in order to accomplish the project's objectives. Chapter Four discusses the solution design. In addition, to make it possible to use the dataset, a corpus tool needed to be specified. The aim of having a corpus tool is to allow users to browse the dataset with annotations and to make the resource available through the Internet.

Evaluation

The project produces three main outputs: first, the unified dataset, which merges the three datasets in a standard format (XML); second, the unified dataset ported to the corpus tool (Sketch Engine) so users can explore and search the dataset in an advanced level such as using Regular expressions; and third, an OWL ontology version of the unified dataset ported to Protégé, an open source ontology editor.

Stage one: The unified annotations XML dataset

To evaluate the unified annotations XML file, I used Xpath expressions to validate the merging process. The Xpath outputs were compared to the equivalent information in each selected dataset. This was during the design phase.

Since unifying the datasets would involve pre-processing the three datasets individually, I converted each dataset to an XML version. Thus, these XML files needed to be verified and evaluated. To accomplish that, I used again the Xpath expressions to query the XML file, then compared and validated the results with the actual datasets tool output.

Stage two: Evaluating the Sketch Engine corpora

After the annotations had been merged and the unified file ported to the Sketch Engine tool, some contributed typical users (Quranic researchers and Arabic language experts) evaluated the unified dataset using the Sketch Engine tool. The aim of their evaluation was mainly to find concordance, frequencies and thesaurus. According to dukes et al (2010), the Quranic Annotation corpus has made a verb and lemma concordance listing the words with their frequencies. I used his words as a foundation for the Sketch Engine tool. Identical results would show that the tool is perfect.

Stage three: Evaluating the unified OWL ontology through Protégé editor:

The evaluation was achieved by porting the unified Quranic ontology to Protégé editor and evaluating the OWL ontology by querying the Quranic words using SPARQL language. The evaluation was made by questioning the ontology and getting the answers. Then, the query results were validated by comparing them with previous known answers. Identical results would indicate that the OWL ontology is accurate.

Deployments:

Unifying the datasets is not the end of the project. The clients' feedback is vital in developing the dataset accuracy and presentation. The feedback involved Re-implementing the data mining process to add/change the solution design.

Deliverables:

At the end of the project, four software prototypes will have been developed based on two main functions.

The first function was to convert each dataset to a standard format. Since there are three datasets, then creating a single converter prototype was a hard task. Thus, each dataset had to have its own format converter.

The second function was the merging process. The fourth prototype merged the three datasets into one dataset. The output of the fourth prototype is the unified annotations dataset and the Sketch Engine corpus.

There was another planned prototype to convert the unified XML dataset to OWL ontology, but then this was accomplished by using Protégé XMLTab.

Project plan:

The first project plan was changed due the unexpected long prototypes implementation and background reading as well. The project direction was changed to address two main parts: the unified Quranic dataset and the unified Quranic ontology. Thus, more tasks were need such as ontology background research, finding an ontology editor and evaluating the ontology. The old project plan is in appendix B.

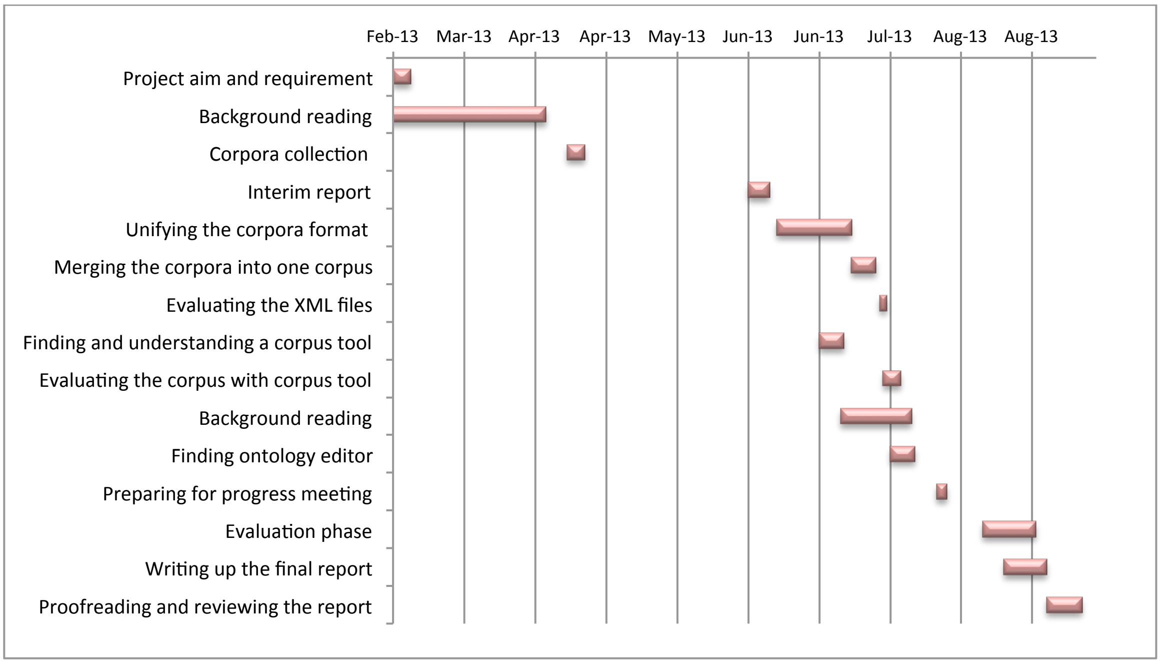


Figure (1.1): Project plan

Chapter 2: Background research

2.1 What is the Holy Quran?

The Holy Quran is the religious text of Islam. The Quranic text is divided into 114 chapters, and each chapter consists of a varying number of verses. A verse consists of a group of words. The Holy Quran is Prophet Mohammed's miracle and the most accurate resource of the Arabic language. Therefore, finding incorrect syntax in the Quran is unusual.

2.2 The current Quran-related projects

There are many existing Quran-related projects, each having different purposes and functions. However, all Quranic projects share the same Quranic texts but produce different annotations, such as annotations focusing on the morphological analyses of each word, while others focus on the semantic analyses for each verse. In addition, these projects have different data models, such as using OWL to describe the relations between Quranic vocabularies, while others use XML and HTML to present their projects' outputs. The following sections will provide more detail about these projects and annotations.

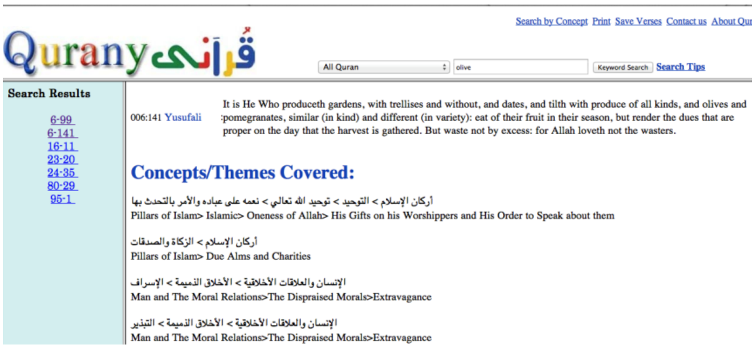
2.2.1 Semantic Opposition Ontology in the Holy Quran (SemQ)

SemQ is a framework that comprises many steps to produce the output. The main purpose of SemQ is to find the antonyms for an input verse. This project was developed by King Saud University in Saudi Arabia. There are two basic components in the SemQ framework: the ontology file and the SemQ tool (Al-Yahy et al., 2010). The ontology used in the framework is a domain ontology, or OWL model, which describes the relations between Quranic vocabularies without listing and indexing every occurrence of the Quranic words. The aim was to pass the verse's noun words to the ontology model via a tool to investigate the relations between any two nouns in the verse. However, the output of the framework depends on whether the input verse has an opposite noun associated with a certain degree of relationship, such as absolute and scalar. Nonetheless, the project addressed the problem of finding an accurate tool to discover the antonyms of Arabic words. Studying semantic opposition is considered a helpful resource in understanding a word's meaning by presenting the opposite meaning (Al-Khalifa et al., 2009).

Since finding the opposite words in the Quran requires a huge effort, the objective was reduced to presenting one concept feature (Time features). Basically, the domain shows the Arabic nouns associated with Time concepts in the Quran. The domain was restricted to the noun words describing Time features only (Al-Khalifa et al., 2009). Thus, it can be indicated that the SemQ is concerned with the word level only.

2.2.2 Qurany Concept search tool:

Qurany is a search-by-concept tool which was developed by the postgraduate student Noorhan Abbas at the University of Leeds as an MSc research project. The tool is available online http://www.comp.leeds.ac.uk/nora/html, Figure 2.2. The purpose of the project is to search for a keyword's concept. So, searching for a keyword leads to finding similar keywords in every Quranic verse, and then the search outputs lists verses containing the targeted keyword associated with many concepts.



Search Results

6-99
6-141
16-11
23-20
24-35
80-29
95-1

006:141 Yusufali It is He Who produceth gardens, with trellises and without, and dates, and tilth with produce of all kinds, and olives and :pomegranates, similar (in kind) and different (in variety): eat of their fruit in their season, but render the dues that are proper on the day that the harvest is gathered. But waste not by excess: for Allah loveth not the wasters.

Concepts/Themes Covered:

أركان الإسلام > التوحيد > توحيد الله تعالي > نعمه على عباده والأمر بالتحدث بها
Pillars of Islam> Islamic> Oneness of Allah> His Gifts on his Worshippers and His Order to Speak about them

أركان الإسلام > الزكاة والصدقات
Pillars of Islam> Due Alms and Charities

الإنسان والعلاقات الأخلاقية > الأخلاق الذميمة > الإسراف
Man and The Moral Relations>The Dispraised Morals>Extravagance

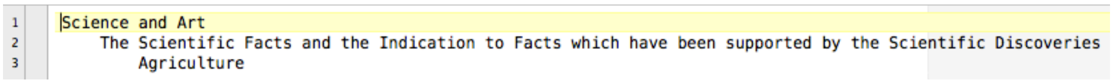
الإنسان والعلاقات الأخلاقية > الأخلاق الذميمة > التبذير
Man and The Moral Relations>The Dispraised Morals>Extravagance

Figure 2.2: Qurany tool website

For example, in a search for the English keyword (olive), which is mentioned in the Quran, the search tool will retrieve every verse that has the keyword (olive) and any words associated with the lists of concepts. One of these concepts is the Agriculture concept, which is related to the olive concept. In addition, the concepts list has a hierarchical structure, which means the concepts listed as one line have angle brackets between them:

Science and Art> The Scientific Facts and the Indication to Facts which have been supported by the Scientific Discoveries>Agriculture

This line is read as follows:



```
Science and Art
    The Scientific Facts and the Indication to Facts which have been supported by the Scientific Discoveries
        Agriculture
```

Moreover, the input keyword is lemmatized to increase the similarity measure. Then, the tool finds the verses that match the lemmatized keyword. To increase the accuracy of the similarity, eight English translations have been attached for each verse beside the Arabic verse text. Using these translations, the system performance for extracting the (English) keywords is increased to 87% (Abbas, 2009) but the search for Arabic keywords has a low performance, showing some unwanted results. Nonetheless, tagging the Quranic words with their morphological annotations, such as the root and lemma, would increase the retrieval performance.

Qurany has two module tools; the first module is the concept search tool as mentioned earlier, and the second module is the Tree of concepts browser, which presents hierarchal predefined Quranic topic groups. Each verse in the Quran is classified to one or more of these predefined concepts. The

Quranic concepts were taken from Mushaf Al Tajweed's book and are considered as the gold standard for evaluating the system classification. In Mushaf Al Tajweed, each topic has a list of all common verses in the Quran (Abbas et al., 2009).

Qurany was developed to deal with the semantic retrieval of the verses' keywords. So, each verse has been annotated with many semantic tags to make it possible to search for similar verses sharing the same concepts. Moreover, some verses were untagged with semantics due to implantation issues (Abbas, 2009). However, having verses annotated with their semantic tags helps in finding relationships between verses in general and words in principle as well.

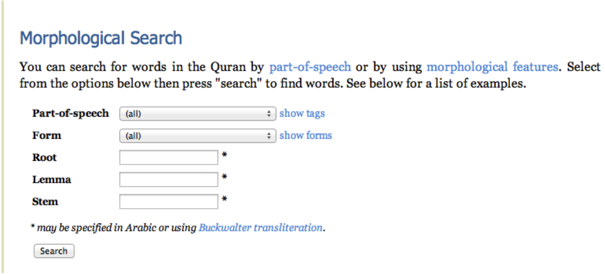
Finally, the verse-concept files are stored in HTML format. The usefulness of the Concept tree is in finding all verses under a topic easily. The results will be considered a valuable resource for Quranic students and linguistic researchers.

The Qurany project has some limitations in retrieving Arabic keywords, and furthermore, does not present the morphological and syntactic analyses for a target keyword. Thus, I argue that merging Qurany with the Quranic Arabic Corpus will increase the retrieval performance for Arabic keywords.

2.2.3 Quranic Arabic Corpus (QAC)

The Quranic Arabic Corpus is published through the web and available at http://corpus.quran.com. The website has many services such as a Quranic dictionary, an English translation of the Quran, and the syntax Treebank. These services use the Quran Annotation Corpus as a dataset.

The Quranic dictionary has many advantages, such as the verb concordance, lemma concordance, and morphology search tool. The verb and lemma concordances list the most frequent verbs in the Quran and the most frequent lemma with their frequencies while the morphological search tool aims to retrieve the morphological annotations for the input keyword. As Figure 2.4 shows, the tool makes it possible to search for part-of-speech tags, such as searching for all adverbs in the Quran. Other options are form, root, lemma and stem. The search tool accepts Arabic keywords and the Buckwalter translation as well (Dukes et al., 2012).



Morphological Search

You can search for words in the Quran by part-of-speech or by using morphological features. Select from the options below then press "search" to find words. See below for a list of examples.

Part-of-speech (all)

Form (all)

Root *

Lemma *

Stem *

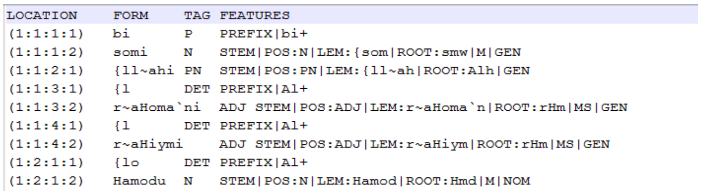
* may be specified in Arabic or using Buckwalter transliteration.

Figure (2.4): The QAC morphological search tool.

In QAC, each word in the Quran is given its morphological and syntactical annotations. However, the project lacks the addition of the semantic meaning to each Quranic word. This is because the aim was to analyse each Quranic word morphologically and syntactically.

The annotation tag set is the syntactic and morphological features of the Arabic language, such as POS tag, Stem/prefix/suffix feature, Gender, Person, Number, Aspect, Case, Mood, Verb form, Prefix features and other grammatical features.

The corpus format is a text format with tab separation. Each segment or word appears on a line beside the morphological annotations and its location information. The location is presented between parentheses in the pattern (x:y:z:w), where x is the chapter number, y is the verse number in chapter x, z is the segment number in verse y, and w is the token number in word z. As shown in Figure (2.5) not all words have more than one token. If a word has many tokens, such as words with the determiner '{l' , a syntactic analysis has to be applied in order to find the appropriate tag annotation for each segment (Dukes et al., 2011).

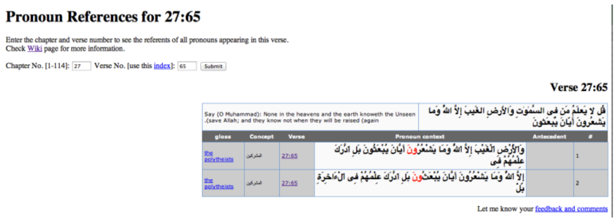


```
LOCATION     FORM     TAG  FEATURES
(1:1:1:1)    bi       P    PREFIX|bi+
(1:1:1:2)    somi     N    STEM|POS:N|LEM:{som|ROOT:smw|M|GEN
(1:1:2:1)    {ll~ahi  PN   STEM|POS:PN|LEM:{ll~ah|ROOT:Alh|GEN
(1:1:3:1)    {l       DET  PREFIX|Al+
(1:1:3:2)    r~aHoma`ni   ADJ STEM|POS:ADJ|LEM:r~aHoma`n|ROOT:rHm|MS|GEN
(1:1:4:1)    {l       DET  PREFIX|Al+
(1:1:4:2)    r~aHiymi     ADJ STEM|POS:ADJ|LEM:r~aHiym|ROOT:rHm|MS|GEN
(1:2:1:1)    {lo      DET  PREFIX|Al+
(1:2:1:2)    Hamodu   N    STEM|POS:N|LEM:Hamod|ROOT:Hmd|M|NOM
```

Figure (2.5): The morphological annotation in the Quran Annotation Corpus

2.2.4 QurAna

QurAna is a search tool for the Text mining of the Quran project which can be found on the website http://www.textminingthequran.com/apps/pron.php . This tool aims to show the pronoun references by entering the verse and chapter numbers. Thus, this system retrieves the pronoun references by entering an index consisting of chapter and verse indices, but cannot retrieve the verse by entering a keyword. The user must know beforehand which verse s/he is going to search for (see Figure 2.6).



Pronoun References for 27:65

Enter the chapter and verse number to see the referents of all pronouns appearing in this verse.
Check Wiki page for more information.

Chapter No. [1-114]: 27 Verse No. [use this index]: 65 Submit

Verse 27:65

Say (O Muhammad): None in the heavens and the earth knoweth the Unseen .(save Allah; and they know not when they will be raised (again	قُل لا يَعلَمُ مَن فى السَّمٰوٰتِ وَالأَرضِ الغَيبَ إِلَّا اللَّهُ وَما يَشعُرونَ أَيّانَ يُبعَثونَ

gloss	Concept	Verse	Pronoun context	Antecedent	#
the polytheists	المشركين	27:65	وَالأَرضِ الغَيبَ إِلَّا اللَّهُ وَما يَشعُرونَ أَيّانَ يُبعَثونَ بَلِ ادّٰرَكَ عِلمُهُم فى		1
the polytheists	المشركين	27:65	إِلَّا اللَّهُ وَما يَشعُرونَ أَيّانَ يُبعَثونَ بَلِ ادّٰرَكَ عِلمُهُم فِى الءاخِرَةِ بَل		2

Let me know your feedback and comments

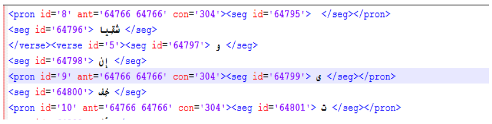
Figure (2.6): The pronoun references tool

The project's purpose is to tag each pronoun in the Quran with the reference concepts. For example, verse 5 in chapter 1 has the pronoun <yaka, which mainly refers to Allah. The discourse level in the corpus is word level and the project covers all chapters in the Quran (Sharaf et al., 2012).

As shown in Figure (2.7), the QurAna tool uses a dataset formatted with XML. Thus, each chapter has a <chapter> tag with id, each verse has a <verse> tag with id, and each verse segment has a <seg> tag with id. If a segment is a pronoun, then the pronoun reference annotation is also applied. The pronoun annotation theme contains three main parts.

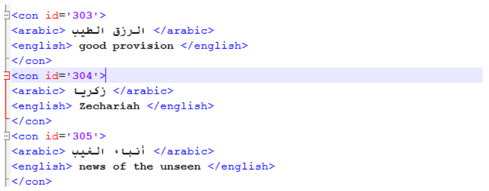
The corresponding concept in the concept XML file will be referred to as <con> with the reference id. The tag <pron> is the pronoun index in its chapter; <ant> is the segment id where the antecedent is found in any chapters.

1- The pronoun id <pron id='X'>: the pronoun index in a chapter.
2- The concept id: <con id='X'>: the tool uses the concept file to find the corresponding reference's concept by using the id. The concept id retrieves the English and Arabic concepts, (see Figure 2.8).
3- The antecedent id <ant id='X'>: this mainly refers to the reference position in the Quran, such as the reference concept in <yaka is Allah, which has the segment id 11. Returning to the segment id, we find the same reference concept 'Allah' (Sharaf, 2012).



```
<pron id='8' ant='64766 64766' con='304'><seg id='64795'>  </seg></pron>
<seg id='64796'> شقيا </seg>
</verse><verse id='5'><seg id='64797'> و </seg>
<seg id='64798'> إن </seg>
<pron id='9' ant='64766 64766' con='304'><seg id='64799'> ى </seg></pron>
<seg id='64800'> خف </seg>
<pron id='10' ant='64766 64766' con='304'><seg id='64801'> ت </seg></pron>
```

Figure 2.7: Part of QurAna corpus file



```
<con id='303'>
<arabic> الرزق الطيب </arabic>
<english> good provision </english>
</con>
<con id='304'>
<arabic> زكريا </arabic>
<english> Zechariah </english>
</con>
<con id='305'>
<arabic> أنباء الغيب </arabic>
<english> news of the unseen </english>
</con>
```

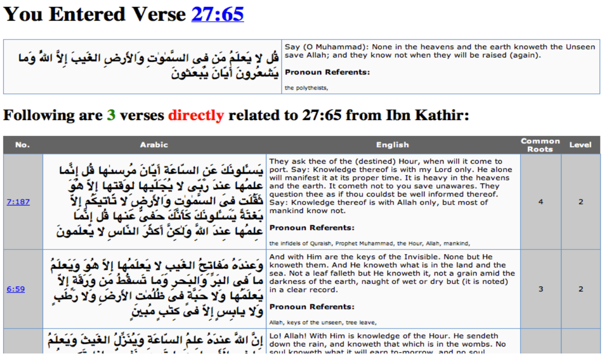
Figure 2.8: Concept file in QurAna.

By examining the difference between the antecedent id and the reference concept, we infer that both of them represent the same information. In addition, the pronoun reference tool shows the antecedent id by replacing the segment id with the referred segment text.

2.2.5 QurSim

QurSim is another search tool for the Text mining of the Quran project. This tool retrieves similar verses by entering any verse id. The search output lists any similar verses as shown in Figure 2.9. This tool focuses on finding semantically related pairs of verse in the Holy Quran. The relatedness is based on 'Tafsir', a book written by the Quranic scholar Ibn Kathir (Sharaf, 2012).

This tool could be a valuable resource in understanding and discovering similar verses, which might have common patterns. In addition, the discourse level in the QurSim corpus is verse level, where each verse has been tagged with the most similar verse.



You Entered Verse 27:65

قُل لا يَعلَمُ مَن فِى السَّمٰوٰتِ وَالأَرضِ الغَيبَ إِلَّا اللَّهُ وَما يَشعُرونَ أَيّانَ يُبعَثونَ	Say (O Muhammad): None in the heavens and the earth knoweth the Unseen save Allah; and they know not when they will be raised (again). **Pronoun Referents:** the polytheists,

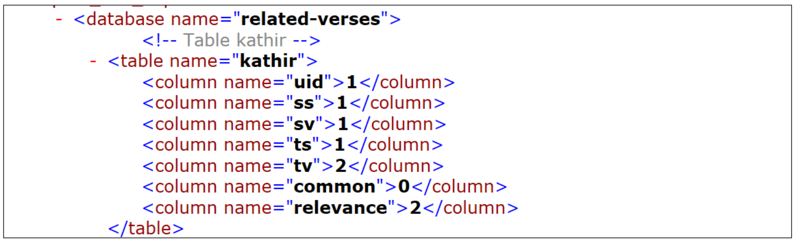
Following are 3 verses directly related to 27:65 from Ibn Kathir:

No.	Arabic	English	Common Roots	Level
7:187	يَسـَٔلونَكَ عَنِ السّاعَةِ أَيّانَ مُرسىٰها قُل إِنَّما عِلمُها عِندَ رَبّى لا يُجَلّيها لِوَقتِها إِلّا هُوَ ثَقُلَت فِى السَّمٰوٰتِ وَالأَرضِ لا تَأتيكُم إِلّا بَغتَةً يَسـَٔلونَكَ كَأَنَّكَ حَفِىٌّ عَنها قُل إِنَّما عِلمُها عِندَ اللَّهِ وَلٰكِنَّ أَكثَرَ النّاسِ لا يَعلَمونَ	They ask thee of the (destined) Hour, when will it come to port. Say: Knowledge thereof is with my Lord only. He alone will manifest it at its proper time. It is heavy in the heavens and the earth. It cometh not to you save unawares. They question thee as if thou couldst be well informed thereof. Say: Knowledge thereof is with Allah only, but most of mankind know not. **Pronoun Referents:** the infidels of Quraish, Prophet Muhammad, the Hour, Allah, mankind,	4	2
6:59	وَعِندَهُ مَفاتِحُ الغَيبِ لا يَعلَمُها إِلّا هُوَ وَيَعلَمُ ما فِى البَرِّ وَالبَحرِ وَما تَسقُطُ مِن وَرَقَةٍ إِلّا يَعلَمُها وَلا حَبَّةٍ فى ظُلُمٰتِ الأَرضِ وَلا رَطبٍ وَلا يابِسٍ إِلّا فى كِتٰبٍ مُبينٍ	And with Him are the keys of the Invisible. None but He knoweth them. And He knoweth what is in the land and the sea. Not a leaf falleth but He knoweth it, not a grain amid the darkness of the earth, naught of wet or dry but (it is noted) in a clear record. **Pronoun Referents:** Allah, keys of the unseen, tree leave,	3	2
	إِنَّ اللَّهَ عِندَهُ عِلمُ السّاعَةِ وَيُنَزِّلُ الغَيثَ وَيَعلَمُ	Lo! Allah! With Him is knowledge of the Hour. He sendeth down the rain, and knoweth that which is in the wombs. No soul knoweth what it will earn to-morrow, and no soul		

Figure (2.9): QurSim outputs

In Tafsir Ibn Kathir, the total number of related verses in the Quran is 7,679 pairs. Many verses sharing the same word root have an understandable relatedness. On the other hand, 883 verses have an ambiguous relationship and therefore are flagged as 'not obvious'. The total number of obviously related verses is 6,796. Thus, the tool has flagged each pair with their degree of relevance (Sharaf, 2012).

The dataset format used in the QurSim tool is formatted by XML as shown in Figure (2.10). Each verse has seven XML tags as follows:<uid> is the incremental index of verse; <ss> the source chapter; <sv> the source verse; <ts> the target chapter; <tv> the target verse; <common> is the number of shared root words between the pair;; and <relevance> is the degree of similarity, which takes a value of 0 or 1 or 2. A value of 0 means the verses are related but, to understand the relation, you need to read the explanation of the two verses in Tafsir Ibn Kathir. A value of 1 means the verses are related, but still inappropriate for training purpose. A value of 2 means the verses are very similar and are ideal for training purposes (Sharaf, 2012).



```
- <database name="related-verses">
        <!-- Table kathir -->
    - <table name="kathir">
        <column name="uid">1</column>
        <column name="ss">1</column>
        <column name="sv">1</column>
        <column name="ts">1</column>
        <column name="tv">2</column>
        <column name="common">0</column>
        <column name="relevance">2</column>
    </table>
```

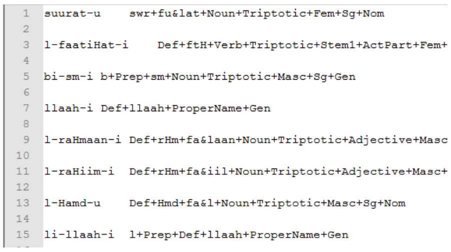
Figure 2.10 : QurSim dataset format.

2.2.6 Morphological Analysis of the Quran

A morphological analysis of the Quran project has been developed by the University of Haifa in Israel. The project has two main parts; the first part is the morphological annotated corpus and the second is the GUI search tool, mainly to retrieve Quranic words with specific morphological features (Dror et al., 2004).

The Arabic language has complicated morphological analyses. Each Arabic root can have many derived word forms. Therefore, a simple retrieval system will fail to find Arabic word occurrences in a corpus, and the result will be restricted to the same lemma form in the source word. The system of morphological analyses of the Quran has solved this issue by exhaustively analysing each Arabic word. So, the system will retrieve all the derived forms of that lemma (Dror et al., 2004).

The dataset format is a plain text format with tab separation (see Figure 2.11). This type of dataset is called an annotated corpus, where each word has its own annotation. The purpose of this corpus is to be run under the SQL server or GUI interface in order to retrieve the Quranic word. So, the main function of the corpus is to retrieve as much word information as possible in the Quran.



```
suurat-u    swr+fu&lat+Noun+Triptotic+Fem+Sg+Nom

l-faatiHat-i    Def+ftH+Verb+Triptotic+Stem1+ActPart+Fem+

bi-sm-i b+Prep+sm+Noun+Triptotic+Masc+Sg+Gen

llaah-i Def+llaah+ProperName+Gen

l-raHmaan-i Def+rHm+fa&laan+Noun+Triptotic+Adjective+Masc

l-raHiim-i  Def+rHm+fa&iil+Noun+Triptotic+Adjective+Masc+

l-Hamd-u    Def+Hmd+fa&l+Noun+Triptotic+Masc+Sg+Nom

li-llaah-i  l+Prep+Def+llaah+ProperName+Gen
```

Figure (2.11): Part of the dataset format

As Figure (2.11) shows, the file is simply a text file, and each line has a single root word with its affix and suffix separated by the '-' character. In contrast, the morphological annotations are shown beside each word. The annotations might have the following structure:

Root + Pattern + Part of speech + Case marking + gender + number + case.

The '+' character is used as a separator between the structural analyses.

The purpose of this project was to retrieve the Quranic words associated with the morphological annotations. The retrieving performance guarantees that each word sharing the same root will be shown in the search outputs (Dror et al., 2004).

Unfortunately, this project came to a halt due to project team issues, and only a few chapters were covered in the project.

2.3 Choosing the project datasets:

This project aims to merge three datasets. To choose the dataset, we had to consider many criteria, such as the chapter's coverage, and the project's completeness and function, as shown in Table 2.1.

#	Dataset	Function	Format	Discourse Level	Corpus coverage	Completeness
1	SemQ ontology	The opposite semantic ontology	OWL ontology	Word	Only one word feature has been covered.	No
2	QurSim dataset	The relatedness verses	XML dataset	Verse	All chapters	Yes
3	Qurany dataset	Find the verse concept	HTML dataset	Verse	All chapters	Yes
4	QurAna dataset	Antecedence assignment for each pronoun	XML dataset	Word	All chapters	Yes

#	Dataset	Function	Format	Discourse Level	Corpus coverage	Completeness
5	QAC Corpus	Morphological analyses	Plain text corpus	Word	All chapters	Yes
6	Morphological Analyses of the Quran corpus	Comprehensive morphological analyses of Quranic words.	Plain text corpus	Word	A few chapters	No

Table (2.1): summary of the existing Quranic datasets

SemQ ontology: To consider the SemQ ontology as one of the targeted datasets, I considered the following:

1- SemQ ontology is an OWL model, which does not list all the Quranic words sharing the same relations. For instance, the class (TimeFeature1) should be associated with certain instances such as the instance word_id_1, which is located in chapter id 1 and verse id 1.

2- SemQ addresses only one feature type (Time feature). Therefore, I could not rely on the ontology to express any opposite occurrences.

For the above reasons, it was concluded that SemQ has a different philosophy and cannot be proposed in our unifying annotations ontologies.

Morphological Analyses of the Quran corpus: This project was not completed; therefore, it was excluded from the targeted datasets because I was seeking to cover all Quranic chapters.

Quranic Arabic Corpus (QAC): This is one of the ideal datasets. The project was completed and covers all Quranic chapters. This corpus has an obvious structure and can be pre-processed for merging with other datasets.

The aim of choosing this corpus was to develop the morphological analysis search tool by adding more annotations to it, such as the pronoun reference. In addition, I aimed to add all the morphological and syntactic features that were not listed in the morphological search tool, such as retrieving all words in the Quran having the feminine gender.

The pronoun reference search tool (QurAna): This is another of the ideal datasets. The dataset format has an obvious structure; therefore, merging it with the QAC dataset would develop the output by showing the QAC and QurAna annotations.

In fact, there is a previous study which merged the QurAna and QAC datasets. This merge was done by the undergraduate student Zainb Alqassem in her final year project. The integrated file was uploaded to a linguistic tool ‘Sketch Engine’ on the web. By using the Sketch Engine tool, many users around the world will be able to use the integrated file easily. Her project produced four corpora files:

an Arabic language vowelled corpus , an Arabic language unvowelled corpus, a Latin alphabetic vowelled corpus, and a Latin alphabetic unvowelled corpus (see Figure 2.11).
In fact, the merged file has some limitations, such as the lack of any search for the words according to their grammatical features, such as the morphological search tool does partially. For example, it is not possible to retrieve a specific type of grammar annotations, such as the feminine nouns in the Alqassem project. More limitations regarding her project can be found in Chapter Four. Thus, I needed to re-merge QAC and QurAna to add the annotations labels.

Qurany search tool dataset: Qurany is a verse-based dataset. Since we do not have semantic concepts at the word level and the available semantic concepts are at the verse level, I aimed in this project to add a diversity of language analyses annotations. Since we have the morphological and syntax annotations (QAC and QurAna), I chose Qurany to add the semantic analyses annotations.

QurSim: QurSim and Qurany share the same objective, that is, finding similar verses. However, Qurany adds the concepts to each verse. Thus, choosing one of them would be enough to accomplish the third dataset. There were more advantages to adding Qurany than to adding Qursim. In fact, Qurany adds the concepts which might be inherited to the word level, but this inheritance can be seen as a disadvantage because not every word in the verse can express the same concept.

The selected datasets: Thus, the target datasets were the Quranic Arabic Corpus (QAC), QurAna and Qurany datasets. The challenge of this merging is the different discourse levels each; for example, the QAC and Qurana datasets have a word level while Qurany has a verse level.

Chapter 3: Ontology and Corpus Representations and Tools

This chapter discusses the representation of datasets and ontology data models and investigates some frequently used corpus tools and ontology editors.

3.1 Ontology data model:

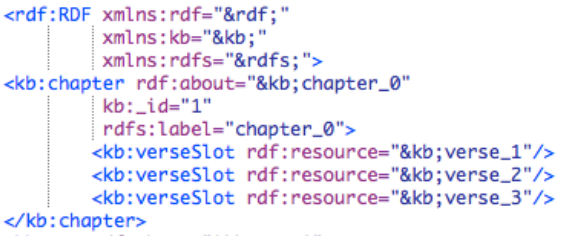
Ontology aims to represent a domain by defining its vocabularies and their relationships. According to Noy et al (2001), ontology can help researchers to share the common vocabulary in such a domain. Moreover, ontology has extendability and re-usability features, so it can be extended to cover more vocabularies by further researches. This kind of data model assumes that using the ontology in a search engine will increase the retrieval performance by finding other inferred instances.

Examples of data models encoding the semantic are as follows:

3.1.1 RDF model:

This stands for Resource Description Framework (RDF). According to Lassila et al (1999), the syntax used in RDF is much closer to XML syntax. In addition, one version of the popular RFD model is the abstract model, which describes two related instances called a graph or triple. A triple has a subject and object nodes which are related components. The subject is an entity and the object might be another entity or a value (Lassila et al, 1999).

An example of Quranic RDF triples produced by Protégé editor is shown in the following figure(3.1):



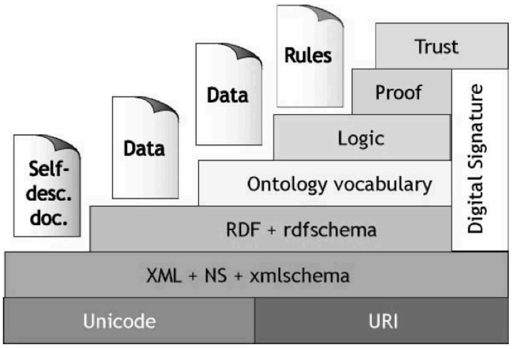
```
<rdf:RDF xmlns:rdf="&rdf;"
         xmlns:kb="&kb;"
         xmlns:rdfs="&rdfs;">
<kb:chapter rdf:about="&kb;chapter_0"
         kb:_id="1"
         rdfs:label="chapter_0">
        <kb:verseSlot rdf:resource="&kb;verse_1"/>
        <kb:verseSlot rdf:resource="&kb;verse_2"/>
        <kb:verseSlot rdf:resource="&kb;verse_3"/>
</kb:chapter>
```

Figure(3.1): An example of RDF triples.

The RDF model has limitations in describing the relationship feature between subject and object. Such transitive and symmetric relations cannot be represented in RDF. The relationship types have the ability to infer more knowledge.

3.1.2 Web Ontology Language (OWL) model:

According to Antoniou (2004), the semantic web layers in figure (3.2) indicates that Ontology Web Language (OWL) is built on top of the RDF model.



Figure(3.2): The semantic web layers; adopted **from Antoniou (2004)**

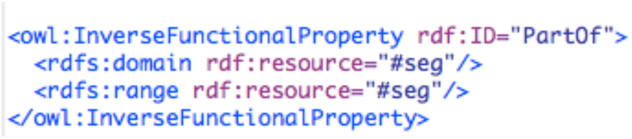
Moreover, OWL solved some RFD weaknesses such as adding the features types to the relationships. Therefore, OWL adds the semantic to the data by defining the relation type. There are lots of relation types and there are relation types specific to OWL such as the following:

- Functional Property:

  If a relationship between an individual A and literal B is Functional Property, this means each individual must have at least one relation to B. For example, in the unified uranic dataset every segment must have Functional Property to POS literal, because every segment must have a POS tag.

- Inverse Functional Property:

  If two individuals A and B have an inverse relation R, it means that this is a unique relationship and B cannot be shared with other individuals. For example, a verse id has an inverse Functional Property in relation to its segments. So segment id 5 is the only feature with a relation with verse id 1.

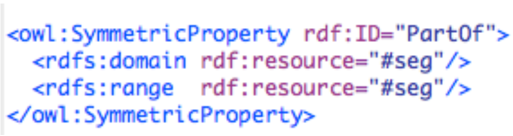


```
<owl:InverseFunctionalProperty rdf:ID="PartOf">
  <rdfs:domain rdf:resource="#seg"/>
  <rdfs:range rdf:resource="#seg"/>
</owl:InverseFunctionalProperty>
```

Figure (3.3): The inverse Functional Property

- Symmetric Property:

  If A and B have a symmetric property relationship, then this means that A infers B and B infers A. A and B together is literally an instance of the one class. However, in this study's unified dataset there is no example of this relationship. However, assume that segment id 1 and 2 represent a compound segment which forms a symmetric relationship with part of such as the following figure:

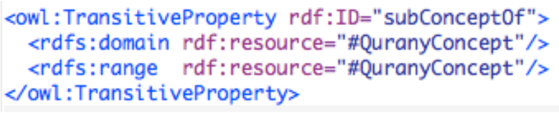


```
<owl:SymmetricProperty rdf:ID="PartOf">
  <rdfs:domain rdf:resource="#seg"/>
  <rdfs:range  rdf:resource="#seg"/>
</owl:SymmetricProperty>
```

Figure (3.4): the symmetric relation

- Transitive Property:

  If A infers B and B infers C, then A and C individually have a transitive relationship. An example is if Qurany Concept Pillars of Islam infers Islamic, and Islamic infers Oneness of Allah, then Pillars of Islam infers the Oneness of Allah. Therefore, Pillars of Islam and Oneness of Allah have a transitive relationship (figure 3.5).



```
<owl:TransitiveProperty rdf:ID="subConceptOf">
  <rdfs:domain rdf:resource="#QuranyConcept"/>
  <rdfs:range  rdf:resource="#QuranyConcept"/>
</owl:TransitiveProperty>
```

Figure (3.5): the transitive property

3.2 Ontology editors and tools:

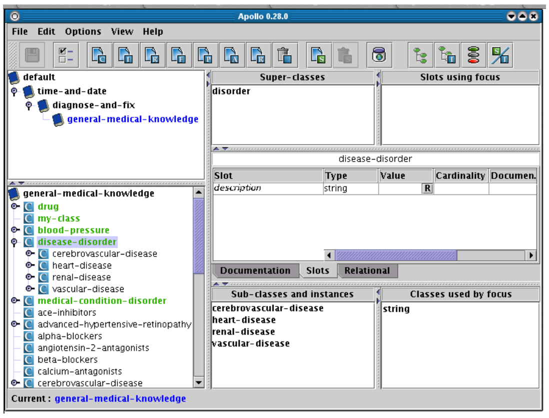
The ontology Editor is defined as editors enabling the knowledge modelling which might have the following features:

- Editor has a friendly graphical user interface.
- It can create ontology by writing the classes and instances with relations.
- Some editors can generate ontology from XML files and spreadsheets.
- It can explore the ontology's class, instances and relations.
- Some editors can present the classes with relations as a graph view.
- A reasoning engine which applies the properties of the instances and obtains new knowledge.
- Can query the ontology by using a query language such as SPARQL.

There are lots of current ontology editors, which cannot all be listed in this project. According to Alatrish (2012), Apollo and Protégé editors are the most frequently used in designing and processing ontologies. He stated that Apollo and Protégé have significant features which are listed below.

3.2.1 Apollo:

Apollo is a free ontology, editor-developed by the Knowledge Media Institute of The Open University and it is available through http://apollo.open.ac.uk. This editor has many significant features, such as the robust consistency engine, which ensure that individuals and relations are consistent. In addition, Apollo has the ontologies inheritance feature, so an ontology can inherit other external ontology vocabularies, because they are a single ontology. However, Apollo lacks supporting importing XML files, an important criterion in choosing the project ontology editor. Moreover, Apollo does not support the graphical view of ontology, which might improve understanding of the ontology design (Alatrish, 2012).



Slot	Type	Value	Cardinality	Documen.
description	string	R		

Figure(3.6): Apollo editor interface (adopted from Apollo website http://apollo.open.ac.uk/index.html).

3.2.2 Protégé:

According to Alani et al (2005), Protégé has been described as "the killer app". Protégé has many significant features that make it a distinctive ontology editor. Protégé is a free application developed by Stanford University and it is available through http://protege.stanford.edu. Protégé has a friendly GUI that enables browsing of the ontology in hierarchal view.

Protégé has the ability to add an external plug-in to the application, enhancing its features. Some plug-ins need to be installed by the user and some are built in features of the full version of Protégé. The purpose of a plug-in is to add features such as visioning and merging ontologies, importing XML files and many more features.

Jambalaya is a graphical viewer plug-in used by Protégé. It presents the ontology design in a dynamic graphical view thereby visualizing ontology summaries by the main concepts /classes with relations. In Jambalaya, classes are represented in square shapes and by clicking inside a square it shows the class's instances. The relations between classes are shown by red arrows. The representation shows the type of relations between classes representing the property type, such as the transitive property links between two un-successive classes.

Another important plug-in is the Protégé-OWL. It is a part of the Protégé full version. This plug-in enables the use of the ontology reasoning by inferring the relation property.

The most important feature in Protégé is the ability to import XML files. It accepts XML through the XML tab, which can be enabled through configuration options. Then, the imported file is mapped to an ontology format such as OWL or RDF. This facility does not accept XML schema and accepts only (.xml) files. Unfortunately, a research study examining the efficiency of importing XML file and mapping to OWL ontology could not be found, but it seems like a novel approach.

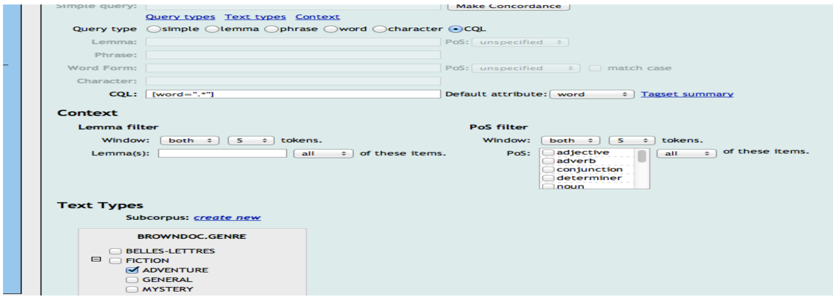
Protégé has an efficient ability to manipulate ontologies of large size, which might be helpful in dealing with the Unified Quranic ontology.

An example of using Protégé in developing Quranic ontology is the ontology based semantic search in the Holy Quran project developed by Khan et al (2013). This project aims to build a question-and-answer knowledge base. Queries can be constructed using SPARQL to retrieve instances having an inferred relationship property.

Because of the above features, Protégé has been chosen as the ideal editor for this study's XML unified dataset. Using protégé's facilities, an OWL ontology version of this XML file can be generated.

3.3 Corpus tools:

Undoubtedly, having a corpus without a tool enabling an explore function will not obtain the aim of this project. Thus, a corpus tool must be determined. There are many corpus tools that met the project requirement and can be used. Alqassem (2013), in her final year project, examined many corpus tools such as the aConCorde, IntelliText and Sketch engine. But, Sketch Engine tool, a recent online corpus tool which has many significant features. Another corpus using Sketch engine is the Arabic massive corpus arTenTen (Habash et al, 2013). However, it is argued that Sketch engine has a significant analyses features and it is able to operate the Arabic text. It has facilities enabling the manipulation of Arabic text such as building the concordance and Thesaurus. But Sketch Engine has extraordinary features that make it distinctive from other tools. This feature is the Corpus Query Language CQL, which is involved in querying the corpus by using regular expressions. CQL is considered as an advanced tool feature. Using CQL, words with specific grammar features or genre can be retrieved, see figure (3.7) (3.8).



Simple query: Make Concordance

Query types Text types Context

Query type ○simple ○lemma ○phrase ○word ○character ●CQL

Lemma: PoS: unspecified

Phrase:

Word Form: PoS: unspecified □ match case

Character:

CQL: [word=".*"] Default attribute: word Tagset summary

Context

Lemma filter

Window: both 5 tokens.

Lemma(s): all of these items.

PoS filter

Window: both 5 tokens.

PoS: □ adjective □ adverb □ conjunction □ determiner □ noun all of these items.

Text Types

Subcorpus: *create new*

BROWNDOC.GENRE

□ BELLES-LETTRES

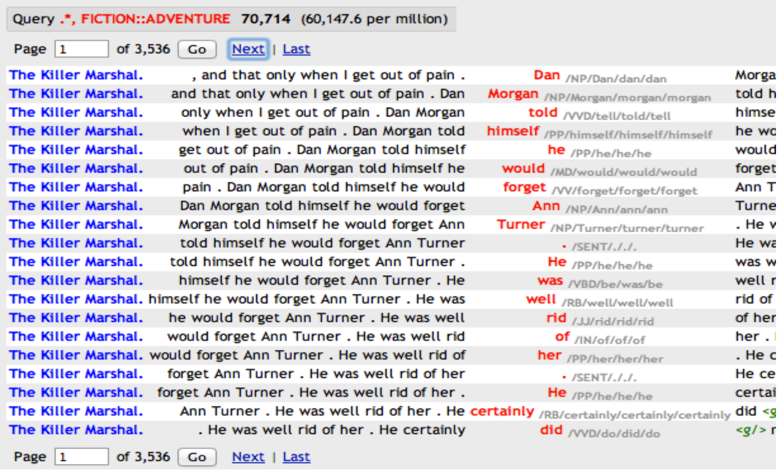
⊟ □ FICTION

☑ ADVENTURE

□ GENERAL

□ MYSTERY

Figure(3.7): Sketch engine Search options in Brown Corpus.



Query .*, FICTION::ADVENTURE 70,714 (60,147.6 per million)

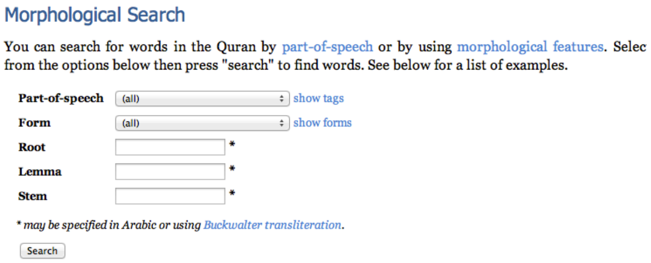
Page 1 of 3,536 Go Next | Last

The Killer Marshal.	, and that only when I get out of pain .	Dan /NP/Dan/dan/dan	Morga
The Killer Marshal.	and that only when I get out of pain . Dan	Morgan /NP/Morgan/morgan/morgan	told h
The Killer Marshal.	only when I get out of pain . Dan Morgan	told /VVD/tell/told/tell	himse
The Killer Marshal.	when I get out of pain . Dan Morgan told	himself /PP/himself/himself/himself	he wo
The Killer Marshal.	get out of pain . Dan Morgan told himself	he /PP/he/he/he	would
The Killer Marshal.	out of pain . Dan Morgan told himself he	would /MD/would/would/would	forget
The Killer Marshal.	pain . Dan Morgan told himself he would	forget /VV/forget/forget/forget	Ann T
The Killer Marshal.	Dan Morgan told himself he would forget	Ann /NP/Ann/ann/ann	Turne
The Killer Marshal.	Morgan told himself he would forget Ann	Turner /NP/Turner/turner/turner	. He v
The Killer Marshal.	told himself he would forget Ann Turner	. /SENT/././.	He wa
The Killer Marshal.	told himself he would forget Ann Turner .	He /PP/he/he/he	was w
The Killer Marshal.	himself he would forget Ann Turner . He	was /VBD/be/was/be	well r
The Killer Marshal.	himself he would forget Ann Turner . He was	well /RB/well/well/well	rid of
The Killer Marshal.	he would forget Ann Turner . He was well	rid /JJ/rid/rid/rid	of her
The Killer Marshal.	would forget Ann Turner . He was well rid	of /IN/of/of/of	her .
The Killer Marshal.	would forget Ann Turner . He was well rid of	her /PP/her/her/her	. He c
The Killer Marshal.	forget Ann Turner . He was well rid of her	. /SENT/././.	He ce
The Killer Marshal.	forget Ann Turner . He was well rid of her .	He /PP/he/he/he	certai
The Killer Marshal.	Ann Turner . He was well rid of her . He	certainly /RB/certainly/certainly/certainly	did <g
The Killer Marshal.	. He was well rid of her . He certainly	did /VVD/do/did/do	<g/> r

Page 1 of 3,536 Go Next | Last

Figure(3.8): The concordance of all Adventure words in Brown Corpus in Sketch engine

Another Corpus tool dedicated for Quranic text is LAMP tool, available in http://corpus.quran.com. The tool mainly deals with two databases; the Quraic text database and the Quranic audio and visual database. The tool is able to search for morphological features on Quranic text (figure 3.9), show the syntactically parsing verses and visualize the initial Quranic ontology for Quranic nouns and pronouns (Dukes, 2012). However, the tool can do search only for morphological features and has limitations in search options as been described in chapter two. The tool is customized to Quran Arabic Corpus and therefore can't be adapted to be used in this project.



Morphological Search

You can search for words in the Quran by part-of-speech or by using morphological features. Selec from the options below then press "search" to find words. See below for a list of examples.

Part-of-speech	(all)	show tags
Form	(all)	show forms
Root		*
Lemma		*
Stem		*

* *may be specified in Arabic or using Buckwalter transliteration.*

Figure(3.9): Morphological search tool in Quran Corpus (dukes,2012).

3.4 Corpus representation:

A corpus can be defined as a bunch of text which can be plain text or text tagged with linguistics annotations such as POS tags (annotated corpus), the Brown corpus, or a structured text in XML syntax, such as Wikipedia XML Corpus by Denoyer et al (2006). Hence, the Unified Quranic Annotation dataset is considered to be an XML corpus.

3.4.1 Sketch engine representation (format):

The corpus in Sketch engine could be in plain text format; successive words in a txt file or an XML file, or an xml tag specifying sentences and words. In fact, there are many standard file formats that are accepted by Sketch Engine; tools such as PDF, MICROSOFT DOCUMENT, HTML, XML and TXT. However, of more interest is XML, used in this study's unified dataset. According to Sketch Engine wiki, complex XML files such as schema will not be processed properly, but simple XML files will be accepted and processed.

Chapter 4: Design the Solution

This chapter discusses the most suitable and reasonable design and format for unifying different Quranic datasets. As we have different file formats, and we want to merge them in one file, we need to unify each file format to make merging them much easier.

The first step in designing the solution is to have the selected datasets in a unified format or to preprocess the datasets.

4.1 The design aims:

There are many formats that can be used in unifying the datasets, such as XML, CSV (comma separated value) and txt. However, finding a prober format is a vital decision. So, it is crucial to consider the main goals of the design:

1- to have unified datasets to be used by a corpus exploring system (sketch engine).

2- for, the unified datasets to be mapped at the same time to the ontology format (OWL ontology).

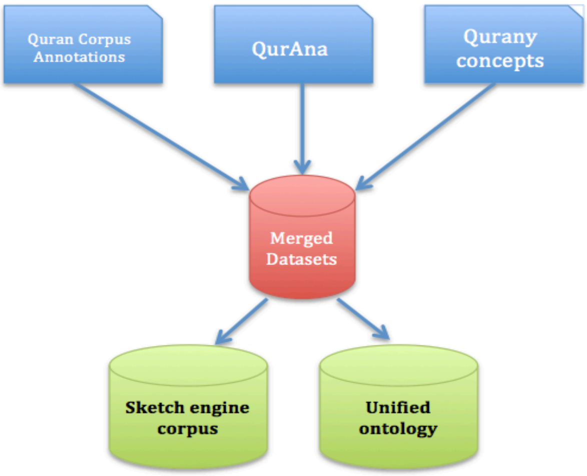


Figure (4.1): The aim of merging the three datasets

Sketch engine is a corpus exploring system which enables users to extract information and summarize it efficiently. Sketch engine uses XML to define the corpus structure and hierarchy (Kilgarriff et al., 2004). However, it does not accept the files format such as OWL or RDF. In addition, there are many available approaches in mapping XML to the OWL ontology (see the Ontology chapter).

However, XML is a common file format in many corpus tools and would be considered as a proper structure to accomplish our goals. Moreover, comparing XML to CSV, XML has the ability to describe the hierarchy and relationships in an obvious and readable structure.

Regarding the XML complexity level, Sketch engine does not accept complex XML files, such as the XML schema (Kilgarriff et al., 2004). So we have to restrict the XML complexity to a simple XML dataset by describing the data without its constraints and data type definitions as the schema describes.

The eXtensible Markup Language provides robust information storage. Its importance comes from its ability to store data with its description or metadata. An obvious feature of XML is the ability to describe the hierarchal data, as will be seen in the Qurany concepts files; each concept is in a hierarchy structure. Therefore, XML can describe these concepts in a multilevel structure.

Furthermore, XML is machine- and human-readable. Anyone who reads the XML file can initially understand the main structure, but to retrieve specific content, the Xpath technique can be used. Another advantage of XML is that it is an extensible file, which means it is applicable to extend the dataset with more annotations (Fawcett et al., 2012).

Perhaps the most serious disadvantage of the final unified XML dataset is the increase in the file size. Since each annotation is described by an attribute, then this adds more size to the file (Fawcett et al., 2012). However, the final output has a size of 20MB and can be ported to Sketch engine successfully.

4.2 Preprocessing the data:

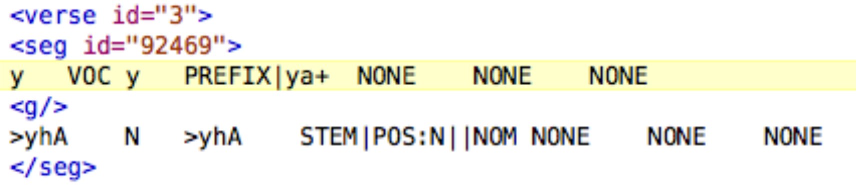
Alqassem merged file:

By looking at the available datasets, one merged dataset was done by Zainab Alqassem in her final year project at the University of Leeds. She merged two datasets, namely, Quranic Arabic Corpus (QAC) and QurAna. Both datasets have syntactic and morphological annotations per words. Alqassem merged them and ported the merged file to Sketch engine. However, this project "A unified Quranic Annotations and ontology" aims to extend the last merged dataset by Alqassem by merging one more dataset (Qurany Concept) with the previous two. Therefore, I had to investigate her merged dataset and decide if the merged file would be consistent with the third dataset (Qurany Concept). Consequently, I came with following problems:

Alqassem used a combination of XML structure and vertical structure to define the word attributes as the Sketch engine corpus format does. In the vertical line, each segment has been associated with its POS tag, lemma, QAC features, and QurAna features. QurAna is concerned with annotating each pronoun in the Quran Annotations corpus. In other words, any pronoun should have two kinds of annotations: a QAC feature and a QurAna feature. Other kinds of words (verb, adjective, etc.) do not have a particular annotation in QurAna. However, Alqassem decided to give each segment its QurAna annotations (antecedent, aconcept and econcept) (see Figure 4.2); if the word is not a pronoun, a NONE value has been given.

In exploring her merged file, I found many parts that were inconsistent with my aims, such as the following:

1- Compound segments such Prefix and Stem have the same <seg id>, which leads to inconsistency of segments ids, Figure (4.2).

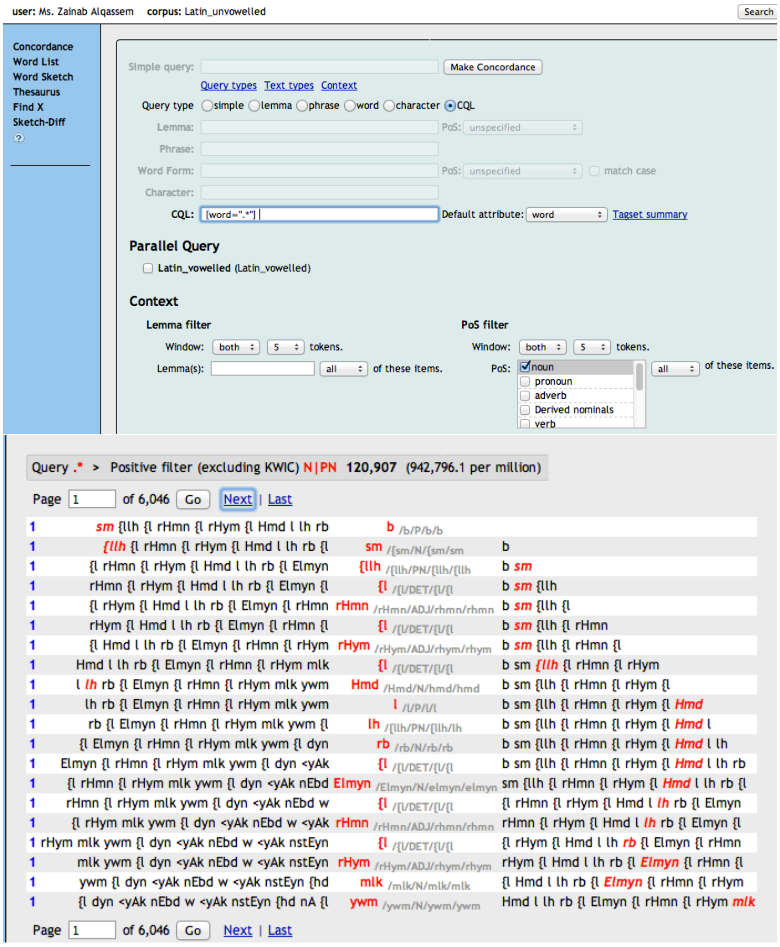


```
<verse id="3">
<seg id="92469">
y    VOC y    PREFIX|ya+  NONE     NONE     NONE
<g/>
>yhA     N    >yhA     STEM|POS:N||NOM NONE     NONE     NONE
</seg>
```

Figure (4.2): Two segments share the same <seg id>.

2- The QAC grammar features were undivided. Therefore, retrieving words with a specific grammar function is not applicable in her approach. For instance, if someone is interested in finding feminine nouns in the Quran, unfortunately, this is not applicable. Sketch engine gives facilities to restrict the search by typing the features as CQL or ticketing from the check box, but to have this feature in Sketch engine, each grammar feature needs to be declared by an XML attribute or a vertical line attribute separately.

3- The POS filter in her corpus does not work perfectly. I examined it by retrieving all nouns in the corpus leading to incorrect results, and the result showed unwanted POS tags as shown in Figure (4.3). In fact, this is a common problem in Sketch engine. To solve this issue, POS tags must be declared as an XML segment's attribute as well.



user: Ms. Zainab Alqassem   corpus: Latin_unvowelled

Query type ○simple ○lemma ○phrase ○word ○character ●CQL

Lemma: PoS: unspecified

Word Form: PoS: unspecified □ match case

CQL: [word=".*"]   Default attribute: word   Tagset summary

Parallel Query

□ Latin_vowelled (Latin_vowelled)

Context

Lemma filter

Window: both 5 tokens.

Lemma(s): all of these items.

PoS filter

Window: both 5 tokens.

PoS: ☑ noun □ pronoun □ adverb □ Derived nominals □ verb   all of these items.

Query .* > Positive filter (excluding KWIC) N|PN 120,907 (942,796.1 per million)

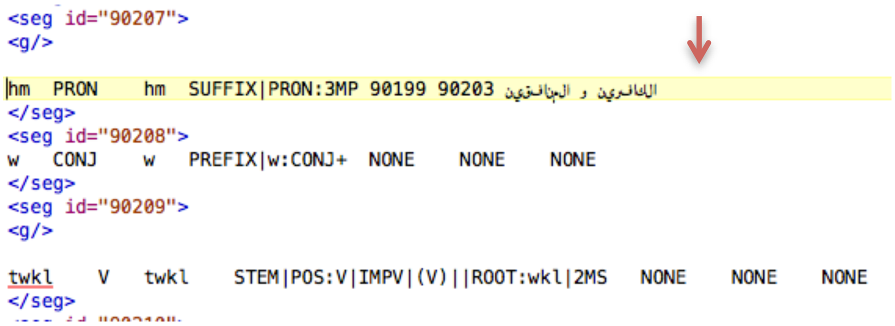
Page 1 of 6,046 Go Next | Last

```
1          sm {llh {l rHmn {l rHym {l Hmd l lh rb          b /b/P/b/b
1          {llh {l rHmn {l rHym {l Hmd l lh rb {l        sm /{sm/N/{sm/sm          b
1          {l rHmn {l rHym {l Hmd l lh rb {l Elmyn     {llh /{llh/PN/{llh/{llh    b sm
1          rHmn {l rHym {l Hmd l lh rb {l Elmyn {l      {l /{l/DET/{l/{l           b sm {llh
1          {l rHym {l Hmd l lh rb {l Elmyn {l rHmn   rHmn /rHmn/ADJ/rhmn/rhmn     b sm {llh {l
1          rHym {l Hmd l lh rb {l Elmyn {l rHmn {l      {l /{l/DET/{l/{l           b sm {llh {l rHmn
1          {l Hmd l lh rb {l Elmyn {l rHmn {l rHym   rHym /rHym/ADJ/rhym/rhym     b sm {llh {l rHmn {l
1        Hmd l lh rb {l Elmyn {l rHmn {l rHym mlk       {l /{l/DET/{l/{l           b sm {llh {l rHmn {l rHym
1        l lh rb {l Elmyn {l rHmn {l rHym mlk ywm     Hmd /Hmd/N/hmd/hmd           b sm {llh {l rHmn {l rHym {l
1          lh rb {l Elmyn {l rHmn {l rHym mlk ywm        l /l/P/l/l                b sm {llh {l rHmn {l rHym {l Hmd
1          rb {l Elmyn {l rHmn {l rHym mlk ywm {l       lh /{llh/PN/{llh/lh        b sm {llh {l rHmn {l rHym {l Hmd l
1        {l Elmyn {l rHmn {l rHym mlk ywm {l dyn        rb /rb/N/rb/rb             b sm {llh {l rHmn {l rHym {l Hmd l lh
1      Elmyn {l rHmn {l rHym mlk ywm {l dyn <yAk        {l /{l/DET/{l/{l           b sm {llh {l rHmn {l rHym {l Hmd l lh rb
1       {l rHmn {l rHym mlk ywm {l dyn <yAk nEbd Elmyn /Elmyn/N/elmyn/elmyn  sm {llh {l rHmn {l rHym {l Hmd l lh rb {l
1       rHmn {l rHym mlk ywm {l dyn <yAk nEbd w         {l /{l/DET/{l/{l           {l rHmn {l rHym {l Hmd l lh rb {l Elmyn
1       {l rHym mlk ywm {l dyn <yAk nEbd w <yAk     rHmn /rHmn/ADJ/rhmn/rhmn     rHmn {l rHym {l Hmd l lh rb {l Elmyn {l
1 rHym mlk ywm {l dyn <yAk nEbd w <yAk nstEyn          {l /{l/DET/{l/{l           {l rHym {l Hmd l lh rb {l Elmyn {l rHmn
1       mlk ywm {l dyn <yAk nEbd w <yAk nstEyn     rHym /rHym/ADJ/rhym/rhym     rHym {l Hmd l lh rb {l Elmyn {l rHmn {l
1       ywm {l dyn <yAk nEbd w <yAk nstEyn {hd       mlk /mlk/N/mlk/mlk           {l Hmd l lh rb {l Elmyn {l rHmn {l rHym
1       {l dyn <yAk nEbd w <yAk nstEyn {hd nA {l     ywm /ywm/N/ywm/ywm           Hmd l lh rb {l Elmyn {l rHmn {l rHym mlk
```

Page 1 of 6,046 Go Next | Last

Figure (4.3): Searching for all nouns in the Quran retrieved incorrect results.

4- Some QurAna annotations had a missed value in the English concept as shown in Figure (4.4).



```
<seg id="90207">
<g/>

hm   PRON     hm   SUFFIX|PRON:3MP 90199 90203 الكافرين و المنافقين
</seg>
<seg id="90208">
w    CONJ     w    PREFIX|w:CONJ+  NONE     NONE     NONE
</seg>
<seg id="90209">
<g/>

twkl     V    twkl     STEM|POS:V|IMPV|(V)||ROOT:wkl|2MS    NONE     NONE     NONE
</seg>
```

Figure (4.4): Missed value in QurAna English concept value.

However, the merged file by Alqassem had missed some annotations and needed to be revised. Therefore, re-preprocessing the two annotations would solve these issues. The processing strategy must be done in such a way that each segment must have a unique id. This is because we aim to convert the XML to an ontology format (OWL), and in ontology, each instance should have a unique id.

Processing the three datasets:

The final merged XML file aims to merge three datasets: Quranic Arabic Corpus (QAC), QurAna annotations and Qurany concepts. To merge the three datasets, I investigated two approaches:

First approach:

This involved merging the datasets by reading the three datasets in their actual format and fetching the identical chapter id and verse id and segment id together. Then the three outputs were merged into one line and written as an element in the XML tree. Using this approach, the processing took hours to cover a few chapters in the Quran. In addition, two of the datasets are inconsistent in the segment id. The Quran Corpus Annotation file has a segment id with a maximum of 128,119, while QurAna has a segment id with a maximum of 127,795. The result has 1,000 segments without QurAna annotations. So, this approach was not practical.

Second approach:

This involved converting each dataset into the XML version, and then selecting the wanted data using Xpath expressions. Xpath makes it possible to extract a specific type of data in the file without reading every line and to make comparisons each time. For example, Xpath could be used to extract the English Qurany Concept element with the "tree" attribute by using python Element Tree API:

```
if verse_qurany.getchildren():
        for QC in verse_qurany.findall('ENQuranyConcepts/[@tree]'):
                QuranyConceptElem = ET.SubElement(verse_MT, 'QuranyConcepts')
```

In implementing this approach, the merging processing went quickly; it took about minutes to merge the three datasets into one file. In addition, having an XML version for each dataset can be useful in further research as it can be uploaded in a corpus tool like Sketch engine to make it accessible to other researchers.

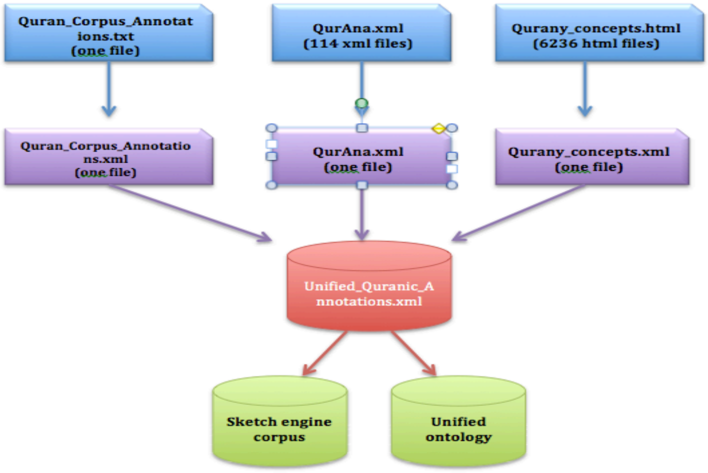

Quran_Corpus_Annotations.txt (one file)
QurAna.xml (114 xml files)
Qurany_concepts.html (6236 html files)
Quran_Corpus_Annotations.xml (one file)
QurAna.xml (one file)
Qurany_concepts.xml (one file)
Unified_Quranic_Annotations.xml
Sketch engine corpus
Unified ontology

Figure (4.5): Merging the three datasets in the second approach

First dataset: Qurany Concept:

As mentioned in Chapter One, Qurany Concept is a search tool for Quranic keywords. The tool retrieves verses that have the target keyword associated with the verse concepts. In addition, verses are clustered according to their concepts. In contrast, some verses have no concepts, while other verses might have more than one concept. However, we are concerned more about the verse clusters, which might be used as a genre for each verse in each chapter. The data are published as html files in the website http://www.comp.leeds.ac.uk/nora/html/. Each html file contains the information and concepts for one verse.

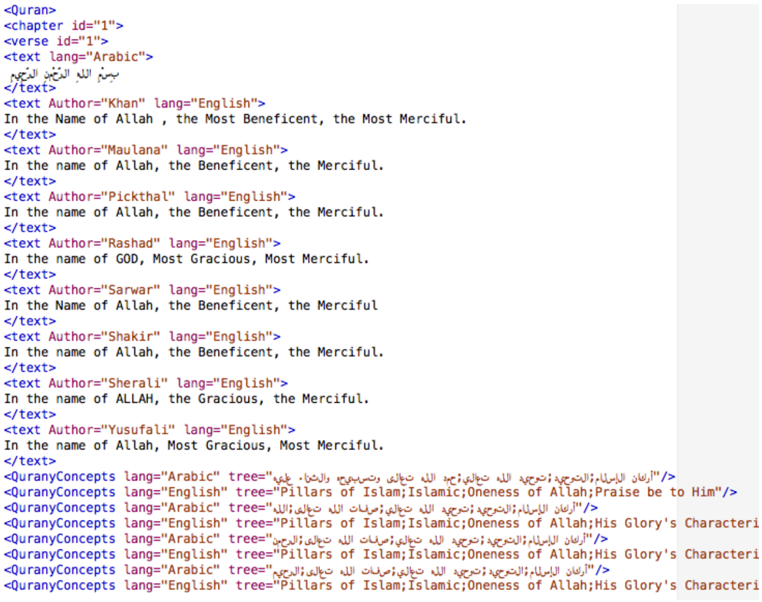
By following the second approach, the 6,323 html files will be converted completely to one XML structure. This conversion reduces the number of output files. So, instead of building 6,323 XML files, it is possible to build one XML file containing the information regarding 6,323 verses; this leads to a reduction in the time consumed and in the computer's memory usage in reading the Qurany files. The XML file contains all the information listed in the html file as the following elements and attributes:

	Element	Attribute	Level
1	Quran	Nothing	Root
2	chapter	Id	Quran's sub element
3	verse	Id	Chapter's sub element
4	text	Lang (Arabic/English), author	Verse's sub element
5	QuranyConcept	Lang(Arabic/Engish), tree	Verse's sub element

There are nine English translations for the verse. Each translation has been done by a different Islamic scholar. In addition, I kept the original verse concepts without splitting the verse. The only modification

I did was to replace the angle bracket (>) with a semi colon (;). This is to avoid problems in translating to XML entity references.

Figure (4.6) shows a sample of the desired XML design for the first chapter, first verse in the dataset.



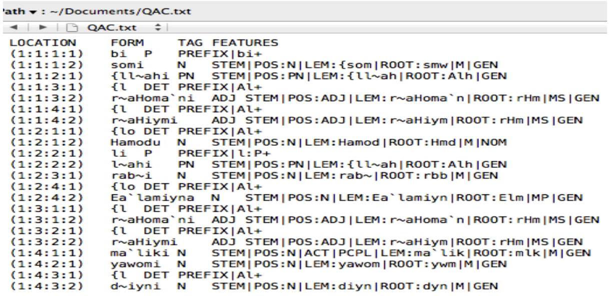
```
<Quran>
<chapter id="1">
<verse id="1">
<text lang="Arabic">
بِسمِ اللهِ الرَّحمٰنِ الرَّحِيمِ
</text>
<text Author="Khan" lang="English">
In the Name of Allah , the Most Beneficent, the Most Merciful.
</text>
<text Author="Maulana" lang="English">
In the name of Allah, the Beneficent, the Merciful.
</text>
<text Author="Pickthal" lang="English">
In the name of Allah, the Beneficent, the Merciful.
</text>
<text Author="Rashad" lang="English">
In the name of GOD, Most Gracious, Most Merciful.
</text>
<text Author="Sarwar" lang="English">
In the Name of Allah, the Beneficent, the Merciful
</text>
<text Author="Shakir" lang="English">
In the name of Allah, the Beneficent, the Merciful.
</text>
<text Author="Sherali" lang="English">
In the name of ALLAH, the Gracious, the Merciful.
</text>
<text Author="Yusufali" lang="English">
In the name of Allah, Most Gracious, Most Merciful.
</text>
<QuranyConcepts lang="Arabic" tree="أركان الإسلام;التوحيد;توحيد الله تعالي;حمد الله تعالى وتسبيحه والثناء عليه"/>
<QuranyConcepts lang="English" tree="Pillars of Islam;Islamic;Oneness of Allah;Praise be to Him"/>
<QuranyConcepts lang="Arabic" tree="أركان الإسلام;التوحيد;توحيد الله تعالي;صفات الله تعالى;الله"/>
<QuranyConcepts lang="English" tree="Pillars of Islam;Islamic;Oneness of Allah;His Glory's Characteri
<QuranyConcepts lang="Arabic" tree="أركان الإسلام;التوحيد;توحيد الله تعالي;صفات الله تعالى;الرحمن"/>
<QuranyConcepts lang="English" tree="Pillars of Islam;Islamic;Oneness of Allah;His Glory's Characteri
<QuranyConcepts lang="Arabic" tree="أركان الإسلام;التوحيد;توحيد الله تعالي;صفات الله تعالى;الرحيم"/>
<QuranyConcepts lang="English" tree="Pillars of Islam;Islamic;Oneness of Allah;His Glory's Characteri
```

Figure (4.6): The XML version of Qurany Concept dataset

Second dataset: Quranic Arabic Corpus (QAC):

The Quranic Arabic Corpus file is a tab separated text file. Each line has four main parts as shown in Figure (4.7). These parts are location, form, tag and features respectively.

Location describes the segment's location in four levels (chapter id: verse id: word id: token id). The form part is the token text in the Buckwalter translation. The tag part is the token's POS tag. The last part is the grammar features of the token including Stem/Prefix/Affix, lemma and root, gender, number and many other features.



```
LOCATION       FORM      TAG  FEATURES
(1:1:1:1)      bi   P    PREFIX|bi+
(1:1:1:2)      somi      N    STEM|POS:N|LEM:{som|ROOT:smw|M|GEN
(1:1:2:1)      {ll~ahi PN   STEM|POS:PN|LEM:{ll~ah|ROOT:Alh|GEN
(1:1:3:1)      {l   DET PREFIX|Al+
(1:1:3:2)      r~aHoma`ni   ADJ STEM|POS:ADJ|LEM:r~aHoma`n|ROOT:rHm|MS|GEN
(1:1:4:1)      {l   DET PREFIX|Al+
(1:1:4:2)      r~aHiymi     ADJ STEM|POS:ADJ|LEM:r~aHiym|ROOT:rHm|MS|GEN
(1:2:1:1)      {lo DET PREFIX|Al+
(1:2:1:2)      Hamodu   N    STEM|POS:N|LEM:Hamod|ROOT:Hmd|M|NOM
(1:2:2:1)      li   P    PREFIX|l:P+
(1:2:2:2)      l~ahi    PN   STEM|POS:PN|LEM:{ll~ah|ROOT:Alh|GEN
(1:2:3:1)      rab~i    N    STEM|POS:N|LEM:rab~|ROOT:rbb|M|GEN
(1:2:4:1)      {lo DET PREFIX|Al+
(1:2:4:2)      Ea`lamiyna   N    STEM|POS:N|LEM:Ea`lamiyn|ROOT:Elm|MP|GEN
(1:3:1:1)      {l   DET PREFIX|Al+
(1:3:1:2)      r~aHoma`ni   ADJ STEM|POS:ADJ|LEM:r~aHoma`n|ROOT:rHm|MS|GEN
(1:3:2:1)      {l   DET PREFIX|Al+
(1:3:2:2)      r~aHiymi     ADJ STEM|POS:ADJ|LEM:r~aHiym|ROOT:rHm|MS|GEN
(1:4:1:1)      ma`liki N    STEM|POS:N|ACT|PCPL|LEM:ma`lik|ROOT:mlk|M|GEN
(1:4:2:1)      yawomi   N    STEM|POS:N|LEM:yawom|ROOT:ywm|M|GEN
(1:4:3:1)      {l   DET PREFIX|Al+
(1:4:3:2)      d~iyni   N    STEM|POS:N|LEM:diyn|ROOT:dyn|M|GEN
```

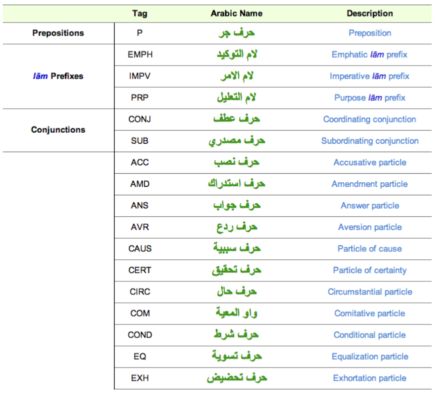
Figure (4.7): Quranic Arabic Corpus text file.

Converting the file to XML format by adding the following elements and attributes:

	Element	Attribute	Level
1	Quran	nothing	Root
2	Chapter	id	Quran’s sub element
3	Verse	id	Chapter’s sub element
4	Seg	Id, POS tag, lemma, root, gender, number, person, grammar, token_id	Verse’s sub element

The token_id indicates whether a segment has more than one part. For example, Arabic words may be linked with determiners (ال) as a prefix. There is no white space dividing the segment’s tokens, but each token has its grammar features. This attribute will be significant in adding the Glue tag in Sketch engine file format.

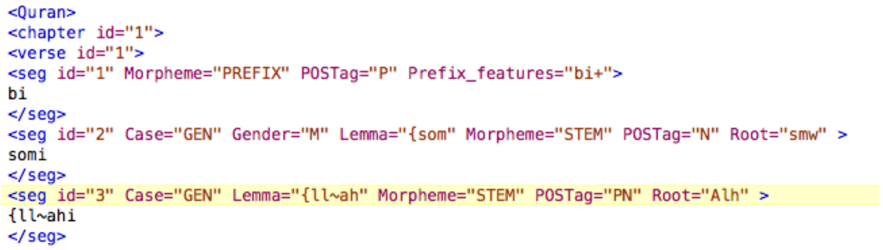
The grammar line has many features but no labels. For example, a grammar of D,M,3MP or EMPH has no obvious meaning to the user. Thus, labelling each feature leads to more clarity than the actual QAC list. To do this, each grammar line has been split and then labelled according to each Quranic Arabic Corpus tag set in Figure (4.8).



	Tag	Arabic Name	Description
Prepositions	P	حرف جر	Preposition
lām Prefixes	EMPH	لام التوكيد	Emphatic *lām* prefix
	IMPV	لام الامر	Imperative *lām* prefix
	PRP	لام التعليل	Purpose *lām* prefix
Conjunctions	CONJ	حرف عطف	Coordinating conjunction
	SUB	حرف مصدري	Subordinating conjunction
	ACC	حرف نصب	Accusative particle
	AMD	حرف استدراك	Amendment particle
	ANS	حرف جواب	Answer particle
	AVR	حرف ردع	Aversion particle
	CAUS	حرف سببية	Particle of cause
	CERT	حرف تحقيق	Particle of certainty
	CIRC	حرف حال	Circumstantial particle
	COM	واو المعية	Comitative particle
	COND	حرف شرط	Conditional particle
	EQ	حرف تسوية	Equalization particle
	EXH	حرف تحضيض	Exhortation particle

Figure (4.8): Sample of the tag sets in the Quran Corpus web site.

The implemented design is shown in Figure (4.9):



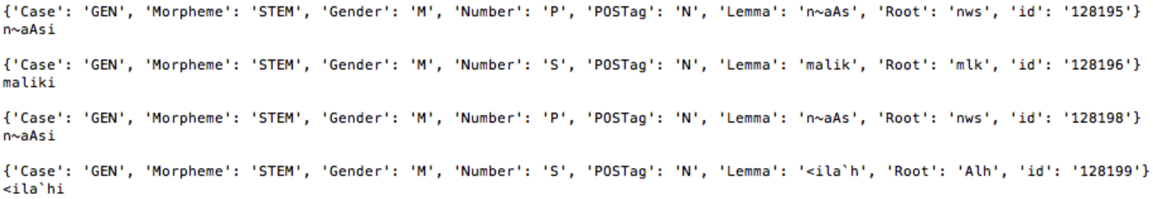
```
<Quran>
<chapter id="1">
<verse id="1">
<seg id="1" Morpheme="PREFIX" POSTag="P" Prefix_features="bi+">
bi
</seg>
<seg id="2" Case="GEN" Gender="M" Lemma="{som" Morpheme="STEM" POSTag="N" Root="smw" >
somi
</seg>
<seg id="3" Case="GEN" Lemma="{ll~ah" Morpheme="STEM" POSTag="PN" Root="Alh" >
{ll~ahi
</seg>
```

Figure (4.9): Quran Corpus Annotations in XML format.

With this XML structure, it is possible to retrieve a specific word with a specific grammar feature or POS tag using Xpath, such as the following python code:

```
import xml.etree.ElementTree as ET
root = ET.parse(‘QAC.xml’).getroot()
for segment in root.findall('.//seg/[@POSTag="N"]'):
        print segment.attrib, segment.text
```

The results of Xpath are shown in Figure (4.10).

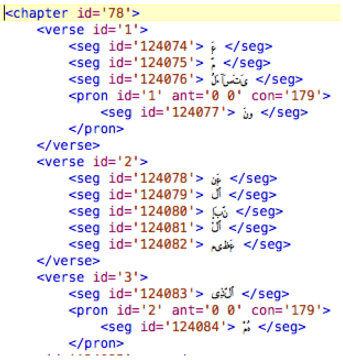


```
{'Case': 'GEN', 'Morpheme': 'STEM', 'Gender': 'M', 'Number': 'P', 'POSTag': 'N', 'Lemma': 'n~aAs', 'Root': 'nws', 'id': '128195'}
n~aAsi

{'Case': 'GEN', 'Morpheme': 'STEM', 'Gender': 'M', 'Number': 'S', 'POSTag': 'N', 'Lemma': 'malik', 'Root': 'mlk', 'id': '128196'}
maliki

{'Case': 'GEN', 'Morpheme': 'STEM', 'Gender': 'M', 'Number': 'P', 'POSTag': 'N', 'Lemma': 'n~aAs', 'Root': 'nws', 'id': '128198'}
n~aAsi

{'Case': 'GEN', 'Morpheme': 'STEM', 'Gender': 'M', 'Number': 'S', 'POSTag': 'N', 'Lemma': '<ila`h', 'Root': 'Alh', 'id': '128199'}
<ila`hi
```

Figure (4.10): Xpath to retrieve all nouns in QAC.xml dataset

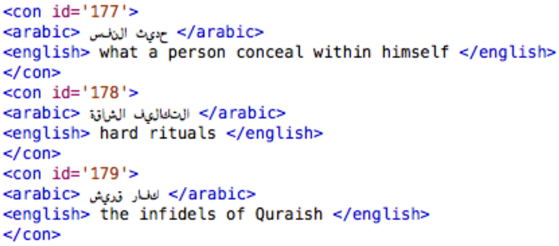
Third dataset: QurAna annotations:

In the QurAna dataset, there are 114 XML files and one concept file as well. Each XML file has a chapter as a root element, and verse, segment and pron as sub elements as shown in Figure (4.11). The aim of the preprocessing is to merge all XML files in one file and to eliminate the concept file. So, each con id is replaced with its actual concepts in the concept file.



```
<chapter id='78'>
    <verse id='1'>
        <seg id='124074'> عَ </seg>
        <seg id='124075'> مَّ </seg>
        <seg id='124076'> يَتَسَآءَلُ </seg>
        <pron id='1' ant='0 0' con='179'>
            <seg id='124077'> ونَ </seg>
        </pron>
    </verse>
    <verse id='2'>
        <seg id='124078'> عَنِ </seg>
        <seg id='124079'> ٱل </seg>
        <seg id='124080'> نَّبَإِ </seg>
        <seg id='124081'> ٱلْ </seg>
        <seg id='124082'> عَظِيمِ </seg>
    </verse>
    <verse id='3'>
        <seg id='124083'> ٱلَّذِى </seg>
        <pron id='2' ant='0 0' con='179'>
            <seg id='124084'> هُمْ </seg>
        </pron>
```

Figure (4.11): A part of pronxml-78.xml file



```
<con id='177'>
<arabic> حديث النفس </arabic>
<english> what a person conceal within himself </english>
</con>
<con id='178'>
<arabic> التكاليف الشاقة </arabic>
<english> hard rituals </english>
</con>
<con id='179'>
<arabic> كفار قريش </arabic>
<english> the infidels of Quraish </english>
</con>
```

Figure (4.12): part of concept.xml file

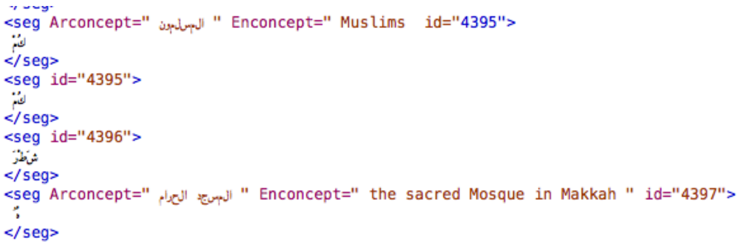
Besides the many XML files, the <pron> element has the concepts' information as (con) attribute. The attribute's value is an id used to import the actual concepts in Arabic and English from the concept file (see Figure 4.12). As we want to decrease the number of files and make processing XML files more practical and faster, I added the concept information (antecedent id, English and Arabic concept) to <pron> element as attributes.

However, merging QurAna XML files into one XML file means integrating the QurAna dataset into one file and so speeding up the final merging process. In addition, using this design, we can retrieve all pronouns referring to any concept by using the Xpath.

The desired XML structure is in the following table:

	Element	Attribute	Level
1	Quran	Nothing	Root
2	Chapter	Id	Quran’s sub element
3	Verse	Id	Chapter’s sub element
4	Seg	Id, ArConcept (Arabic concepts), EnConcept (English concept) and Antecedent id.	Verse’s sub element

A sample of QurAna.xml is seen in Figure (4.13).



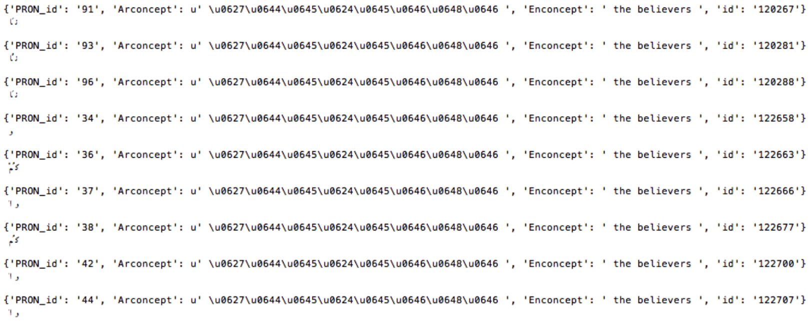
```
<seg Arconcept=" المسلمون " Enconcept=" Muslims  id="4395">
لَكُمْ
</seg>
<seg id="4395">
لَكُمْ
</seg>
<seg id="4396">
شَطْرَ
</seg>
<seg Arconcept=" المسجد الحرام " Enconcept=" the sacred Mosque in Makkah " id="4397">
هُ
</seg>
```

Figure (4.13): QurAna in a merged XML file

By using this XML structure, it is possible to retrieve all pronouns sharing the (the believers) concepts, for example. Xpath can be as follows:

words= root.findall('.//seg[@Enconcept=" the believers "]')

The results of that query are shown in Figure (4.14)



```
{'PRON_id': '91', 'Arconcept': u' \u0627\u0644\u0645\u0624\u0645\u0646\u0648\u0646 ', 'Enconcept': ' the believers ', 'id': '120267'}
 نَ
{'PRON_id': '93', 'Arconcept': u' \u0627\u0644\u0645\u0624\u0645\u0646\u0648\u0646 ', 'Enconcept': ' the believers ', 'id': '120281'}
 نَ
{'PRON_id': '96', 'Arconcept': u' \u0627\u0644\u0645\u0624\u0645\u0646\u0648\u0646 ', 'Enconcept': ' the believers ', 'id': '120288'}
 نَ
{'PRON_id': '34', 'Arconcept': u' \u0627\u0644\u0645\u0624\u0645\u0646\u0648\u0646 ', 'Enconcept': ' the believers ', 'id': '122658'}
 و
{'PRON_id': '36', 'Arconcept': u' \u0627\u0644\u0645\u0624\u0645\u0646\u0648\u0646 ', 'Enconcept': ' the believers ', 'id': '122663'}
 كُمْ
{'PRON_id': '37', 'Arconcept': u' \u0627\u0644\u0645\u0624\u0645\u0646\u0648\u0646 ', 'Enconcept': ' the believers ', 'id': '122666'}
 وا
{'PRON_id': '38', 'Arconcept': u' \u0627\u0644\u0645\u0624\u0645\u0646\u0648\u0646 ', 'Enconcept': ' the believers ', 'id': '122677'}
 كُمْ
{'PRON_id': '42', 'Arconcept': u' \u0627\u0644\u0645\u0624\u0645\u0646\u0648\u0646 ', 'Enconcept': ' the believers ', 'id': '122700'}
 وا
{'PRON_id': '44', 'Arconcept': u' \u0627\u0644\u0645\u0624\u0645\u0646\u0648\u0646 ', 'Enconcept': ' the believers ', 'id': '122707'}
 وا
```

Figure (4.14): shows the result of using Xpath to retrieve (the believers) concept

4.3 Design the unified Quranic Annotation dataset:

4.3.1 The Quran Structure:

The holy book Quran has 114 chapters. Each chapter has a different range of verses with a maximum of 286 verses and a minimum of 3 verses. Each verse has many words or segments.

In our XML file, the root node is <Quran> as it contains all the chapters. XML should have one root element, and 114 chapter sub elements.

4.3.2 XML rules:

To build an XML file, some W3C rules should be considered:

An XML file must have one root and the elements follow the root in a strict hierarchy. XML elements must follow the XML naming rules. Thus, a space and starting with numbers in an element’s name is not accepted. However, attributes' values can have any kind of characters inside quotations. Wrapping the XML elements is quite a sensitive process and should be done automatically using XML API. Any invalid element leads to an invalid XML file.

4.3.3 Distributing the annotations:

As we have three different annotations, I suggested the following annotation distribution:

- ***Quran level:***

Quran is the parent for all elements in the tree, and has no attribute.

- ***Chapter level:***

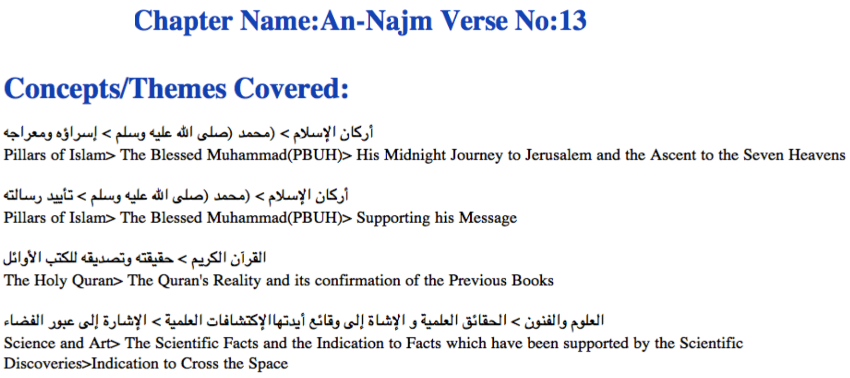
This level has no annotations to be associated except the chapter id.

- ***Verse level***:

By giving each verse a unique id, we avoid id duplications and make sure that each verse instance has its unique id. This solution will be more practical in converting XML files to OWL ontology as is shown in the ontology chapter.

In this level, Qurany concepts annotations are associated as a sub element to verse node.

The dilemma here in Qurany concepts is that each verse might have zero or more than one concept. Another issue is that each concept in Qurany has many sub concepts in a hierarchical structure. For example, Figure (4.15) shows four main concepts for one verse. A third issue is that each concept has been written in the English and Arabic languages without them being labelled with the language name, which makes splitting them more difficult.



Chapter Name:An-Najm Verse No:13

Concepts/Themes Covered:

أركان الإسلام > (محمد (صلى الله عليه وسلم > إسراؤه ومعراجه
Pillars of Islam> The Blessed Muhammad(PBUH)> His Midnight Journey to Jerusalem and the Ascent to the Seven Heavens

أركان الإسلام > (محمد (صلى الله عليه وسلم > تأييد رسالته
Pillars of Islam> The Blessed Muhammad(PBUH)> Supporting his Message

القرآن الكريم > حقيقته وتصديقه للكتب الأوائل
The Holy Quran> The Quran's Reality and its confirmation of the Previous Books

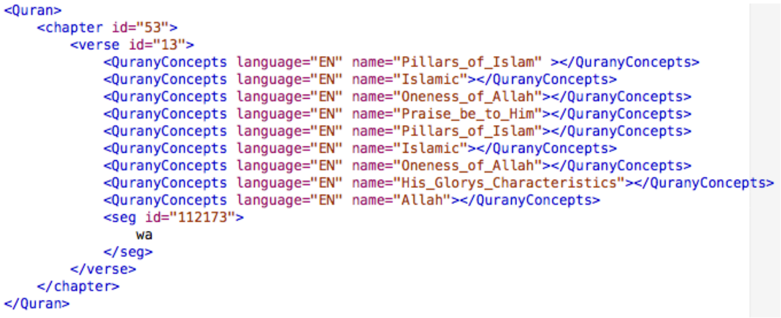
العلوم والفنون > الحقائق العلمية و الإشاة إلى وقائع أيدتها الإكتشافات العلمية > الإشارة إلى عبور الفضاء
Science and Art> The Scientific Facts and the Indication to Facts which have been supported by the Scientific Discoveries>Indication to Cross the Space

Figure (4,15): concepts covered in verse 13,chapter 53(An-Najm).

To solve these issues, I investigated many approaches and tried to find the most appropriate design to describe element hierarchy.

Approach#1:

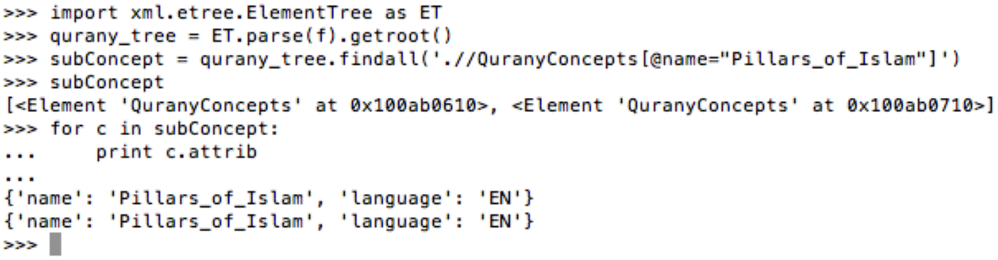
The first approach is to define an element <QuranyConcepts> with language and concept name attributes and store the concept as a name string value such as that shown in Figure (4.16).



```
<Quran>
    <chapter id="53">
        <verse id="13">
            <QuranyConcepts language="EN" name="Pillars_of_Islam" ></QuranyConcepts>
            <QuranyConcepts language="EN" name="Islamic"></QuranyConcepts>
            <QuranyConcepts language="EN" name="Oneness_of_Allah"></QuranyConcepts>
            <QuranyConcepts language="EN" name="Praise_be_to_Him"></QuranyConcepts>
            <QuranyConcepts language="EN" name="Pillars_of_Islam"></QuranyConcepts>
            <QuranyConcepts language="EN" name="Islamic"></QuranyConcepts>
            <QuranyConcepts language="EN" name="Oneness_of_Allah"></QuranyConcepts>
            <QuranyConcepts language="EN" name="His_Glorys_Characteristics"></QuranyConcepts>
            <QuranyConcepts language="EN" name="Allah"></QuranyConcepts>
            <seg id="112173">
                wa
            </seg>
        </verse>
    </chapter>
</Quran>
```

Figure (4.16): XML Qurany attributes design (1)

But after parsing the tree, getting the hierarchal concepts was inapplicable, as shown in Figure (4.17). The only thing we can retrieve is the node attribute itself because there is no hierarchy structure between the concepts.

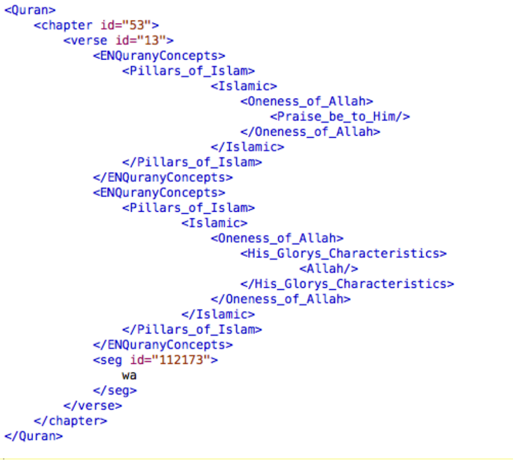


```
>>> import xml.etree.ElementTree as ET
>>> qurany_tree = ET.parse(f).getroot()
>>> subConcept = qurany_tree.findall('.//QuranyConcepts[@name="Pillars_of_Islam"]')
>>> subConcept
[<Element 'QuranyConcepts' at 0x100ab0610>, <Element 'QuranyConcepts' at 0x100ab0710>]
>>> for c in subConcept:
...     print c.attrib
...
{'name': 'Pillars_of_Islam', 'language': 'EN'}
{'name': 'Pillars_of_Islam', 'language': 'EN'}
>>> 
```

Figure (4.17): Python program to retrieve Qurany concepts in XML in approach (1).

Approach#2:

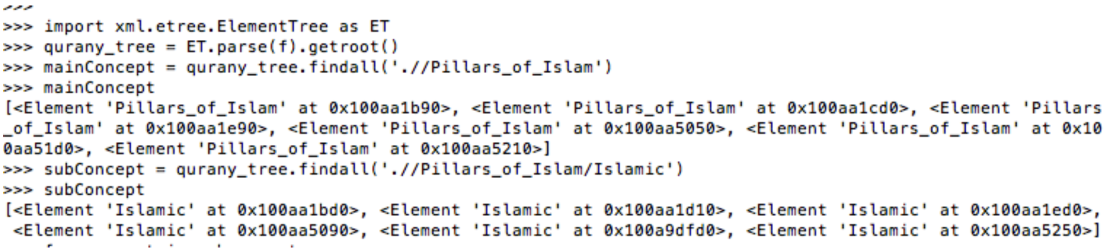
This approach creates a sub element of the verse node with <ENQuranyConcept> for English Qurany concepts, and <ARQuranyConcept> for Arabic Qurany concepts. These two sub elements are the parents for all the verse's concepts. The concepts are added to the previous elements as a sub element by splitting the concepts line into many concepts and creating the sub elements; see Figure (4.18).



```
<Quran>
    <chapter id="53">
        <verse id="13">
            <ENQuranyConcepts>
                <Pillars_of_Islam>
                            <Islamic>
                                <Oneness_of_Allah>
                                    <Praise_be_to_Him/>
                                </Oneness_of_Allah>
                            </Islamic>
                </Pillars_of_Islam>
            </ENQuranyConcepts>
            <ENQuranyConcepts>
                <Pillars_of_Islam>
                        <Islamic>
                            <Oneness_of_Allah>
                                <His_Glorys_Characteristics>
                                        <Allah/>
                                </His_Glorys_Characteristics>
                            </Oneness_of_Allah>
                        </Islamic>
                </Pillars_of_Islam>
            </ENQuranyConcepts>
            <seg id="112173">
                wa
            </seg>
        </verse>
    </chapter>
</Quran>
```

Figure (4.18): The hierarchy structure of Qurany Concepts dataset.

With this approach, the concepts hierarchy is obvious and readable. Moreover, it is possible to retrieve any concept or sub concept using Xpath, such as the following:



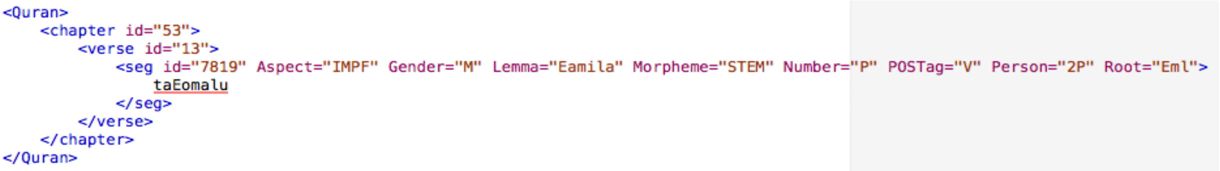
```
>>> import xml.etree.ElementTree as ET
>>> qurany_tree = ET.parse(f).getroot()
>>> mainConcept = qurany_tree.findall('.//Pillars_of_Islam')
>>> mainConcept
[<Element 'Pillars_of_Islam' at 0x100aa1b90>, <Element 'Pillars_of_Islam' at 0x100aa1cd0>, <Element 'Pillars
_of_Islam' at 0x100aa1e90>, <Element 'Pillars_of_Islam' at 0x100aa5050>, <Element 'Pillars_of_Islam' at 0x10
0aa51d0>, <Element 'Pillars_of_Islam' at 0x100aa5210>]
>>> subConcept = qurany_tree.findall('.//Pillars_of_Islam/Islamic')
>>> subConcept
[<Element 'Islamic' at 0x100aa1bd0>, <Element 'Islamic' at 0x100aa1d10>, <Element 'Islamic' at 0x100aa1ed0>,
 <Element 'Islamic' at 0x100aa5090>, <Element 'Islamic' at 0x100a9dfd0>, <Element 'Islamic' at 0x100aa5250>]
```

Figure (4.19): Python program to parse and query xml file.

I chose the second approach, as it is more practical in information retrieval and ontology mapping.

- ***Segment level:***

Any verse consists of many words or segments. A segment is a sub element of the verse node. It is the word morpheme (stem, prefix or suffix). Each segment has a unique id to avoid instance duplications. Beside the segment id, QAC and QurAna annotations are associated in the segment level. To unify QurAna and QAC annotations in one file, each annotation is associated as a segment attribute; see Figure (4.20).

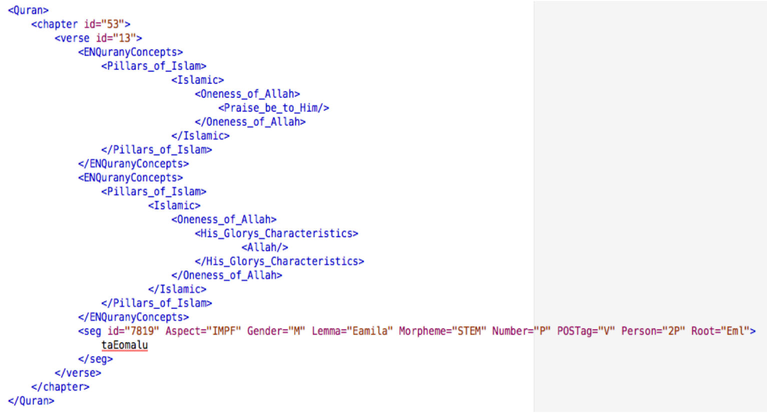


```
<Quran>
    <chapter id="53">
        <verse id="13">
            <seg id="7819" Aspect="IMPF" Gender="M" Lemma="Eamila" Morpheme="STEM" Number="P" POSTag="V" Person="2P" Root="Eml">
                taEomalu
            </seg>
        </verse>
    </chapter>
</Quran>
```

Figure (4.20): Unifying segment's annotations

Each segment has been associated with the Quran Annotations Corpus. In addition, pronoun segments have been associated with QurAna annotations as well.

An example of the overall design of the merged XML file is shown in Figure (4.21).



```
<Quran>
    <chapter id="53">
        <verse id="13">
            <ENQuranyConcepts>
                <Pillars_of_Islam>
                            <Islamic>
                                <Oneness_of_Allah>
                                    <Praise_be_to_Him/>
                                </Oneness_of_Allah>
                            </Islamic>
                </Pillars_of_Islam>
            </ENQuranyConcepts>
            <ENQuranyConcepts>
                <Pillars_of_Islam>
                        <Islamic>
                            <Oneness_of_Allah>
                                <His_Glorys_Characteristics>
                                        <Allah/>
                                </His_Glorys_Characteristics>
                            </Oneness_of_Allah>
                        </Islamic>
                </Pillars_of_Islam>
            </ENQuranyConcepts>
            <seg id="7819" Aspect="IMPF" Gender="M" Lemma="Eamila" Morpheme="STEM" Number="P" POSTag="V" Person="2P" Root="Eml">
                taEomalu
            </seg>
        </verse>
    </chapter>
</Quran>
```

Figure (4.21): Part of the Merged Quranic XML file:

Chapter 5: Implementation:

5.1 Choosing the programming language:

There are a plenty of programming languages that would be sufficiently mature and effective in text processing. However, we are seeking output efficiency. Obtaining an accurate and well-structured output is the main concern when choosing the programming language. In fact, the execution time factor was not considered in choosing the language; a target user will not use the software appropriate to a unified Quranic dataset, but one that addresses the contents of the output file.

Dealing with the text of the Quran requires sensitive consideration, as it is a holy book. Missing some Quranic text is unacceptable to the Muslim community. Thus, merging datasets dealing with thousands of files can result in some of the Quranic text being missed during processing. Of course, there are programming languages dedicated to processing holy books. However, a tracking process must be considered when developing user techniques (e.g. Xpath) to validate output as accurate by testing different segments and verses and comparing them with the actual text in the Holy Quran.

There are many choices of programming language that meet the requirements identified here, such as Python, Java and c++. I have had experience of using Python through a language module and also experience in processing corpora such as building n-grams, language dedication software and POS tags n-grams. This was my first experience of using Python but it was found to be easy to learn and expand knowledge in this area of programming.

Python has significant features which make it a friendly programming language. Python has a dynamic coding feature, which reduces the amount of code lines (Rossum, 1997). For example, there is no need to have type declarations for variables in Python whereas Java requires it.

On the other hand, Python has some disadvantages such as the limited speed of execution time and memory usage. Python's execution time is not the most perfect among programming languages. There are many programming languages which outperform Python in execution time and memory usage, such as C++ (Prechelt et al., 2000).

However, since the disadvantages of Python do not affect the output efficiency, it is preferred in writing this project's prototypes.

5.2 Using XML API in implementation

To create an XML tree, I used the python Application Programming Interface (API) Element Tree in creating the XML file for each of the three datasets. In addition, in the merging phase, I used the same API in order to parse the tree and merge specific information.

The Element Tree is a fixable and fast container and uses the Xpath technique. The XML Element Tree API confirmed that the XML structure is valid. Each opened element is closed automatically when creating a new instance of the same element. Another reason for choosing Element Tree is that it is built in all Python versions. There are some faster APIs, such as lxml, but since it needs to be installed, and its installation depends on the operating system, I restricted my choices to Element Tree API. I examined the Element Tree API and found it has a reasonable processing speed (Garabík, 2006).

Another advantage of using XML is in parsing the html file. It has been shown that Element Tree API can parse simple html files, but complicated html files have inconsistent open and closed XML tags.

One disadvantage of using Element Tree API is the limitation of Xpath expressions. Only simple expressions can be implemented in Element tree. While lxml implements much more complicated expressions (Element Tree XML API documentations).

5.3 Converting Qurany to xml (QuranytoXML.py)

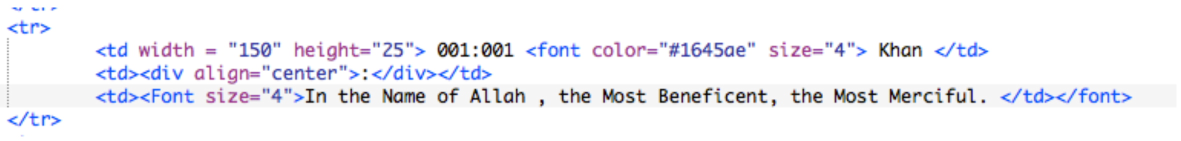
As mentioned earlier, the Qurany concept search tool is published through the http://*www*.quranytopics.appspot.com website as a list of html files. The files have a unified naming theme starting with chapter number, then the dash '-'symbol followed by verse number. This naming theme makes extracting the chapter and verse id straightforward as stated below. To upload the 6,323 html files, I used the "DownThemAll" utility in Firefox browser, which uploads all html files with one click.

Each HTML file consists of three main parts. The first part shows the chapter and verse information (chapter number, chapter name, verse number). The second part shows the eight English translations of the verse. The third section shows the concepts/themes in Arabic and English languages respectively.

The html markup has the same structure in all Qurany files. However, parsing the html file using the XML parser in Python seemed impossible as I received the following error message:

```
xml.etree.ElementTree.ParseError: not well-formed (invalid token): line 6, column 20
```

By revising the html tags, I found many invalid tokens such as changing the letter case in the opening and closing <font> tags, as shown in Figure (5.1). Since the XML parser is case sensitive, this is seen as two different tags.



```
<tr>
        <td width = "150" height="25"> 001:001 <font color="#1645ae" size="4"> Khan </td>
        <td><div align="center">:</div></td>
        <td><Font size="4">In the Name of Allah , the Most Beneficent, the Most Merciful. </td></font>
</tr>
```

Figure (5.1): Font tag with different letter case.

- Determine the root:

However, the first step in building the Qurany Concept XML tree is to determine the root of the tree. As mentioned in the Design chapter, the root will be <Quran>; the container of all Quranic chapters.

```
root = ET.Element('Quran')
```

- Reading Qurany files and creating chapter and verse elements:

The collection of html files is contained in the glob container. The glob library contains a directory's files by providing the directory's path as follows:

```
Qurany_files = glob.glob("qurany/*")
```

Thus, we can read any file inside the glob container as follows:

```
For htmlfile in sorted(Qurany_files):
```

At this stage, the implantation was working but the sorted function was not working properly. The files were reading in ascending order. However, the file name has combinations of numbers and the character '-'. Thus the results were ordered with chapter id first ignoring the verse id.

To solve the reading issue, two for loops were constructed to obtain the file name. The aim of the first loop was to get the chapter id with a maximum range of 115, and the second for loop was to get the verse id with a maximum range of 290. As mentioned earlier, each file had two parts with a dash in the middle. The first part is the chapter number and second part is the verse number.

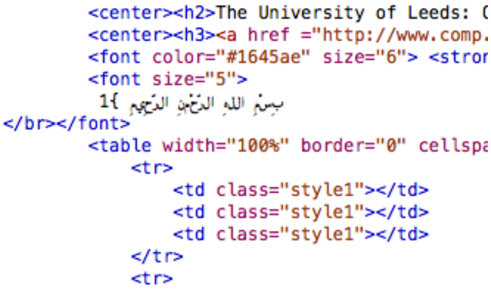
Another issue in the Element Tree API was when the chapter XML element definition was triggered, a new chapter element was created and in the final design, each verse had its own chapter element. This would add more XML tags since the chapter element is repeated for every verse. Since the aim is to have one chapter element for all related verses, this issue was solved by comparing each new file to see if its chapter id is the same as the previous file's chapter id. The initial value for the previous chapter was set to zero to get the condition for the first file in the directory.

Once the filename parameters were obtained (chapter id and verse id), the file name was constructed by concatenating this information with the directory's path then the file name was validated by testing if the file is found in the directory. If yes, then a verse sub element was created for the chapter element, then each line in the file name was read using the file function readlines(). After this, the number of lines in the file to be used were counted in a further loop.

```
prev_chapter=0
for chapter_id in range(1,115):
        if      pchapter!=chapter_id:
                chapter = ET.SubElement(root,'chapter')
                chapter.set('id',str(chapter_id))
        for verse_id in range(1,290):
                filename = path+str(chapter_id)+'-'+str(verse_id)+'.html'
```

```
if filename in Qurany_files:
        verse = ET.SubElement(chapter,'verse')
        verse.set('id', str(verseid))
        htmltxt = open(filename).readlines()
        lenhtmltxt = len(htmltxt)
```

- Extracting the verse text:



```
        <center><h2>The University of Leeds: (
        <center><h3><a href ="http://www.comp.
        <font color="#1645ae" size="6"> <stron
        <font size="5">
         1{ بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ
</br></font>
        <table width="100%" border="0" cellspa
            <tr>
                <td class="style1"></td>
                <td class="style1"></td>
                <td class="style1"></td>
            </tr>
            <tr>
```

Figure (5.2): the verse Arabic text

To extract the verse Arabic text information, the file's lines were read and specific html tags compared. As shown in Figure 5.2 , the Arabic verse text starts after the <font size="5"> tag. Thus, if the tag <font size="5"> is in the line, then the following line was read. This comparison is made by using the 'search regular expression' function, which mainly returns true if the searched text is in the line without considering its position.

The second step is to write this information in the verse text sub element to verse element and set the element's language attribute to Arabic.

Another issue in dealing with Arabic text in Python is the unicoding problems. Python reads the text in binary by default, and while processing the Arabic text, Python encodes it using the default encoding mode 'UTF-8'. However, writing the encoded text into a file must be done in binary; thus, a decoding function has to be set before writing the XML tree in the output file.

```
for line in range(lenhtmltxt):
        if re.search('<font size="5">',htmltxt[line]):
                Arabic_verse = htmltxt[line+1]
                Arabic_verse = re.sub(removechars,'',Arabic_verse)
                verseText = ET.SubElement(verse,'text')
                verseText.set('lang', 'Arabic' )
                #convert Byte to Unicode using Decode function
                verseText.text = Arabic_verse.decode('utf-8')
```

The Arabic verse has the verse id + the '}'character since the verse id is repeated and the character } is to separate the verse id and text. A regular expression pattern '[\d+/}]' is constructed to remove this unwanted digit and character.

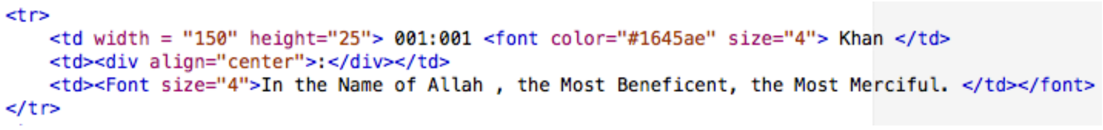
Another problem was in making a comparison where the compared line consisted of a tab and new line tag. These tags needed to be removed from each line to avoid incorrect matching. There were two solutions to remove the tags. One was by using the regular expression, and the other was by using the string function 'replace'. By applying both each time, I found that the string function is much faster in processing the file than the regular expression. It has been shown by many programmers that regular expression slows down the processing. Thus, I used the string function replace as the solution.

```
htmltxt[line] = htmltxt[line].replace('\t’,'')
htmltxt[line] = htmltxt[line].replace('\n','')
```

- Extracting the eight translations and author name:

There are eight verse English translations after the verse Arabic text, and each has its own author. To get these translations with author name, each one was assigned to the verse Text element but setting the language attribute to English and adding the author attribute.

To get the author name, it can be seen in Figure 5.3 that the author name is located after the tag <font color="#1645ae" size="4"> then by splitting the line with the mentioned tag. The result of splitting the line shows two elements: the first is repeated verse and chapter ids, and the second is the author name. Then, the second element is taken and the html tags removed using the regular expression with the pattern removeHTLtags = '<.*?>'.



```
<tr>
    <td width = "150" height="25"> 001:001 <font color="#1645ae" size="4"> Khan </td>
    <td><div align="center">:</div></td>
    <td><Font size="4">In the Name of Allah , the Most Beneficent, the Most Merciful. </td></font>
</tr>
```

Figure 5.3: the author and English translation html tags.

Moreover, the author name has spaces at the sides. These spaces were removed using the strip string function, which is dedicated to this purpose.

The extracted author name was assigned to a text verse element with the attribute Author.

```
if re.search('<font color="#1645ae" size="4">',htmltxt[line]):
        htmltxt[line] = htmltxt[line].split('<font color="#1645ae" size="4">')[1]
        htmltxt[line] = re.sub(removeHTLtags,'',htmltxt[line])
        htmltxt[line]=htmltxt[line].strip()
        Author = htmltxt[line]
        verseText = ET.SubElement(verse,'text')
        verseText.set('Author',Author)
```

To get the translations of the verse, as shown in Figure 5.3, the English translated verse started after the tag <td><Font size="4">. Then, the html tags were removed and English was assigned in the language attribute.

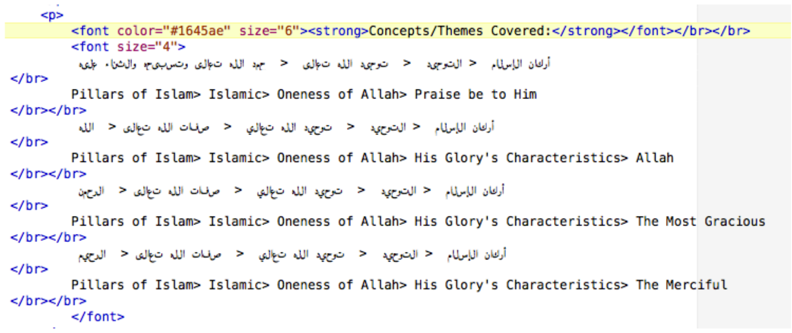
```
if re.search(r'<td><Font size="4">',htmltxt[line]):
                htmltxt[line] = re.sub(removeHTLtags,'',htmltxt[line])
                English_verse = htmltxt[line]                                        #write the
verse text in English in Verse Node
                verseText.set('lang','English')
                verseText.text = English_verse
```

- Extracting verse Concept/themes:

In this part, to find a general theme of the html tags was complicated. I set the for loop to read the concepts Arabic and English lines. However, I restricted the loop range to start from the index in the <font size=”4”> line;

```
if re.search('Concepts/Themes Covered:',htmltxt[line]):
                    index = line+1
```

It can be seen in Figure 5.4 that each Arabic concept line has a previous tag <front size=”4”> or </br></br>. Also, the English concept line has a previous tag </br> while the concepts list ends before the tag </font>. Thus, the for loop stops if the end of list is reached.



```
    <p>
        <font color="#1645ae" size="6"><strong>Concepts/Themes Covered:</strong></font></br></br>
        <font size="4">
         أركان الإسلام > التوحيد > توحيد الله تعالى > حمد الله تعالى وتسبيحه والثناء عليه
</br>
        Pillars of Islam> Islamic> Oneness of Allah> Praise be to Him
</br></br>
         أركان الإسلام > التوحيد > توحيد الله تعالي > صفات الله تعالى > الله
</br>
        Pillars of Islam> Islamic> Oneness of Allah> His Glory's Characteristics> Allah
</br></br>
         أركان الإسلام > التوحيد > توحيد الله تعالي > صفات الله تعالى > الرحمن
</br>
        Pillars of Islam> Islamic> Oneness of Allah> His Glory's Characteristics> The Most Gracious
</br></br>
         أركان الإسلام > التوحيد > توحيد الله تعالي > صفات الله تعالى > الرحيم
</br>
        Pillars of Islam> Islamic> Oneness of Allah> His Glory's Characteristics> The Merciful
</br></br>
        </font>
```

Figure (5.4): the concepts section and html tags

The software compares each line in the range to get the Arabic and English concepts. Once the target tags are found, then the following line is read as it must be the Arabic or English concepts.

As mentioned already, the tab and new line tags had been removed from the concepts. However, there was still unwanted space beside the sentence, which could be removed by keeping the actual sentence’s space in two steps. This could be achieved first by stripping the text using the string function ‘strip’, which meant removing spaces in the string’s first and last index and second, by splitting the concepts line in the angle bracket < and creating a list of the concepts. This step would remove the unwanted spaces and replace < with a semi colon. The replacement by semi colon was vital since XML API treats the < by defining the reference XML entity. Thus, this kept the concepts line clean of symbols, so it could be used by ‘merging.py’ software.

```
if htmltxt[i]=='</br></br>' or htmltxt[i]=='<font size="4">':
        ArabicConcepts = htmltxt[i+1]
        ArabicConcepts = ArabicConcepts.replace('\n','')
        ArabicConcepts = ArabicConcepts.replace('\t','')
        if ArabicConcepts !='</font>':
        # remove speces but keep the concept's inside spaces.
                ArabicConcepts = ArabicConcepts.strip()
                ArabicConcepts = ArabicConcepts.split('>')
                B=""
                for c in ArabicConcepts:
                        B +=c.strip() +';'
                #strip the last symbol
                B = B.strip(';')
```

After getting the concepts, it was possible to create the Qurany Concept sub element to verse by setting the language attribute and setting the concept list as a value of the tree attribute. Again, the Arabic text needed to be decoded to avoid Unicode problems.

```
                    QuranyConcepts = ET.SubElement(verse,'QuranyConcepts')
                    QuranyConcepts.set('lang', 'Arabic')
                    QuranyConcepts.set('tree',B.decode('utf-8'))
```

The same coding was used in extracting the English version except the compared tag was only </br>.

- Building the tree:

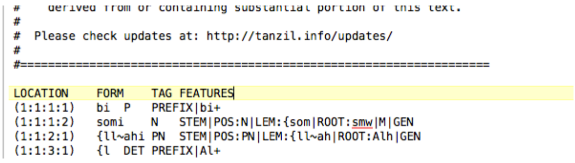
After building the XML tree, the last step was to build the tree using the functions Element Tree (root) and write. The write function output the tree in an XML file name 'Qurany.xml'. Also, the encoding mode in writing the contents was utf-8.

```
          tree = ET.ElementTree(root)
          tree.write(output,encoding='utf-8')
```

UTF-8 is the standard encoding in Python, which handles every character in the Unicode character set. It has been proved that utf-8 is ideal in dealing with the Arabic language.

5.4 Converting Quran Annotation Corpus (QAC) to XML (QACtoXML.py):

The Quran Annotation Corpus has a separated tab file; each line handles segment index, form, POS tag and grammar features.



```
#    derived from or containing substantial portion of this text.
#
#  Please check updates at: http://tanzil.info/updates/
#
#=================================================================

LOCATION	FORM	TAG	FEATURES
(1:1:1:1)	bi	P	PREFIX|bi+
(1:1:1:2)	somi	N	STEM|POS:N|LEM:{som|ROOT:smw|M|GEN
(1:1:2:1)	{ll~ahi	PN	STEM|POS:PN|LEM:{ll~ah|ROOT:Alh|GEN
(1:1:3:1)	{l	DET	PREFIX|Al+
```

Figure (5.5): the Corpus headline

The corpus has the copyright in the first lines. To start reading the segments, we need to jump to the first segment by using the search function in regular expression module to find the corpus headline as shown in Figure (5.5):

```
for i in range(QACLinesLength):
          if re.search("LOCATION\tFORM\tTAG\tFEATURES",QAC[i]):
                    index=i+1
```

Then, the line was split using the split function with tab '\t' parameter. The result was a list containing the four headline elements as follows:

```
          QACLine = QAC[line].split("\t")
          Location = QACLine[0].replace('\n','')
          form = QACLine[1]
          postag = QACLine[2]
          features = QACLine[3].replace('\n','')
```

```
        features = features.strip()
```

Since feature and location elements were in the sides of the line, then they had to have new line tags and spaces as well. To remove these spaces and tags, I used the replace function for the new line tag and the strip function to delete the white spaces at the sides.

- Extracting the chapter, verse, segment and token ids:

I defined a function to extract the four elements of segment location. Then, the Location variable was sent to the Location_extractor function. The function returned the four indices by splitting the location with a colon : . The splitting result was a list with four elements, but, there was a problem with segment id. A segment id is not a unique id in the same tree; this means it is possible for two different segments to have the same id under the same verse. Relying on the QAC segment id would lead to the loss of many segments. However, setting up an incremental segment index solved this issue.

According to Quranic Arabic Corpus documentation, the four elements were assigned to the following structure:

```
def Location_extractor(Location):
        Location = Location.split(":")
        chapterid = Location[0]
        verseid = Location[1]
        segmentid= Location[2]
        tokenid=Location[3]
        return chapterid,verseid,tokenid
```

The returned values were stored in the main part with the following line while the segment index incremented by one:

```
chapterid,verseid,tokenid= Location_extractor(Location)
segmentid+=1
```

In addition, a token refers to a compound segment. So, I translated the token as a segment feature where token id=2 means this segment has two components. seg.set('token_id', str(tokenid))

- Creating chapter, verse, segment, token elements:

To keep all the verses belonging to one chapter element, and to keep all segments belonging to one verse element, the chapter and verse id had to be compared with the previous one. If it was the same, then there was no need to define a new element. This was to avoid the redundancy of chapter id for each verse related to it; the same happened for segments. Below is an example of creating a chapter element.

```
        if      prev_chapter_id != chapterid:
                chapter = ET.SubElement(root,'chapter')
                chapter.set('id', str(chapterid))
```

```
            prev_chapter_id = chapterid
```

- Assigning form to segment's text:

The segment element was constructed as a sub element of verse. The aculae value of the segment is the form. Assigning the form to the segment is as follows: seg.text=form

- Assigning POS tag to segment's attribute:

As a POS tag describes the segment, it was ideal to add it as a segment attribute not as a segment sub element. This facilitated retrieving the segments with specific POS tags (e.g. noun) in Xpath query.

- Assigning features to a segment's attributes:

I defined a Features_extractor function, which takes the feature line as an input. The function splits the line with pipe '|' and creates a list of all features. Then, each feature in the list must pass through the style detector. This detector checks the feature and classifies its style.

By observing the QAC annotations style, I found that there are three main styles for the feature annotations as follows:

1- If the feature is Prefix, then the next feature is a prefix feature. Basically, the prefix annotations are one of these two styles: PREFIX|A:INTG+ or PREFIX|ka+. Then, this feature is passed to the predefined function PREFIX_Tager.

2- If a feature has a colon :, then this feature has two features as PRON:3MP, or a label with a feature such as ROOT:qwl. Then, this kind of feature is passed to the defined function Twinsfeatures.

3- However, if this is a single feature, then go to the labelling process. The actual QAC annotations lack labels. Thus, a feature like 3MP or PCPL is not clear for the user. The conversion software tries to solve this issue by adding a label for each feature in QAC. To do this, a feature must be compared for all probable features. As a single feature might be MS, which would mean this segment has a gender feature with M or Masculine and a number feature with S or Singular (Dukes, 2011). The following is a sample code of feature comparison from Singlefeature function:

```
if  f1=='M' or f1=='F':
        seg.set('Gender',f1)
```

The comparisons were made using different techniques depending on the feature complexity. For those features with two or three excepted values, a simple if comparison was constructed while if a feature had many values, such as the verb form, then a regular expression was used, such as the following:

```
elif  re.match(r'\(([IVX]+\)',f1):
            f1=f1.strip(")")
```

```
                    f1=f1.strip("(")
                    seg.set('Verb_Form',f1)
```

These values and labels were taken from the Quran Annotation Corpus tag set available online

The final step was to write all this information to the tree's root (Quran) element and print the tree in a QAX.xml file.

5.5 Converting QurAna to xml format (QurAnatoXML.py):

As mentioned in the chapter on design, QuraAna has 115 xml files and this software merges them in one XML file. So, the first step is to put all QurAna xml files in the glob [? global] container with the path QurAna_files = glob.glob('Quran-pron/*').

The reading process follows. Reading xml files means parsing their contents without actually reading the lines of the files (. Readlines()) as the previous software. Thus, this process does not need a close file function since no open file function was made.

As mentioned in relation to QuranytoXML.py software, getting an ordered list of files can be accomplished through constructing a "for loop". The "for loop" range is 1 to 115 and the file name is constructed by using the iterator " i" as in the following:

```
for i in range(1,115):
            file = 'Quran-pron/pronxml-'+str(i)+'.xml'
```

The above code cannot read the file name (concepts.xml) for the concept file. To do that, a separate code was developed to obtain a concepts file; the following lines parse the file and fetch the root.

```
Conceptfile='Quran-pron/concepts.xml'
_concepts = ET.parse(conceptfile).getroot()
```

Again, the root for the final XML file is Quran:

```
root = ET.Element('Quran')
```

Then, the "for loop" is iterated and gets each QurAna xml file and processes it as follows.

The root in QurAna xml is the chapter, while the final output is Quran. The change of root derives from the merging idea where 115 chapters are merged into one Quran file. Thus, the XML file must have one root element and the chapter element is repeated and cannot be used as the root. Thus, a root element must be changed to a Quran element.

```
file_root = ET.parse(file).getroot()
```

Then, obtain the chapter (root) id by using .get XML function. The output is assigned to a constructed chapter element.Then, iterating the root nodes using and examining the node tag by the following:

```
if node.tag=='verse':
```

If the node is verse then verse sub element is created and verse id is obtained and assigned to verse element.

```
elif node.tag=='seg':
```

If the node is segment then segment sub element is created and segment id and text are obtained and assigned to segment element.

If the node is the concept element, then getting the con attribute and iterating the concept tree is necessary in order to find the equivalent con id. Then, if the corresponding concept is found in the concepts file, the Arabic and English concepts are obtained and assigned to the segment attributed to Arconcept and Enconcept respectively.

```
for concept in _concepts.iter():
    if concept.get('id') == node.get('con'):
        for x in concept.getchildren():
            if x.tag=='arabic':
        seg.set('Arconcept',x.text)
            if x.tag=='english':
                seg.set('Enconcept',x.text)
```

The remaining attributes in the current node are pron id and Ant.

```
    if  node.get('id'):
            seg.set('PRON_id', node.get('id'))
```

The final step is to close the tree using tree= ET.ElementTree(root) and printing the tree using Element tree write function (QurAnaxmlfile).

5.6 Merging the three XML files in one XML file (merge.py):

The first step is to parse and obtain the root for the three datasets as the following:

```
qac_root= ET.parse(qac).getroot()
qurana_root = ET.parse(qurana).getroot()
qurany_root = ET.parse(qurany).getroot()
```

Then, for loops were constructed to get chapter id, verse id segment id.

Qurany verses having id and concepts were assigned to the merge tree MT_verse node.

QAC and QurAna have the same approach, but QurAna and QAC have different segment indexing. In QAC there is a segment with maximum id 128219 and in QurAna the maximum segment’s id is 127119. Thus, a new approach has to be constructed in order to merge two different indices.

According to Sharaf et al.(2012), QurAna uses QAC to obtain the pronoun segments only. The QurAna project aims to annotate every pronoun in the Quran with referenced information. Thus, only pronoun segments in QurAna are annotated, while other segments are not. Consequently, using the QurAna index is not important because the QAC index has annotated segments. The approach used here is declared in the following pseud code:

```
For QAC_segment in QAC:
    If QAC_segment is Pronoun then
        For QurAna_segment in QurAna:
            If QurAna_segment is Pronoun then
                Add QurAna_segment to QAC_segment
                Delete QurAna_segment
```

The programme reads the first pronoun in QurAna, then extends QAC Pronoun annotations by adding QurAna segment information, and finally deletes the QurAna_segment so it can be used again.

This approach does not compare indices because they are different. However, it assumes that the first pronoun to appear in QurAna is related to the QAC pronoun segment. To ensure this relationship, the pronoun segment in QurAna is deleted.

To validate the approach, last pronoun in the unified Quran dataset has been checked to find the associated QurAna annotation. Originally, the last pronoun in the Quran is located in the last verse in chapter 112. Back to QurAna dataset, the last pronoun in the Quran has the concept (Allah). By validating the concept in the unified Quranic XML dataset, the pronoun has the same concept. Thus, the approach is accurate and correct.

However, as the segments ids in the unified dataset and QurAna are different, the Antecedent id attribute was not added to segment annotations in unified dataset. The reason is that antecedent id uses Qurana segment id and this segment id is different in the unified dataset. Also, some Antecedent ids refer to the same value in QurAna concept. Thus, deleting the antecedent attribute will not influence the pronoun reference concepts.

To present English Qurany Concepts in hierarchal elements, I used the following code. Any concept contains whilte space or characters out of (Aa-Zz-0-9) will be replaced with underscore character(_):

```
if verse_qurany.findall('.//QuranyConcepts/[@lang="English"]'):
            Econcepts= verse_qurany.findall('QuranyConcepts/[@lang="English"]')
            for concept in c.get('tree').split(';'):
                concept=concept.strip()
                concept=re.sub('\W+','_',concept)
                ENcurrent = ET.SubElement(ENcurrent,concept)
```

During creating Qurany concept nodes, an error was arise which indecated invalid token. By investigating Qurany concepts, there was the concept (12 sons of Jacob) which starts with number and this is invalid XML node. Thus, I replace 12 with twelve to avoid XML naming problems.

The software has two main outputs. The first output is the Unified Quranic Annotation XML Dataset, which will be mapped to OWL ontology as a last step in the project. The second output is the unified Annotation Sketch engine Corpus. Getting two outputs is involved constructing two XML trees, one is a normal XML file and the other with Sketch engine vertical lines.

However, dealing with XML through XML API causes some constriction in writing the BuckWalter translation scripts. XML API translates some symbols to the predefined XML entity reference. Such symbols '>'and '&'are written in XML file as &gt and &amp respectively. This is considered as an obstacle to retrieving words containing such as these interpreted symbols. To solve this issue, a function was defined to replace the entity references with blanks.

```
for line in mergefile:
        line = line.replace('&quot;','')
        line = line.replace('&amp;','')
        line = line.replace('&apos;','')
        line = line.replace('&lt;','')
        line = line.replace('&gt;','')
        finalfile.write(line)
```

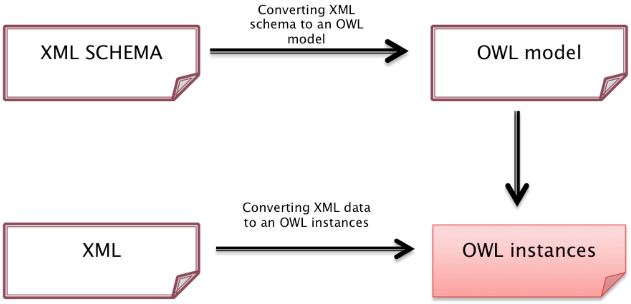
5.7 Mapping the Unified XML file to the Ontology structure:

One of the project's aims is to create the Unified Quranic Ontology using the XML file. Creating ontology from scratch takes a long time and a huge effort. However, there is an XML describing the Quranic annotations relationships and labelling each vocabulary with its meaning, such as labelling the Quranic annotations. Currently we do not have a complete ontology dedicated to describing every Quranic word, for which there are many reasons. Khan et al (2013) stated that defining Quranic taxonomy is causing a great debate among Islamic scholars who have different Islamic opinions. In fact, there are many Quranic words which have having different commentaries, such as Alflq, which has been described as another name for the word Moon or for the name of a hill. These are completely different meanings, therefore, to convert the XML to ontology, an ontology data model must be chosen.

.

As stated in chapter three, OWL model has significant features in describing data semantically, this type of modelling has been chosen as a target output in converting the unified Quranic dataset to an ontology model.

Mapping the XML to OWL Ontology:


XML SCHEMA
Converting XML schema to an OWL model
OWL model
XML
Converting XML data to an OWL instances
OWL instances

Figure(5.6): Mapping XML to OWL proposal by Bohring et al (2005).

There are many published approaches to mapping an XML file to OWL ontology. Such a proposal published by Bohring et al (2005) illustrated a way of converting XML to OWL. It has many steps as shown in figure (5.6). The approach assumes having an XML file, which holds the actual data in hierarchy, and an XML schema which describes XML file data types and hierarchies. The process of conversion is as follows:

1- The XML schema will be converted to OWL model or Domain model. The XML schema and the OWL model are equivalent as they describe a domain by defining data types and constraints.

2- Then, create the OWL instance by using the OWL model and XML data. The output is the actual Quranic data defined by the OWL model definitions.

The obstacle to using this proposal is that the XML dataset lacks an XML schema. There are many proposals for creating the XML schema by developing a prototype, creating the schema automatically by reading the XML data and inferring the structure. But, this could lead to the creation of an imperfect XML schema. However, the schema must be created manually. Another option for solving this problem is to rebuild the OWL model and create it manually. This solution sounds more accurate but is not practical; creating an ontology model from scratch needs a lot of effort and time.

Moreover, the XML structure could be used in modelling the OWL domain. From this understanding of the unified dataset the desired OWL model output design can be drawn, as in the following figure.

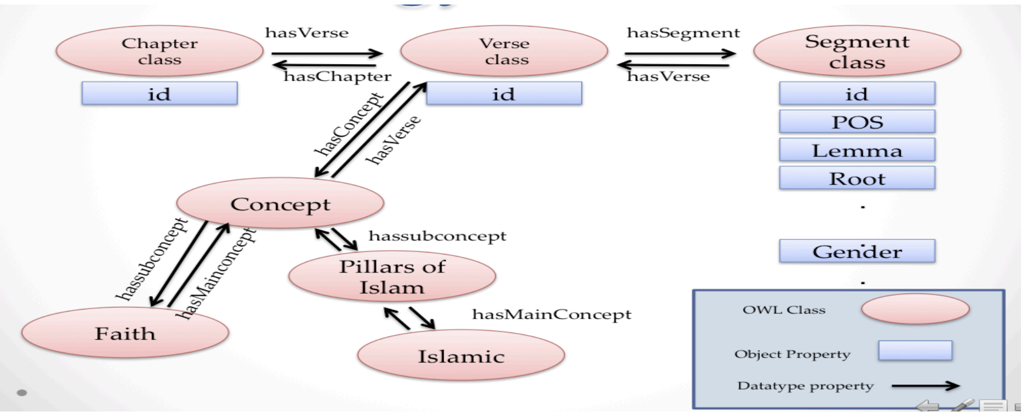


Figure (5.7): The OWL model design

There are three main components of the OWL model design. The first component is class, which can be chapter class, verse class, segment class and Qurany concept classes. Each class instance must have a unique identifier. Thus, chapter, verse, and segment must have a unique id.

The second component is the class's data type property, which is a relationship between a class and a literal value only. For instance, a segment class can have a data type property relationship with POS tag. However, values such POS tag, lemma, gender are considered as literal, not as a class. This is to avoid the functional property influence where each segment must have a relation with gender class where some segment does not have a gender.

The third component is the object property, which defines the relationship between two classes only, such as chapter class and verse class which has the hasVerse object property. The property also has the inverse feature just as verse class and chapter class have the relation hasChapter. These properties could be OWL special features, just as chapter and segment ids have a transitive relation property hasSeg.

To implement this design, a prototype has to be developed. In this study three prototypes were developed and took time to construct correctly. The restricted project plan cannot be extended to develop a fourth prototype. Also, incorporating details about ontologies and how to convert XML to OWL by reading many resources took a long time. Therefore, another approach was tried.

This alternative solution discovered that some ontology editors accept XML files. As discussed in chapter three, Protégé has the facility to convert an XML file to OWL ontology.

This project used the Protégé version 3.5, with full installation, in order to obtain all Protégé features and plug-ins. To start Protégé, create a new project and select OWL/RDF files. Then, to convert the XML file, the XML tab must be enabled in Protégé. To do this, click on the project tab and select configure. Then a list of Protégé plug-ins appears and shows the XML tab option. Clicking on that option will activate the XML tab option and display it alongside the Protégé main tabs.

The XML tab has two options for importing files. The first option is to import xml tree, and the second is to import XML instance. By examining the two options, both of them will be found to have the same function, which is to import an xml file. Then, by selecting the unified Quranic dataset XML file and then clicking on import, the dataset components will be shown in the class panel. Clicking on the OWLClass tab will bring up the converted ontology's components, listed in hierarchy.

Part of the converted OWL ontology is in Appendix K.

5.8 Porting the Unified XML file to the Sketch engine tool:

Preparing the unified Quranic dataset in Sketch Engine format has produced the following main points:

- It is proved that Sketch Engine accepts the XML corpus. The XML tags can describe the document structure (sentence, word, paragraph tags) and grammar features.

- In addition to using XML tags to describe the document structure, Sketch Engine uses vertical lines in listing the word attributes. This vertical line enables the application of language grammar to improve word behaviour such as in word sketches. Word sketches are involved in finding a lemma with the most frequent neighbour words and in listing the findings in tabular form.

  The attributes of vertical lines can be defined in many ways. There is no single way of defining them. Also, some predefined attributes are more strictly defined in Sketch Engine. The important thing in creating the corpus and vertical line is to separate the attributes with tabs. Such a vertical line is used in the project's dataset in the following way:

  Word Lemma TAG

- Features and annotations having a hierarchical structure and multi-valued annotations can be represented in Sketch Engine. This type of annotation has to be defined in a recursive header. Sketch Engine uses symbols to describe the hierarchy and the multi-valued features. According to the Corpora header definition in Sketch Engine, a header having multiple values can be represented using these symbols

  POSTag = Noun|Noun::PN

5.8.1 Implementing Sketch Engine formats:

To implement the Sketch Engine format functions have been defined in merge.py prototype. The process of creating an XML corpus is the same as creating the unified Quranic Annotation dataset, except for adding some changes in the segment text and the hierarchal xml elements.

1- Implementing the vertical line structure.

   To add the vertical line in the core corpus text (segment text), a function has been defined to add the three components of our defined attributes.

```
seg_MT = ET.SubElement(verse_MT,'seg')
seg_MT.text = str(text.decode('utf-8')) +'\t'+ str(Lemma.decode('utf-8')) + '\t' + str(pos)
```

2- Implementing the hierarchal and multi-valued XML elements.

   The hierarchal and multi-valued XML elements are represented in Qurany concept elements. The defined function SketchEngineFormat returns the XML value in Sketch Engine format. The symbol '|' represents the multi-valued element, while '::' represents the hierarchal feature in the element.

```
conceptline = conceptline.split(';')
return '|'.join('::'.join(conceptline[:INDX]) for INDX in range(1,len(conceptline)+1)).strip()
```

Sketch Engine hierarchal format is very sensitive in requiring spaces between the text and the symbols. Thus, a strip function has been used to guarantee that no spaces appear between lines of text.

5.8.2 Configuring the Corpus:

To upload the XML Quranic corpus, Sketch Engine must be configured to compile the corpus structure. There are hierarchal and multi-valued elements, therefore a corpus definition is required.

An example of defining the Qurany concepts has the following components:

an element that has attributes defined by structure; a basic XML element which has no attributes and can be defined simply by writing ATTRIBUTE"word", and a structure defined by a tree attribute, which lists the Qurany concepts.

Another optional feature can be defined as the Label. It can handle a title of the element to be displayed in text type queries in Sketch Engine. A MULTIVALUE feature must be specified to indicate that an element can have more than one value with the same element name. Specifying ‘1’ to MULTIVALUE enables the feature. MULTISEP must be defined with a symbol such as ‘|’. This symbol indicates that an XML element can have more than one value. HIERARCHICAL can also be defined by a symbol, and ‘::’ was used in this study to indicate that a previous element is a parent of the second element, such as Pillars of Islam::Islamic.

```
STRUCTURE "ENQuranyConcepts" {
ATTRIBUTE "tree"{
LABEL "English Qurany Concepts Tree"
MULTIVALUE "1"
MULTISEP "|"
HIERARCHICAL "::"}}
```

Another element, such as chapter element with id attribute, can be defined simply as below. The same definition is used for the verse and segment elements.

```
STRUCTURE "chapter" {
ATTRIBUTE "id"{
LABEL "Chapter Number"}}
```

The complete configuration file can be found in the Appendix J to this dissertation.

Using Text types, a user can specify their search by selecting any of the grammar and/or semantic feature.

Chapter 6: Evaluation

This chapter evaluates three main outputs; the unified annotations XML dataset; the Sketch Engine tool, and the unified Quranic OWL ontology using Protégé editor.

6.1 Evaluating the unified Quranic Annotations XML Dataset:

To evaluate the XML file, Xpath expressions were used to get the answers to the following questions. These questions measure the data integrity and the accuracy of the merged XML dataset (Unified_Quranic_Corpus_v1.xml).

1- **Find all verses which have the concept Hunting**

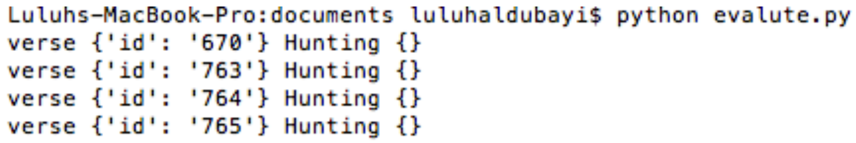
Xpath code:

XML_root= ET.parse("mergeLast_Version.xml").getroot()

verses = XML_root.findall('.//verse')

concepts = verse.findall('.//Hunting')

The answer:



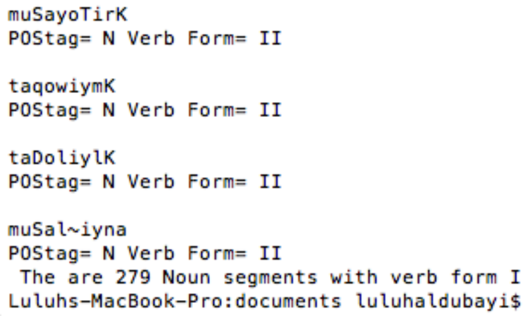
```
Luluhs-MacBook-Pro:documents luluhaldubayi$ python evalute.py
verse {'id': '670'} Hunting {}
verse {'id': '763'} Hunting {}
verse {'id': '764'} Hunting {}
verse {'id': '765'} Hunting {}
```

2- **Find segments having Noun POS tag and verb form II.**

Xpath code:

exp2 = XML_root.findall('.//seg/[@POSTag="N"][@Verb_Form="II"]')

The answer(part of the result with the total number):

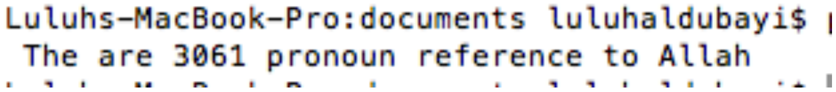


```
muSayoTirK
POStag= N Verb Form= II

taqowiymK
POStag= N Verb Form= II

taDoliylK
POStag= N Verb Form= II

muSal~iyna
POStag= N Verb Form= II
 The are 279 Noun segments with verb form I
Luluhs-MacBook-Pro:documents luluhaldubayi$
```

3- **Find all segments having Pronoun referring to Allah.**

exp2 = XML_root.findall('.//seg/[@Enconcept="Allah"]')

The answer:



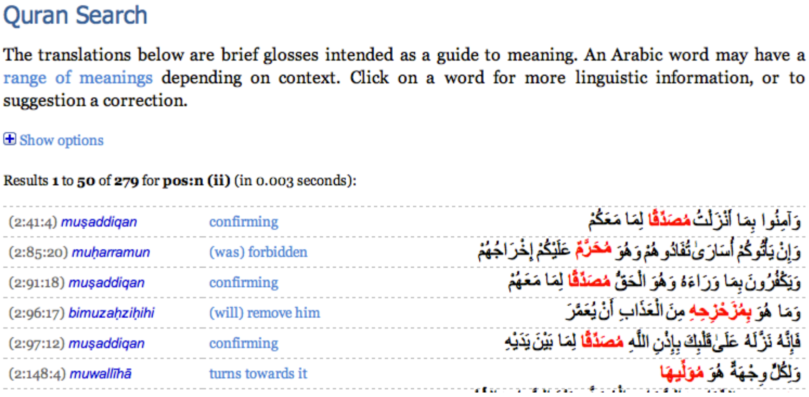
```
Luluhs-MacBook-Pro:documents luluhaldubayi$
 The are 3061 pronoun reference to Allah
```

Evaluation Discussion:

Three questions are posed in order to measure the merged XML file accuracy and evaluate the three datasets by finding the related annotations.

The first question measures the correctness and accuracy of Qurany concepts verse annotations. To know whether the results are accurate or not, Qurany concept/Topic website (http://quranytopics.appspot.com) was used to navigate the concept of 'tree' to Hunting concepts and to browse how many verses are found under that definition. It was found that there are four verses related to the hunting concept. Thus, the results found in the XML file are correct.

The second question measures the accuracy of QAC annotations in the merged dataset. The question posed to find nouns tagged with N and verb form II produced a result of 279 occurrences of segments. To validate the result, the QAC morphology search tool available in Quran corpus website (http://corpus.quran.com/morphologicalsearch.jsp) gave the same result shown in figure (6.1), which indicated that the unified XML file is correct.



Quran Search

The translations below are brief glosses intended as a guide to meaning. An Arabic word may have a range of meanings depending on context. Click on a word for more linguistic information, or to suggestion a correction.

⊞ Show options

Results **1** to **50** of **279** for **pos:n (ii)** (in 0.003 seconds):

(2:41:4) *muṣaddiqan*	confirming	وَآمِنُوا بِمَا أَنْزَلْتُ مُصَدِّقًا لِمَا مَعَكُمْ
(2:85:20) *muḥarramun*	(was) forbidden	وَإِنْ يَأْتُوكُمْ أُسَارَىٰ تُفَادُوهُمْ وَهُوَ مُحَرَّمٌ عَلَيْكُمْ إِخْرَاجُهُمْ
(2:91:18) *muṣaddiqan*	confirming	وَيَكْفُرُونَ بِمَا وَرَاءَهُ وَهُوَ الْحَقُّ مُصَدِّقًا لِمَا مَعَهُمْ
(2:96:17) *bimuzaḥziḥihi*	(will) remove him	وَمَا هُوَ بِمُزَحْزِحِهِ مِنَ الْعَذَابِ أَنْ يُعَمَّرَ
(2:97:12) *muṣaddiqan*	confirming	فَإِنَّهُ نَزَّلَهُ عَلَىٰ قَلْبِكَ بِإِذْنِ اللَّهِ مُصَدِّقًا لِمَا بَيْنَ يَدَيْهِ
(2:148:4) *muwallīhā*	turns towards it	وَلِكُلٍّ وِجْهَةٌ هُوَ مُوَلِّيهَا

Figure (6.1): The morphological search tool

As mentioned in chapter two, the tool has limited search options, for example, it cannot be used to validate noun with gender occurrences.

The third question measures QurAna pronoun references. There are 3,061 pronouns in Quran referring to Allah. In fact, there is no available tool that can be used to validate the result. However, according to QurAna published paper (Sharaf et al., 2012), there are 3,061 pronoun references to Allah. Thus, the unified XML file has proven accuracy.

6.2 Evaluating the unified Quranic Annotations XML Dataset in Sketch engine:

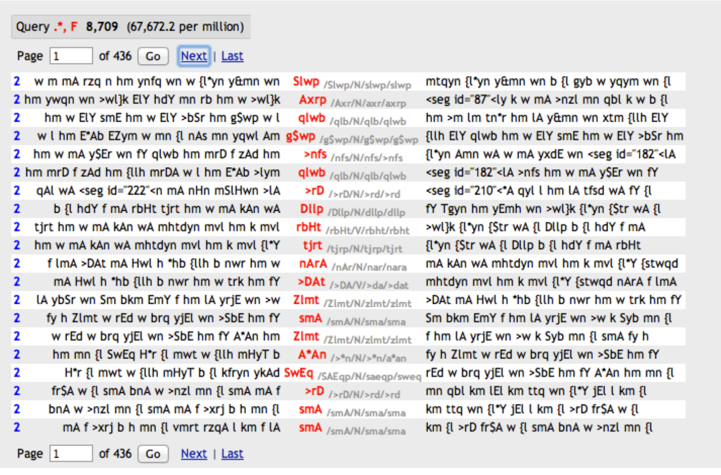
Translating the SKE corpus to Arabic:

Since the contributing users are Arabic native speakers, they cannot evaluate the Latin corpus, so two more datasets were created by replacing the segment text with Arabic text. In Alqassem's (2013) merging of databases project, she first merged and translated the Quran text and created four Quranic corpora; an unvowelled Arabic corpus; a vowelled Arabic corpus; an unvowelled Latin corpus, and a vowelled Latin corpus. Thus, her translations created an Arabic version of the unified dataset. To do this, a prototype was developed which mainly reads Alqassem's four corpora, obtains the text and re-assigns the segment's text.

First, this dataset file parses (SKE_Unfied_Quranic_Corpus_Latin_v1.xml) file and obtains all segments using the Findall function with the Xpath expressions.

Self-Evaluation:

Opening the unified dataset in Sketch Engine reveals many search options. However, there are five main ways to query the corpus. Two ways were tested, the first being the simple query in which a word is simply entered. The second is the advanced query using CQL, such as querying every word in the corpus with Gender F.



Query .*, F 8,709 (67,672.2 per million)

Page 1 of 436 Go Next | Last

2	w m mA rzq n hm ynfq wn w {l*yn y&mn wn	Slwp	/Slwp/N/slwp/slwp	mtqyn {l*yn y&mn wn b {l gyb w yqym wn {l
2	hm ywqn wn >wl}k ElY hdY mn rb hm w >wl}k	Axrp	/Axr/N/axr/axrp	<seg id="87"<ly k w mA >nzl mn qbl k w b {l
2	hm w ElY smE hm w ElY >bSr hm g$wp w l	qlwb	/qlb/N/qlb/qlwb	hm >m lm tn*r hm lA y&mn wn xtm {llh ElY
2	w l hm E*Ab EZym w mn {l nAs mn yqwl Am	g$wp	/g$wp/N/g$wp/g$wp	{llh ElY qlwb hm w ElY smE hm w ElY >bSr hm
2	hm w mA y$Er wn fY qlwb hm mrD f zAd hm	>nfs	/nfs/N/nfs/>nfs	{l*yn Amn wA w mA yxdE wn <seg id="182"<lA
2	hm mrD f zAd hm {llh mrDA w l hm E*Ab >lym	qlwb	/qlb/N/qlb/qlwb	<seg id="182"<lA >nfs hm w mA y$Er wn fY
2	qAl wA <seg id="222"<n mA nHn mSlHwn >lA	>rD	/>rD/N/>rd/>rd	<seg id="210"<*A qyl l hm lA tfsd wA fY {l
2	b {l hdY f mA rbHt tjrt hm w mA kAn wA	Dllp	/Dllp/N/dllp/dllp	fY Tgyn hm yEmh wn >wl}k {l*yn {$tr wA {l
2	tjrt hm w mA kAn wA mhtdyn mvl hm k mvl	rbHt	/rbHt/V/rbht/rbht	>wl}k {l*yn {$tr wA {l Dllp b {l hdY f mA
2	hm w mA kAn wA mhtdyn mvl hm k mvl {l*Y	tjrt	/tjrp/N/tjrp/tjrt	{l*yn {$tr wA {l Dllp b {l hdY f mA rbHt
2	f lmA >DAt mA Hwl h *hb {llh b nwr hm w	nArA	/nAr/N/nar/nara	mA kAn wA mhtdyn mvl hm k mvl {l*Y {stwqd
2	mA Hwl h *hb {llh b nwr hm w trk hm fY	>DAt	/>DA/V/>da/>dat	mhtdyn mvl hm k mvl {l*Y {stwqd nArA f lmA
2	lA ybSr wn Sm bkm EmY f hm lA yrjE wn >w	Zlmt	/Zlmt/N/zlmt/zlmt	>DAt mA Hwl h *hb {llh b nwr hm w trk hm fY
2	fy h Zlmt w rEd w brq yjEl wn >SbE hm fY	smA	/smA/N/sma/sma	Sm bkm EmY f hm lA yrjE wn >w k Syb mn {l
2	w rEd w brq yjEl wn >SbE hm fY A*An hm	Zlmt	/Zlmt/N/zlmt/zlmt	f hm lA yrjE wn >w k Syb mn {l smA fy h
2	hm mn {l SwEq H*r {l mwt w {llh mHyT b	A*An	/>*n/N/>*n/a*an	fy h Zlmt w rEd w brq yjEl wn >SbE hm fY
2	H*r {l mwt w {llh mHyT b {l kfryn ykAd	SwEq	/SAEqp/N/saeqp/sweq	rEd w brq yjEl wn >SbE hm fY A*An hm mn {l
2	fr$A w {l smA bnA w >nzl mn {l smA mA f	>rD	/>rD/N/>rd/>rd	mn qbl km lEl km ttq wn {l*Y jEl l km {l
2	bnA w >nzl mn {l smA mA f >xrj b h mn {l	smA	/smA/N/sma/sma	km ttq wn {l*Y jEl l km {l >rD fr$A w {l
2	mA f >xrj b h mn {l vmrt rzqA l km f lA	smA	/smA/N/sma/sma	km {l >rD fr$A w {l smA bnA w >nzl mn {l

Page 1 of 436 Go Next | Last

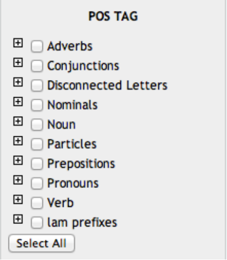
Figure(6.2): CQL query result

Testing the POS Filter:

Sketch engine has the option to create the POS filter using the POS attribute in a corpus vertical line. Thus, by examining the POS filter, the results were shown to be incorrect and the filter needed to be developed. However, the POS filter was adapted by labelling the POS tags using the defined function SKEPosFilter in merge.py prototype. Each POS value is passed to the function and returned with a POS label such as the following:

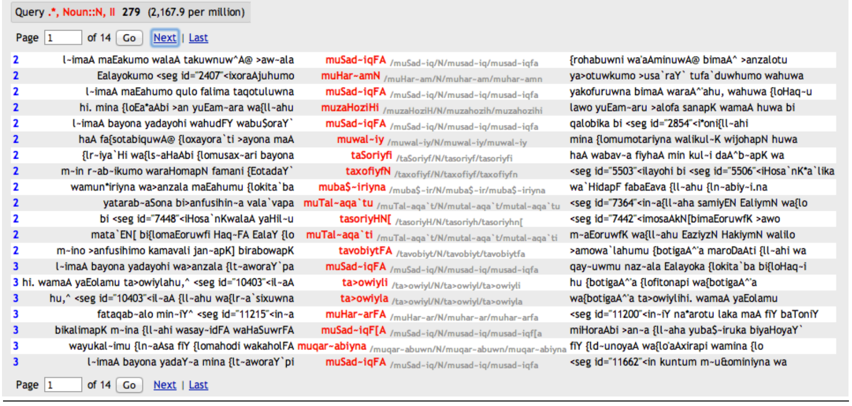
```
if pos == 'N' or pos=='PN' or pos=='IMPN':
                              return "Noun|Noun::"+pos
```

Then, configure the corpus to accept the POS labels and present them in a multi choice box as the following figure:



Figure(6.3): Sketch engine POS filter

To test the POS tag, a query for finding noun words with verb form II was constructed. The result is shown in figure (6.4) . Obviously, the same result of 279 nouns with II verb form was achieved. Thus, the POS filter and grammar filter were proved to be correct and accurate.



Query .*, Noun::N, II 279 (2,167.9 per million)

Page 1 of 14 Go Next | Last

2	l~imaA maEakumo walaA takuwnuw^A@ >aw~ala	muSad~iqFA	/muSad~iq/N/musad~iq/musad~iqfa	{rohabuwni wa'aAminuwA@ bimaA^ >anzalotu
2	Ealayokumo <seg id="2407"<ixoraAjuhumo	muHar~amN	/muHar~am/N/muhar~am/muhar~amn	ya>otuwkumo >usa`raY` tufa`duwhumo wahuwa
2	l~imaA maEahumo qulo falima taqotuluwna	muSad~iqFA	/muSad~iq/N/musad~iq/musad~iqfa	yakofuruwna bimaA waraA^'ahu, wahuwa {loHaq~u
2	hi. mina {loEa*aAbi >an yuEam~ara wa{ll~ahu	muzaHoziHi	/muzaHoziH/N/muzahozih/muzahozihi	lawo yuEam~aru >alofa sanapK wamaA huwa bi
2	l~imaA bayona yadayohi wahudFY wabu$oraY`	muSad~iqFA	/muSad~iq/N/musad~iq/musad~iqfa	qalobika bi <seg id="2854"<i*oni{ll~ahi
2	haA fa{sotabiquwA@ {loxayora`ti >ayona maA	muwal~iy	/muwal~iy/N/muwal~iy/muwal~iy	mina {lomumotariyna walikul~K wijohapN huwa
2	{lr~iya`Hi wa{ls~aHaAbi {lomusax~ari bayona	taSoriyfi	/taSoriyf/N/tasoriyf/tasoriyfi	haA wabav~a fiyhaA min kul~i daA^b~apK wa
2	m~in r~ab~ikumo waraHomapN famani {EotadaY`	taxofiyfN	/taxofiyf/N/taxofiyf/taxofiyfn	<seg id="5503"<ilayohi bi <seg id="5506"<iHosa`nK*a`lika
2	wamun*iriyna wa>anzala maEahumu {lokita`ba	muba$~iriyna	/muba$~ir/N/muba$~ir/muba$~iriyna	wa`HidapF fabaEava {ll~ahu {ln~abiy~i.na
2	yatarab~aSona bi>anfusihin~a vala`vapa	muTal~aqa`tu	/muTal~aqa`t/N/mutal~aqa`t/mutal~aqa`tu	<seg id="7364"<in~a{ll~aha samiyEN EaliymN wa{lo
2	bi <seg id="7448"<iHosa`nKwalaA yaHil~u	tasoriyHN[	/tasoriyH/N/tasoriyh/tasoriyhn[	<seg id="7442"<imosaAkN[bimaEoruwfK >awo
2	mata`EN[bi{lomaEoruwfi Haq~FA EalaY {lo	muTal~aqa`ti	/muTal~aqa`t/N/mutal~aqa`t/mutal~aqa`ti	m~aEoruwfK wa{ll~ahu EaziyzN HakiymN walilo
2	m~ino >anfusihimo kamavali jan~apK] birabowapK	tavobiytFA	/tavobiyt/N/tavobiyt/tavobiytfa	>amowa`lahumu {botigaA^'a maroDaAti {ll~ahi wa
3	l~imaA bayona yadayohi wa>anzala {lt~aworaY`pa	muSad~iqFA	/muSad~iq/N/musad~iq/musad~iqfa	qay~uwmu naz~ala Ealayoka {lokita`ba bi{loHaq~i
3	hi. wamaA yaEolamu ta>owiylahu,^ <seg id="10403"<il~aA	ta>owiyli	/ta>owiyl/N/ta>owiyl/ta>owiyli	hu {botigaA^'a {lofitonapi wa{botigaA^'a
3	hu,^ <seg id="10403"<il~aA {ll~ahu wa{lr~a`sixuwna	ta>owiyla	/ta>owiyl/N/ta>owiyl/ta>owiyla	wa{botigaA^'a ta>owiylihi. wamaA yaEolamu
3	fataqab~alo min~iY^ <seg id="11215"<in~a	muHar~arFA	/muHar~ar/N/muhar~ar/muhar~arfa	<seg id="11200"<in~iY na*arotu laka maA fiY baToniY
3	bikalimapK m~ina {ll~ahi wasay~idFA waHaSuwrFA	muSad~iqF[A	/muSad~iq/N/musad~iq/musad~iqf[a	miHoraAbi >an~a {ll~aha yuba$~iruka biyaHoyaY`
3	wayukal~imu {ln~aAsa fiY {lomahodi wakaholFA	muqar~abiyna	/muqar~abuwn/N/muqar~abuwn/muqar~abiyna	fiY {ld~unoyaA wa{lo'aAxirapi wamina {lo
3	l~imaA bayona yadaY~a mina {lt~aworaY`pi	muSad~iqFA	/muSad~iq/N/musad~iq/musad~iqfa	<seg id="11662"<in kuntum m~u&ominiyna wa

Page 1 of 14 Go Next | Last

Figure(6.4): Testing POS filter options

Testing the Qurany concept 'Hunting':

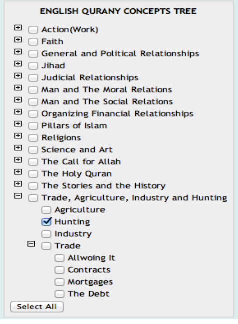
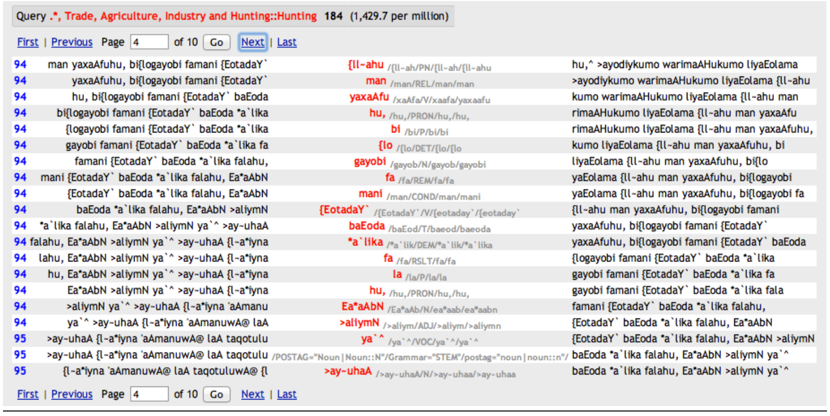

ENGLISH QURANY CONCEPTS TREE
⊞ □ Action(Work)
⊞ □ Faith
⊞ □ General and Political Relationships
⊞ □ Jihad
⊞ □ Judicial Relationships
⊞ □ Man and The Moral Relations
⊞ □ Man and The Social Relations
⊞ □ Organizing Financial Relationships
⊞ □ Pillars of Islam
⊞ □ Religions
⊞ □ Science and Art
⊞ □ The Call for Allah
⊞ □ The Holy Quran
⊞ □ The Stories and the History
⊟ □ Trade, Agriculture, Industry and Hunting
□ Agriculture
☑ Hunting
□ Industry
⊟ □ Trade
□ Allwoing It
□ Contracts
□ Mortgages
□ The Debt
Select All

Figure (6.5): Qurany concepts search options

The query was also made by CQL [word='.*']. By clicking on the Hunting concept on Qurany with the concept 'tree', the following result was produced:



Query .*, Trade, Agriculture, Industry and Hunting::Hunting 184 (1,429.7 per million)

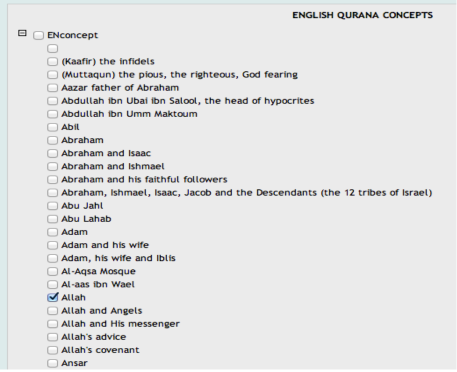
94	man yaxaAfuhu, bi{logayobi famani {EotadaY`	{ll~ahu /{ll~ah/PN/{ll~ah/{ll~ahu	hu,^ >ayodiykumo warimaAHukumo liyaEolama
94	yaxaAfuhu, bi{logayobi famani {EotadaY`	man /man/REL/man/man	>ayodiykumo warimaAHukumo liyaEolama {ll~ahu
94	hu, bi{logayobi famani {EotadaY` baEoda	yaxaAfu /xaAfa/V/xaafa/yaxaafu	kumo warimaAHukumo liyaEolama {ll~ahu man
94	bi{logayobi famani {EotadaY` baEoda *a`lika	hu, /hu,/PRON/hu,/hu,	rimaAHukumo liyaEolama {ll~ahu man yaxaAfu
94	{logayobi famani {EotadaY` baEoda *a`lika	bi /bi/P/bi/bi	rimaAHukumo liyaEolama {ll~ahu man yaxaAfuhu,
94	gayobi famani {EotadaY` baEoda *a`lika fa	{lo /{lo/DET/{lo/{lo	kumo liyaEolama {ll~ahu man yaxaAfuhu, bi
94	famani {EotadaY` baEoda *a`lika falahu,	gayobi /gayob/N/gayob/gayobi	liyaEolama {ll~ahu man yaxaAfuhu, bi{lo
94	mani {EotadaY` baEoda *a`lika falahu, Ea*aAbN	fa /fa/REM/fa/fa	yaEolama {ll~ahu man yaxaAfuhu, bi{logayobi
94	{EotadaY` baEoda *a`lika falahu, Ea*aAbN	mani /man/COND/man/mani	yaEolama {ll~ahu man yaxaAfuhu, bi{logayobi fa
94	baEoda *a`lika falahu, Ea*aAbN >aliymN	{EotadaY` /{EotadaY`/V/{eotaday`/{eotaday`	{ll~ahu man yaxaAfuhu, bi{logayobi famani
94	*a`lika falahu, Ea*aAbN >aliymN ya`^ >ay~uhaA	baEoda /baEod/T/baeod/baeoda	yaxaAfuhu, bi{logayobi famani {EotadaY`
94	falahu, Ea*aAbN >aliymN ya`^ >ay~uhaA {l~a*iyna	*a`lika /*a`lik/DEM/*a`lik/*a`lika	yaxaAfuhu, bi{logayobi famani {EotadaY` baEoda
94	lahu, Ea*aAbN >aliymN ya`^ >ay~uhaA {l~a*iyna	fa /fa/RSLT/fa/fa	{logayobi famani {EotadaY` baEoda *a`lika
94	hu, Ea*aAbN >aliymN ya`^ >ay~uhaA {l~a*iyna	la /la/P/la/la	gayobi famani {EotadaY` baEoda *a`lika fa
94	Ea*aAbN >aliymN ya`^ >ay~uhaA {l~a*iyna	hu, /hu,/PRON/hu,/hu,	gayobi famani {EotadaY` baEoda *a`lika fala
94	>aliymN ya`^ >ay~uhaA {l~a*iyna 'aAmanu	Ea*aAbN /Ea*aAb/N/ea*aab/ea*aabn	famani {EotadaY` baEoda *a`lika falahu,
94	ya`^ >ay~uhaA {l~a*iyna 'aAmanuwA@ laA	>aliymN />aliym/ADJ/>aliym/>aliymn	{EotadaY` baEoda *a`lika falahu, Ea*aAbN
95	>ay~uhaA {l~a*iyna 'aAmanuwA@ laA taqotulu	ya`^ /ya`^/VOC/ya`^/ya`^	{EotadaY` baEoda *a`lika falahu, Ea*aAbN >aliymN
95	>ay~uhaA {l~a*iyna 'aAmanuwA@ laA taqotulu	/POSTAG="Noun|Noun::N"/Grammar="STEM"/postag="noun|noun::n"/	baEoda *a`lika falahu, Ea*aAbN >aliymN ya`^
95	{l~a*iyna 'aAmanuwA@ laA taqotuluwA@ {l	>ay~uhaA />ay~uhaA/N/>ay~uhaa/>ay~uhaa	baEoda *a`lika falahu, Ea*aAbN >aliymN ya`^

Figure (6.6): The query outputs

The result shows how many segments are related to hunting. However, the actual verses id can be viewed through (view option) and by changing the view to verse number. There are four verses; verse 670, 763, 764 and 765. Comparing these with the XML evaluation result using Xpath demonstrated that the result is accurate and correct.

Testing the Pronoun reference to Allah:

By clicking on the concept Allah and retrieving all words, the result showed 3,061 occurrences of pronouns referring to the Allah concept. Thus, the SKE tool was shown to retrieve accurate results.



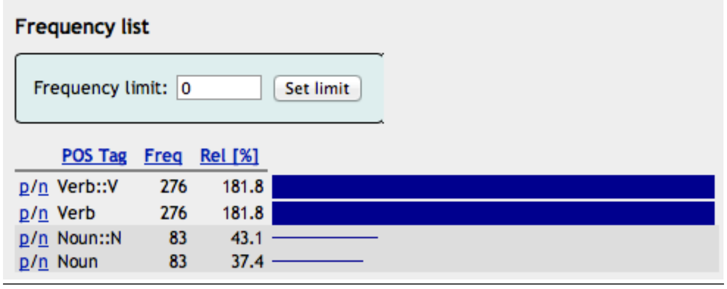
ENGLISH QURANA CONCEPTS

⊟ □ ENconcept
- □
- □ (Kaafir) the infidels
- □ (Muttaqun) the pious, the righteous, God fearing
- □ Aazar father of Abraham
- □ Abdullah ibn Ubai ibn Salool, the head of hypocrites
- □ Abdullah ibn Umm Maktoum
- □ Abil
- □ Abraham
- □ Abraham and Isaac
- □ Abraham and Ishmael
- □ Abraham and his faithful followers
- □ Abraham, Ishmael, Isaac, Jacob and the Descendants (the 12 tribes of Israel)
- □ Abu Jahl
- □ Abu Lahab
- □ Adam
- □ Adam and his wife
- □ Adam, his wife and Iblis
- □ Al-Aqsa Mosque
- □ Al-aas ibn Wael
- ☑ Allah
- □ Allah and Angels
- □ Allah and His messenger
- □ Allah's advice
- □ Allah's covenant
- □ Ansar

Figure(6.7): QurAna concepts search options

Using Sketch Engine features:

- ***Word frequency:***
- Occurrences of the word Eml (work in English) was tested in the corpus. To check the POS tags frequencies of Eml, the Frequency Option was tested, and then the frequency criterion was tested with POS tag. The result is shown in the figure (6.8) below.



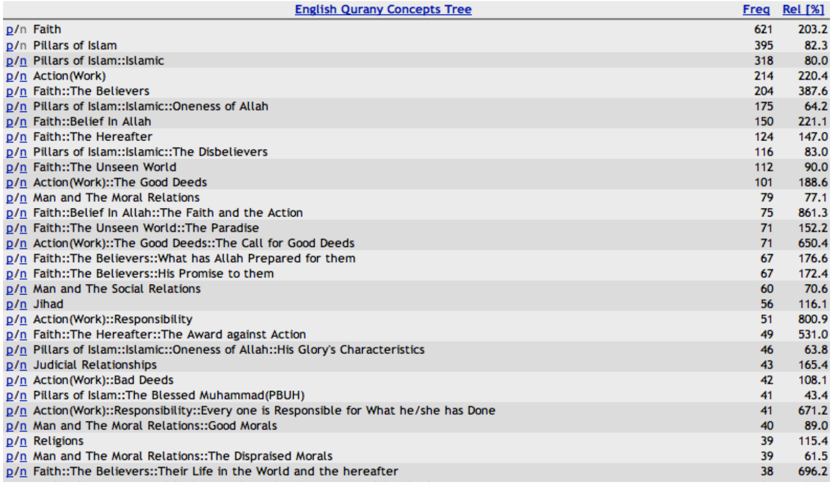
Frequency list

Frequency limit: 0 Set limit

	POS Tag	Freq	Rel [%]
p/n	Verb::V	276	181.8
p/n	Verb	276	181.8
p/n	Noun::N	83	43.1
p/n	Noun	83	37.4

Figure(6.8): The frequency of Eml POS tags

- ***Concept frequency for a word:***
- To view the concept frequency for the word Eml the frequency option was used but in conjunction with the Qurany concept criteria.

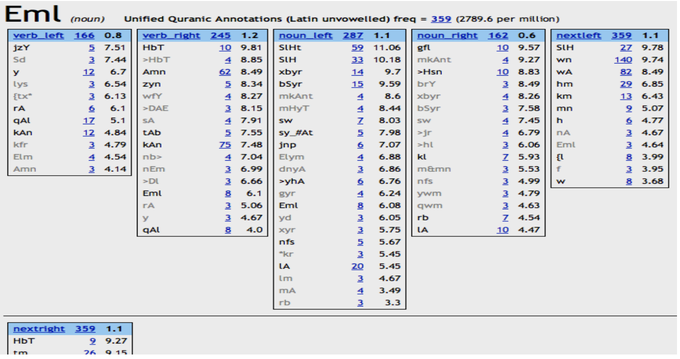


	English Qurany Concepts Tree	Freq	Rel [%]
p/n	Faith	621	203.2
p/n	Pillars of Islam	395	82.3
p/n	Pillars of Islam::Islamic	318	80.0
p/n	Action(Work)	214	220.4
p/n	Faith::The Believers	204	387.6
p/n	Pillars of Islam::Islamic::Oneness of Allah	175	64.2
p/n	Faith::Belief In Allah	150	221.1
p/n	Faith::The Hereafter	124	147.0
p/n	Pillars of Islam::Islamic::The Disbelievers	116	83.0
p/n	Faith::The Unseen World	112	90.0
p/n	Action(Work)::The Good Deeds	101	188.6
p/n	Man and The Moral Relations	79	77.1
p/n	Faith::Belief In Allah::The Faith and the Action	75	861.3
p/n	Faith::The Unseen World::The Paradise	71	152.2
p/n	Action(Work)::The Good Deeds::The Call for Good Deeds	71	650.4
p/n	Faith::The Believers::What has Allah Prepared for them	67	176.6
p/n	Faith::The Believers::His Promise to them	67	172.4
p/n	Man and The Social Relations	60	70.6
p/n	Jihad	56	116.1
p/n	Action(Work)::Responsibility	51	800.9
p/n	Faith::The Hereafter::The Award against Action	49	531.0
p/n	Pillars of Islam::Islamic::Oneness of Allah::His Glory's Characteristics	46	63.8
p/n	Judicial Relationships	43	165.4
p/n	Action(Work)::Bad Deeds	42	108.1
p/n	Pillars of Islam::The Blessed Muhammad(PBUH)	41	43.4
p/n	Action(Work)::Responsibility::Every one is Responsible for What he/she has Done	41	671.2
p/n	Man and The Moral Relations::Good Morals	40	89.0
p/n	Religions	39	115.4
p/n	Man and The Moral Relations::The Dispraised Morals	39	61.5
p/n	Faith::The Believers::Their Life in the World and the hereafter	38	696.2

Figure (6.9): Concepts frequency for word Eml

The output showed that the word Eml is mentioned in the Quran but associated with the concept of 'faith'. The word Eml is associated with Faith 621 times, although there are only 359 occurrences of Eml. However, since a verse could have more than one Qurany concept, the faith concept is repeated with the same word occurrence. Thus, a relation between Eml (which means work in English) and Faith as a word related to human faith can be made.

- ***Word Sketch:***
- Word Sketch is used to show a word's grammatical and collocation behaviour in the corpus.



Eml (noun) Unified Quranic Annotations (Latin unvowelled) freq = 359 (2789.6 per million)

verb_left	166	0.8
jzY	5	7.51
Sd	3	7.44
y	12	6.7
lys	3	6.54
{tx*	3	6.13
rA	6	6.1
qAl	17	5.1
kAn	12	4.84
kfr	3	4.79
Elm	4	4.54
Amn	3	4.14

verb_right	245	1.2
HbT	10	9.81
>HbT	4	8.85
Amn	62	8.49
zyn	5	8.34
wfY	4	8.27
>DAE	3	8.15
sA	4	7.91
tAb	5	7.55
kAn	75	7.48
nb>	4	7.04
nEm	3	6.99
>Dl	3	6.66
Eml	8	6.1
rA	3	5.06
y	3	4.67
qAl	8	4.0

noun_left	287	1.1
SlHt	59	11.06
SlH	33	10.18
xbyr	14	9.7
bSyr	15	9.59
mkAnt	4	8.6
mHyT	4	8.44
sw	7	8.03
sy_#At	5	7.98
jnp	6	7.07
Elym	4	6.88
dnyA	3	6.86
>yhA	6	6.76
gyr	4	6.24
Eml	8	6.08
yd	3	6.05
xyr	3	5.75
nfs	5	5.67
*kr	3	5.45
lA	20	5.45
lm	3	4.67
mA	4	3.49
rb	3	3.3

noun_right	162	0.6
gfl	10	9.57
mkAnt	4	9.27
>Hsn	10	8.83
brY	3	8.49
xbyr	4	8.26
bSyr	3	7.58
sw	4	7.45
>jr	4	6.79
>hl	3	6.06
kl	7	5.93
m&mn	3	5.53
nfs	3	4.99
ywm	3	4.79
qwm	3	4.63
rb	7	4.54
lA	10	4.47

nextleft	359	1.1
SlH	27	9.78
wn	140	9.74
wA	82	8.49
hm	29	6.85
km	13	6.43
mn	9	5.07
h	6	4.77
nA	3	4.67
Eml	3	4.64
{l	8	3.99
f	3	3.95
w	8	3.68

nextright	359	1.1
HbT	9	9.27
tm	26	9.15

Figure(6.10): Word Sketches

Users-Evaluation:

The project aims to help three main user types: Quranic researchers, Arabic language and linguistics experts and Arabic language students.

Dr. Ahlam Aldhubayi **is a** Quranic researcher from Imam Muhammad bin Saud Islamic University in Saudi Arabia. Her research aimed to investigate the explicit and implicit occurrences of the word 'mind' (Eql). Her aim was to perform a comprehensive analysis for the word to indicate a religious pattern of how the Quran uses the word (Eql) and which concepts are shared. She investigated every occurrence of the word (Eql) in Sketch Engine and explored their verse concepts. However, it took her only a short time to find comprehensive analyses for each occurrence in the Quran. She stated in her evaluation that "there is a demand for a Quranic tool to help in studying Quranic word behaviour and to analyse such behaviour comprehensively". Moreover, there is a need for an accurate source showing the Quranic words associated with their grammatical and semantic features. In addition, classifying the words with their grammatical features will help to indicate general patterns and collocations as well.

Dr. Nawal Alhelwa **is** an Arabic linguistic researcher in Princess Noura University, Kingdom of Saudi Arabia. She evaluated the unvowelled Arabic corpus by querying the word 'scourge' (عذاب).Her search aimed to find the semantic concepts related to the word. She stated in her evaluation "the system and the dataset are useful for linguistic researches except for some limitation in the syntax annotations". The syntax annotations do not show the Indicative, Subjunctive and Jussive in detail. Usually, Arabic language researchers aim to find the reason behind verbal moods such as these.

Dr. Amany Altawili is a Quranic researcher from Imam Muhammad bin Saud Islamic University. Currently, she is doing new research to study the behavior of some Quranic words. She evaluated the concordance of the Quranic word 'prefer'(فضل). She stated in her evaluation that "the system is good and it can serve the demands of Quranic researchers".

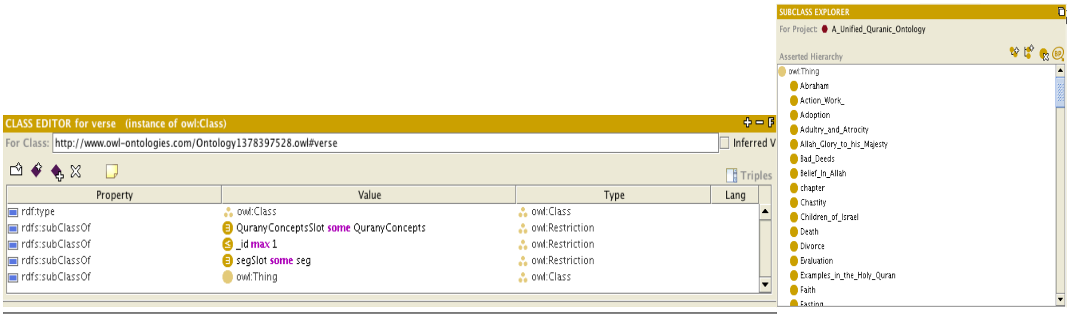
Mr. Abdullah Alfaifi **is a** Lecturer in TAFL Teaching Arabic as Foreign Language, and PhD student. He evaluated the unvowelled Arabic corpus and made queries to find the pronoun (k) (ك) and its grammar features, such as the feminine and masculine. The system offers a variety of analysis options. He found the corpus beneficial for Arabic language researchers and stated "the corpus seems to have well-structured design which provide useful features (PoS, Syntactic, and Semantic) that enable users to search in the Quranic text for different purposes, Quranic and linguistic studies."

However, the brief presentation (manual) provided to the evaluators needs to be expanded to cover more details in the system. (The evaluation is in Appendix L)

Mr. Mishal Alhusan **is an** Arabic language teacher in the Ministry of Education in Saudi Arabia. He evaluated the usage of the word ‘لتاقي’ in relation to Jihad concepts. He was able to cluster the Jihad concepts words based on the Gender type and the POS tags.

6.3 Evaluating the unified Quranic OWL ontology:

The class triples and Class hierarchy:



CLASS EDITOR for verse (instance of owl:Class)

For Class: http://www.owl-ontologies.com/Ontology1378397528.owl#verse

☐ Inferred V

Triples

Property	Value	Type	Lang
rdf:type	owl:Class	owl:Class	
rdfs:subClassOf	QuranyConceptsSlot some QuranyConcepts	owl:Restriction	
rdfs:subClassOf	_id max 1	owl:Restriction	
rdfs:subClassOf	segSlot some seg	owl:Restriction	
rdfs:subClassOf	owl:Thing	owl:Class	

SUBCLASS EXPLORER

For Project: ● A_Unified_Quranic_Ontology

Asserted Hierarchy

- owl:Thing
  - Abraham
  - Action_Work_
  - Adoption
  - Adultry_and_Atrocity
  - Allah_Glory_to_his_Majesty
  - Bad_Deeds
  - Belief_In_Allah
  - chapter
  - Chastity
  - Children_of_Israel
  - Death
  - Divorce
  - Evaluation
  - Examples_in_the_Holy_Quran
  - Faith
  - Fasting

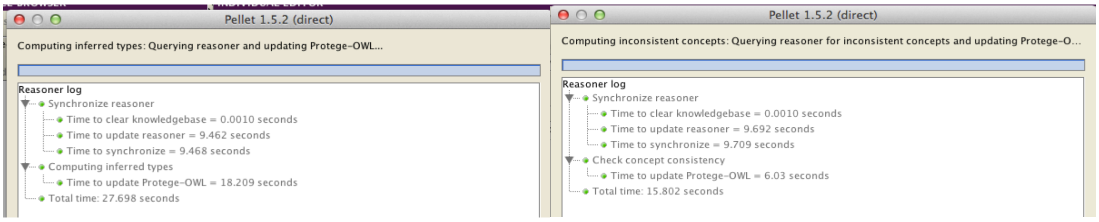
Figure (6.11): The Unified Quranic Ontology in Protégé .

Unfortunately, the size of the ontology is very large (135MB), therefore the heap size of the virtual machine in Protégé requires a greatly increased memory size, to as much as 5,000MB because the reasoning of the ontology means creating more inferred instances, which occupy more memory. Therefore, it is necessary to increase the heap size to the whole required memory.

However, Protégé still cannot run the reasoning of a large ontology. This takes several hours without any noticeable progress in the reason log. To evaluate the mapped ontology, the domain was limited to the second chapter only in the Quran. Chapter 2 is considered to be the longest chapter in Quranic with a maximum of 280 verses. Thus, this chapter could reflect and cover a lot of concepts and grammar annotations.

So, it was necessary to run the semantic reasoner (Pellet 1.5.2) on the Quran’s chapter 2. Pellet and DIG are the standard reasoners in Protégé 3.5. But, DIG suffers from some reasoning limitations, such as the inability to load the whole ontology (Protégé)

The resoner checked the consistency and the inferred instances, the log shows in figure (6.12) has successfully checked the ontology.



Pellet 1.5.2 (direct)

Computing inferred types: Querying reasoner and updating Protege-OWL...

```
Reasoner log
▼ Synchronize reasoner
    Time to clear knowledgebase = 0.0010 seconds
    Time to update reasoner = 9.462 seconds
    Time to synchronize = 9.468 seconds
▼ Computing inferred types
    Time to update Protege-OWL = 18.209 seconds
  Total time: 27.698 seconds
```

Pellet 1.5.2 (direct)

Computing inconsistent concepts: Querying reasoner for inconsistent concepts and updating Protege-O...

```
Reasoner log
▼ Synchronize reasoner
    Time to clear knowledgebase = 0.0010 seconds
    Time to update reasoner = 9.692 seconds
    Time to synchronize = 9.709 seconds
▼ Check concept consistency
    Time to update Protege-OWL = 6.03 seconds
  Total time: 15.802 seconds
```

Figure(6.12): Computing the inferred instance(left) and checking ontology consistency (right).

Some Observations:

The hierarchy of subclasses in the class browser has the same level. For example, verse and Qurany concepts have the same level in the browser but the verse triple shows that the Qurany concepts class is a subclass of verse. Thus, the class browser does not reflect the actual class hierarchy.

When porting the XML file to Protégé through XML Tab, Protégé gave each instance a generated ID and used it to refer to the instance. The generated ID has a different indexing style where chapter id 1 is translated to 67. In the unified XML dataset, chapter ids have there strictly order. For future enhancement, a chapter id can be replaced by the chapter's name. Thus, having an id for a chapter will not change its name.

The properties types have not been specified, thus manual adjustment has to be made to object properties. By making an object property a transitive property, a new knowledge can be inferred.

Testing the OWL ontology using SPARQL on chapter 2:

Protocol and RDF Query Language (SPARQL) is a questioning language which can search in RDF triples and get the answers. To test the ontology, the following questions were used to look for the answers in the ontology. However, the answers were already known.

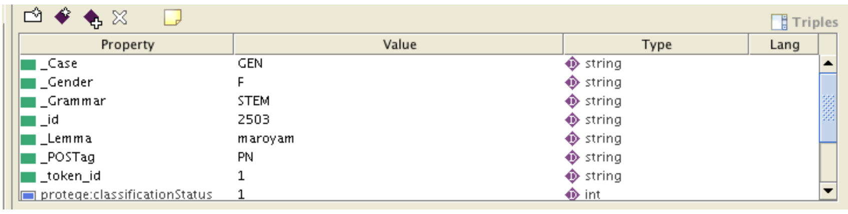
Question1: A women who has been mentioned in chapter 2?

Answer: Mary (Maryam).

SPARQL:

SELECT ?seg

WHERE { ?verse :QuranyConceptsSlot ?Women. ?seg :_POSTag "PN". ?seg :_Gender "F"}



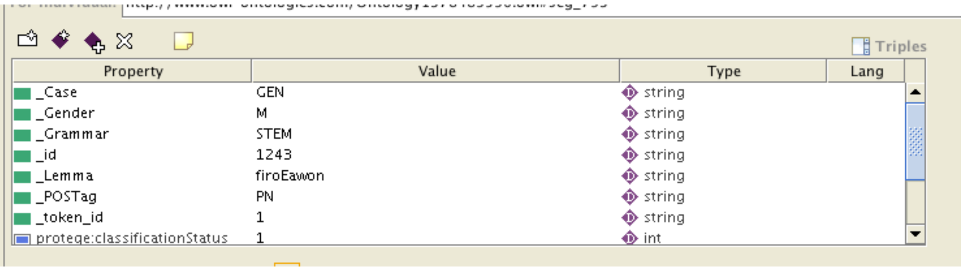
Property	Value	Type	Lang
🟩 _Case	GEN	string	
🟩 _Gender	F	string	
🟩 _Grammar	STEM	string	
🟩 _id	2503	string	
🟩 _Lemma	maroyam	string	
🟩 _POSTag	PN	string	
🟩 _token_id	1	string	
🟦 protege:classificationStatus	1	int	

Question2: Who is the man who lost his deeds in chapter2?

Answer: Fir'aun' (Pharaoh)

SPARQL:

```
SELECT ?seg
WHERE {?verse  :QuranyConceptsSlot ?Lost_Deeds. ?seg :_POSTag "PN". ?seg :_Gender 'M'}
```



Property	Value	Type	Lang
🟩 _Case	GEN	string	
🟩 _Gender	M	string	
🟩 _Grammar	STEM	string	
🟩 _id	1243	string	
🟩 _Lemma	firoEawon	string	
🟩 _POSTag	PN	string	
🟩 _token_id	1	string	
protege:classificationStatus	1	int	

The ontology answers the questions correctly. Although the semantic was assigned to the verses only, the segments inherit the verses concepts. This ontology could be used in a question-and-answer system, where a question is translated to SPRAQL to obtain the answer.

Chapter 7: Conclusion

This project describes the underlying approach of creating Unified Quranic Corpus and Ontology. Three selected datasets were merged in one file in unified format successfully. The Unified XML corpus has been loaded to Sketch engine and been used successfully and efficiently. The evaluators have positive feedback about the dataset and the tool. Thus, Sketch engine is a perfect in dealing with huge datasets and has a robust search tool and options.

In addition, The unified XML dataset has been converted to OWL ontology. The Unified Quranic ontology has been loaded to Protégé ontology editor. But, protégé is not capable of handling large ontology such the entire Quranic ontology with more than 100 MB. Obviously, protégé is perfect tool in processing small ontology.

Overall, Sketch engine and protégé tools have different functions and aims and thus can not be compared. But, Sketch engine outperforms protégé in processing huge datasets, and has effective website that enable sharing the resource. In the other hand, protégé is a stand-alone tool where it has the facility of sharing but with still has limitations.

Ideas for further work:

This project can be extended to have more annotations to be shared through sketch engine tool. Another enhancement is to merge the QAC search interface with Sketch engine to make the merged Quran dataset interfaced more user-friendly. In addition, a follow-on project for another religious text, such as the Bible or the Book of Mormon or a collection of Haddith.

References:

Abbas, N. 2009. Qurany ‘Search for a Concept’ Tool .[online]Available at: http://www.comp.leeds.ac.uk/nora.

Abbas, N., Aldhubayi, L., Al-Khalifa, H., Alqassem, Z., Atwell, E., Dukes, K., Sawalha, M., Sharaf, A., (2013). Unifying linguistic annotations and ontologies for the Arabic Quran. The WACL’2 Second Workshop on Arabic Corpus Linguistics, Lancaster University, UK.

Al-Khalifa, H., Al-Yahya, M. Bahanshal, A. and Al-Odah I. (2009). SemQ: A Proposed Framework for Representing Semantic Opposition in the Holy Quran using Semantic Web Technologies. The 2009 International conference on the Current Trends in Information Technology (CTIT’09), Dubai, UAE. 15-16 December 2009.

Al-Yahya, M., Al-Khalifa, H., Bahanshal, A., Al-Odah, I., & Al-Helwah, N. (2010, July). An ontological Model for Representing Semantic Lexicons: An Application on Time Nouns in the Holy Quran. Arabian Journal for Science and Engineering, 35(2), 21.

Alani, H., Hara, K.O., Shadbolt, N., (2005), “Common features of killer apps: A comparison with Protégé”, In: 8th International Prot Conference, 18-21 July 2005, Madrid, Spain.

Alatrish, E. (2012), Comparison of Ontology Editors, eRAF Journal on Computing, University of Belgrade, Serbia, Vol. 4,

Alqassem, Z., (2013). Unifying Quranic Analyses into a Single Database. Final year project report, University of Leeds.

Antoniou, G. (2004). A semantic web primer. (pp. 18-31). MIT Press.

Antoniou, G., & Van Harmelen, F. (2009). Web ontology language: Owl. In Handbook on ontologies (pp. 91-110). Springer Berlin Heidelberg.

Bohring, H., & Auer, S. (2005). Mapping XML to OWL Ontologies. Leipziger Informatik-Tage, 72, 147-156.

Decker, S., Melnik, S., Van Harmelen, F., Fensel, D., Klein, M., Broekstra, J.,& Horrocks, I. (2000). The semantic web: The roles of XML and RDF. Internet Computing, IEEE, 4(5), 63-73.

Dror, J., Shaharabani, D., Talmon, R., & Wintner, S. (2004). Morphological Analysis of the Qur'an. Literary and linguistic computing, 19(4), 431-452.

Dukes, K., & Atwell, E. (2012). LAMP: A Multimodal Web Platform for Collaborative Linguistic Analysis. In Proceedings of LREC.

Dukes, K., & Habash, N. (2010, May). Morphological annotation of quranic Arabic. In Proceedings of the Language Resources and Evaluation Conference (LREC).

Dukes, K., Atwell, E., & Habash, N. (2011). Supervised collaboration for syntactic annotation of Quranic Arabic. Language Resources and Evaluation, 1-30.

Element Tree XML API – Python documentations. [Online]. Available: http://docs.python.org/2/library/xml.etree.elementtree.html

Garabík, R. (2006). Processing XML Text with Python and ElementTree–a Practical Experience. INSIGHT INTO THE SLOVAK AND CZECH CORPUS LINGUISTICS, 160.

Gruber, T. (2008). What is an Ontology. Encyclopedia of Database Systems, 1.

Habash, Y. B. N., Ordan, A. K. N., & Suchomel, R. R. V. (2013) arTenTen: a new, vast corpus for Arabic.

Khan, H. U., Saqlain, S. M., Shoaib, M., & Sher, M. (2013), Ontology Based Semantic Search in Holy Quran, International Journal of future computer and communication, Vol 2, No. 6.

Kilgarriff, A., Rychly, P., Smrz, P., & Tugwell, D. (2004). ITRI-04-08 The Sketch Engine. Information Technology, 105, 116.

Lassila, O., & Swick, R. R. (1999). Resource description framework (RDF) model and syntax specification.

Lenzerini, M. (2002, June). Data integration: A theoretical perspective. In Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (pp. 233-246). ACM.

Motik, B., Parsia, B., & Patel-Schneider, P. F. (2009). OWL 2 Web Ontology Language XML serialization. W3C Recommendation, W3C–World Wide Web Consortium.

Noy, N. F., & McGuinness, D. L. (2001). Ontology development 101: A guide to creating your first ontology.

Prechelt, L. (2000). An empirical comparison of C, C++, Java, Perl, Python, Rexx and Tcl. IEEE Computer, 33(10), 23-29.

Protégé Ontology Library - Protégé Wiki. [Online]. Available: http://protegewiki.stanford.edu/wiki/Protege_Ontology_Library

Rodrigues, T., Rosa, P., & Cardoso, J. (2006). Mapping XML to Exiting OWL ontologies. In International Conference WWW/Internet (pp. 72-77).

Sha‘rāwī, M. M. (1993). Tafsīr al-Sha‘rāwī (Vol. 3). Akhbār al-Yawm. PP(5-15).

Sharaf, A. B. M., & Atwell, E. (2012). QurAna: Corpus of the Quran annotated with Pronominal Anaphora. LREC 2012.

Sharaf, A. B. M., & Atwell, E. (2012). QurSim: A corpus for evaluation of relatedness in short texts. LREC 2012.

Van Deursen, D., Poppe, C., Martens, G., Mannens, E., & Walle, R. (2008, November). XML to RDF conversion: a generic approach. In Automated solutions for Cross Media Content and Multi-channel Distribution, 2008. AXMEDIS'08. International Conference on (pp. 138-144). IEEE.

Yahia, N., Mokhtar, S. A., & Ahmed, A. (2012). Automatic Generation of OWL Ontology from XML Data Source. arXiv preprint arXiv:1206.0570.

Yauri, A. R., Kadir, R. A., Azman, A., & Murad, M. A. A. (2012, March). Quranic-based concepts: Verse relations extraction using Manchester OWL syntax. In Information Retrieval & Knowledge Management (CAMP), 2012 International Conference on (pp. 317-321). IEEE.

Appendix A: Personal Reflection

Undertaking this project was an adventure and a complex journey, full of ups and downs. Through the period of 8 months working on this project, many challenges have been faced. One of the main challenges was the project framework required to achieve the project objectives. As the project revolved around addressing research problems by semantic web/text search and combined two research areas: neutral language processing and knowledge representation and reasoning, thus, intensive reading of background research on the semantic web and data models was involved. In addition, obtaining in great detail knowledge of the building and the evaluation of XML files has been a great experience and given me a better understanding of this field.

The biggest challenge I went through working on this project was to have a clear aim and understanding of the problem that needed to be tackled. At the beginning, there was an unclear appreciation of the difference between corpora and ontology. The project title was ‘A unified Quranic ontologies’ when, in fact, there are few available Quranic ontologies which did not meet the project goal. As a result, the direction followed was to deal with the proposed three datasets that cannot be considered as ontology but rather as an annotated corpus or dataset because they are not in a standard ontology data model such as first order logic or RDF. However, during the interim report The project assessor Dr. Lau has provided valuable feedback on the project and its value. Her feedback pushed the project in a clearer direction, which resulted in changing the project title and plan. Thus, the project title was changed to ‘unified Quranic annotations and ontologies’ in order to accomplish the first goal of merging the datasets in one dataset or annotated XML corpus. The new goal was to convert the merged dataset to an ontology standard data model. As a result of changing the title, the project plan was altered as well in order to consider corpora and ontology as two main parts of the project.

In addition, during term two, I suffered a broken foot. The illness prevented me walking normally for three weeks. This added more pressure since I’m a single mother and have four years old daughter to take care of her. Despite the difficulties that I had, the supervisor Dr. Eric Atwell and the school of computing in the university were supportive, sympathetic and reassuring. So, if you have any personal problems or constraints which affect your project work, seek advice from Supervisor or Director of Student Education rather than struggling on alone.

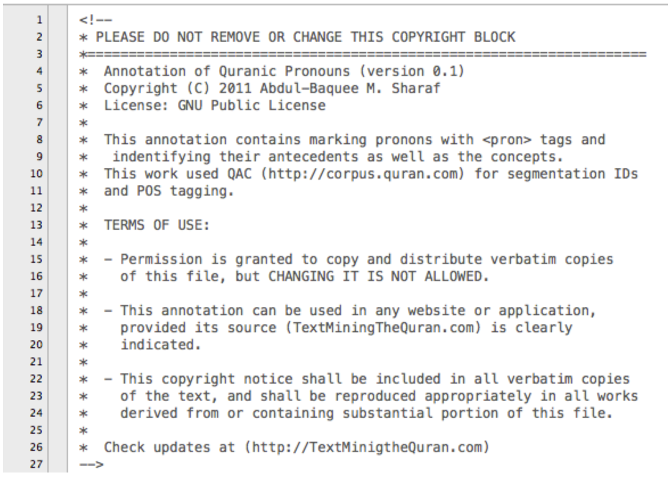
My advice to fellow students and researchers to believe in them selves and never to give up and to always come back to their supervisor if there are any doubts.

In addition, it was a pleasure to be a co-author in presenting the paper 'the Unifying linguistic annotations and ontologies for the Arabic Quran) in WACL'2 Second Workshop on Arabic Corpus Linguistics, Lancaster University in July 2013'. This experience provided me on techniques of presenting an academic paper and I look forward for more to writing more papers.

Finally, in the past year I have continuously developed, the knowledge and experience I have gained through the project is worthwhile. It has deepen my understanding in language processing and semantic search. I truly enjoyed spending my time on it. It is hoped that this project will shed some light in this field of research and provide future researcher with a useful starting point.

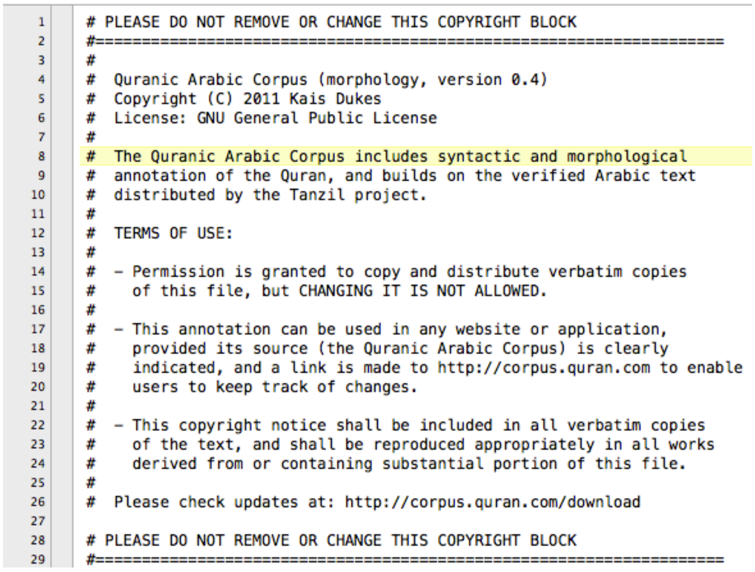
Appendix B: Material Used.

This project has used three datasets without changing their content. As the copyright in Quranic Arabic Corpus and QurAna datasets permit the right to use the data without add changes.



```
<!--
* PLEASE DO NOT REMOVE OR CHANGE THIS COPYRIGHT BLOCK
*=================================================================
*  Annotation of Quranic Pronouns (version 0.1)
*  Copyright (C) 2011 Abdul-Baquee M. Sharaf
*  License: GNU Public License
*
*  This annotation contains marking pronons with <pron> tags and
*   indentifying their antecedents as well as the concepts.
*  This work used QAC (http://corpus.quran.com) for segmentation IDs
*  and POS tagging.
*
*  TERMS OF USE:
*
*  - Permission is granted to copy and distribute verbatim copies
*    of this file, but CHANGING IT IS NOT ALLOWED.
*
*  - This annotation can be used in any website or application,
*    provided its source (TextMiningTheQuran.com) is clearly
*    indicated.
*
*  - This copyright notice shall be included in all verbatim copies
*    of the text, and shall be reproduced appropriately in all works
*    derived from or containing substantial portion of this file.
*
*  Check updates at (http://TextMinigtheQuran.com)
-->
```

QurAna Copyright



```
# PLEASE DO NOT REMOVE OR CHANGE THIS COPYRIGHT BLOCK
#====================================================================
#
#  Quranic Arabic Corpus (morphology, version 0.4)
#  Copyright (C) 2011 Kais Dukes
#  License: GNU General Public License
#
#  The Quranic Arabic Corpus includes syntactic and morphological
#  annotation of the Quran, and builds on the verified Arabic text
#  distributed by the Tanzil project.
#
#  TERMS OF USE:
#
#  - Permission is granted to copy and distribute verbatim copies
#    of this file, but CHANGING IT IS NOT ALLOWED.
#
#  - This annotation can be used in any website or application,
#    provided its source (the Quranic Arabic Corpus) is clearly
#    indicated, and a link is made to http://corpus.quran.com to enable
#    users to keep track of changes.
#
#  - This copyright notice shall be included in all verbatim copies
#    of the text, and shall be reproduced appropriately in all works
#    derived from or containing substantial portion of this file.
#
#  Please check updates at: http://corpus.quran.com/download

# PLEASE DO NOT REMOVE OR CHANGE THIS COPYRIGHT BLOCK
#====================================================================
```

Quranic Arabic Corpus copyright

Appendix C: Ethical issues

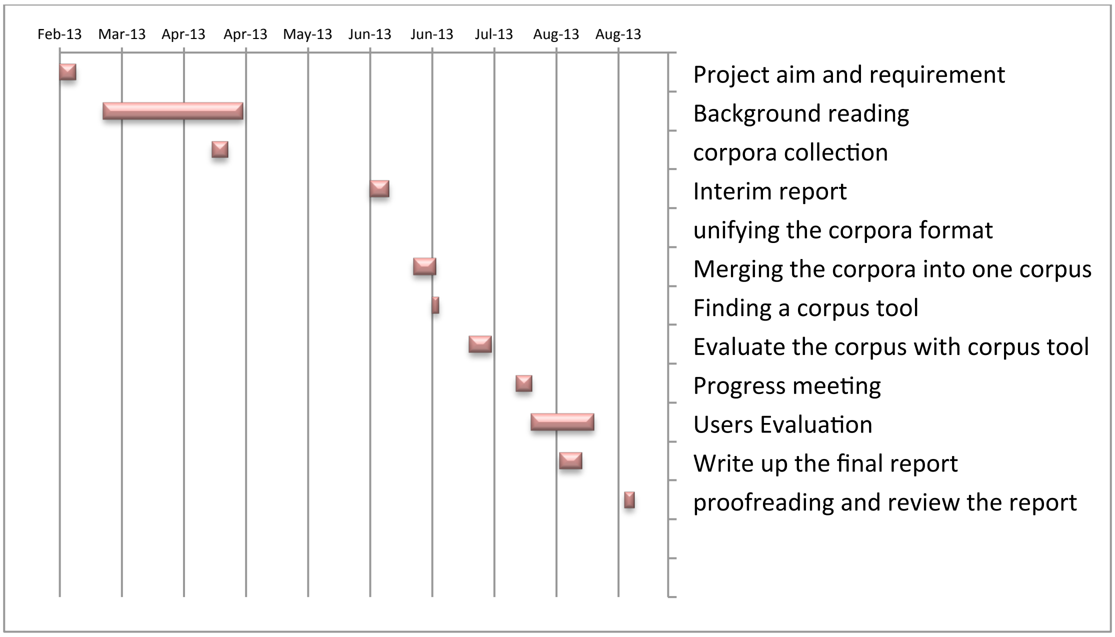
The project doesn’t contain any private or personal data and it is out of ethical issues.

Appendix D: project plan

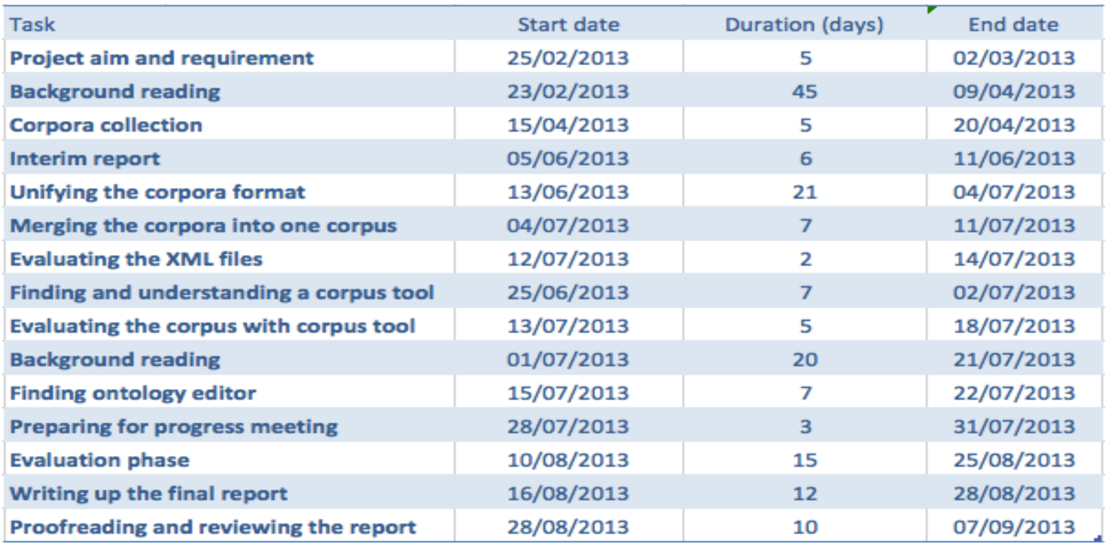
The initial project plan:

Task	Start date	Duration (days)	End date
Project aim and requirement	25/02/2013	5	02/03/2013
Background reading	11/03/2013	45	25/04/2013
corpora collection	15/04/2013	5	20/04/2013
Interim report	05/06/2013	6	11/06/2013
unifying the corpora format	16/06/2003	21	07/07/2003
Merging the corpora into one corpus	19/06/2013	7	26/06/2013
Finding a corpus tool	25/06/2013	2	27/06/2013
Evaluate the corpus with corpus tool	07/07/2013	7	14/07/2013
Progress meeting	22/07/2013	5	27/07/2013
Users Evaluation	27/07/2013	20	16/08/2013
Write up the final report	05/08/2013	7	12/08/2013
proofreading and review the report	26/08/2013	3	29/08/2013

Gantt Chart:

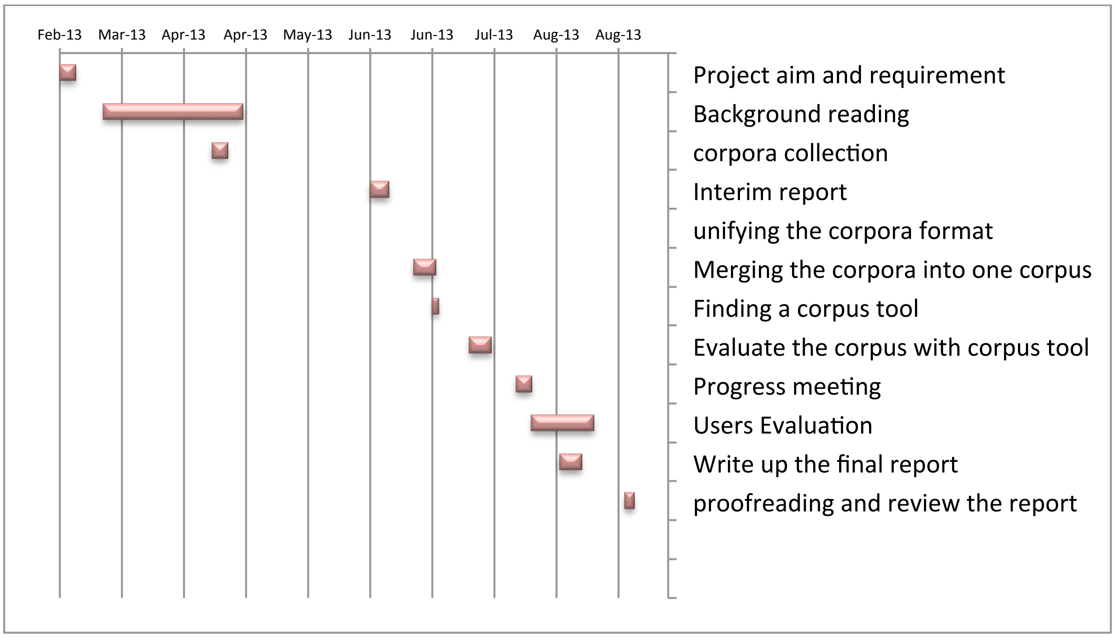


The adjusted project plan:



Task	Start date	Duration (days)	End date
Project aim and requirement	25/02/2013	5	02/03/2013
Background reading	23/02/2013	45	09/04/2013
Corpora collection	15/04/2013	5	20/04/2013
Interim report	05/06/2013	6	11/06/2013
Unifying the corpora format	13/06/2013	21	04/07/2013
Merging the corpora into one corpus	04/07/2013	7	11/07/2013
Evaluating the XML files	12/07/2013	2	14/07/2013
Finding and understanding a corpus tool	25/06/2013	7	02/07/2013
Evaluating the corpus with corpus tool	13/07/2013	5	18/07/2013
Background reading	01/07/2013	20	21/07/2013
Finding ontology editor	15/07/2013	7	22/07/2013
Preparing for progress meeting	28/07/2013	3	31/07/2013
Evaluation phase	10/08/2013	15	25/08/2013
Writing up the final report	16/08/2013	12	28/08/2013
Proofreading and reviewing the report	28/08/2013	10	07/09/2013

Gantt Chart:



Appendix E: Code of Convert Qurany HTML files to XML file (QuranytoXML.py)

```
# -*- coding: utf-8 -*-
import re
import glob
import xml.etree.ElementTree as ET
from xml.etree import ElementTree

#define global variables:
output = open('Qurany.xml','w')
#Specify the tree root
root = ET.Element('Quran')
def main():
    verse_INDX=0
    Qurany_files = glob.glob("qurany/*")
    for chapter_id in range(1,115):
            #Create Chapter node
            chapter = ET.SubElement(root,'chapter')
            chapter.set('id',str(chapter_id))
            #Files declarations
            for verse_id in range(1,288):
                filename = 'qurany/' + str(chapter_id)+'-'+str(verse_id)+'.html'
                if filename in Qurany_files:
                    #write the verse ID in Verse Node
                    verse_INDX+=1
                    verse = ET.SubElement(chapter,'verse')
                    verse.set('id', str(verse_INDX))
                    htmltxt = open(filename).readlines()
                    lenhtmltxt = len(htmltxt)


                    #setup Regular Expression patterns to clean the data
                    Tag_NewLine_markup ='[\t\n]' #removes taps and newline
markup
                    removechars = '[\d+/}/{\n\t]' #removes the verse character e.g.: 1}
                    removeHTLtags = '<.*?>' #removes HTML tags


                    #read the Qurany html file
                    for line in range(lenhtmltxt):
                        if re.search('<font size="5">',htmltxt[line]):
                            Arabic_verse = htmltxt[line+1].strip()
                            Arabic_verse =
re.sub(removechars,'',Arabic_verse)

                            #write the verse text in Arabic in Verse Node
                            verseText = ET.SubElement(verse,'text')
                            verseText.set('lang', 'Arabic' )
                            #convert Byte to Unicode using Decode function
                            verseText.text = Arabic_verse.decode('utf-8')



                        htmltxt[line] = htmltxt[line].replace('\n','')
                        htmltxt[line] = htmltxt[line].replace('\t','')

                        if re.search('<font color="#1645ae"
size="4">',htmltxt[line]):
                            htmltxt[line] = htmltxt[line].split('<font
color="#1645ae" size="4">')[1]
                            htmltxt[line] =
re.sub(removeHTLtags,'',htmltxt[line])
                            htmltxt[line]=htmltxt[line].strip()
                            Author = htmltxt[line]
                            verseText = ET.SubElement(verse,'text')
                            verseText.set('Author',Author)



                        if re.search(r'<td><Font size="4">',htmltxt[line]):
                            htmltxt[line] =
re.sub(removeHTLtags,'',htmltxt[line])
                            English_verse = htmltxt[line].strip()
                            English_verse = English_verse.replace("'",'')
                            English_verse = English_verse.replace('"','')
                            verseText.set('lang','English')
                            verseText.text = English_verse

                        if re.search('Concepts/Themes Covered:',htmltxt[line]):
                            index = line+1

                            for i in range(index,lenhtmltxt):
                                htmltxt[i] = htmltxt[i].replace('\n','')
                                htmltxt[i] = htmltxt[i].replace('\t','')
                                #Arabic Concepts List
                                if htmltxt[i]=='</br></br>' or
htmltxt[i]=='<font size="4">':
                            ArabicConcepts = htmltxt[i+1]
                            ArabicConcepts =
ArabicConcepts.replace('\n','')
```

```
                                    ArabicConcepts =
ArabicConcepts.replace('\t','')
                                    if ArabicConcepts !='</font>':
                                        # remove speces but
keep the concept's inside spaces.
                                        ArabicConcepts =
ArabicConcepts.strip()
                                        ArabicConcepts =
ArabicConcepts.split('>')
                                        B=""
                                        for c in
ArabicConcepts:
                                                B
+=c.strip() +';'
                                        #strip the last symbol
                                        B = B.strip(';')
                                        QuranyConcepts =
ET.SubElement(verse,'QuranyConcepts')

    QuranyConcepts.set('lang', 'Arabic')

    QuranyConcepts.set('tree',B.decode('utf-8'))


                                elif htmltxt[i] == '</br>':
                                    #English Concepts
                                        EnglishConcepts =
htmltxt[i+1]
                                        EnglishConcepts =
EnglishConcepts.replace('\n','')
                                        EnglishConcepts =
EnglishConcepts.replace('\t','')
                                        EnglishConcepts =
EnglishConcepts.split('>')
                                        B=""
                                        for c in
EnglishConcepts:
                                                B
+=c.strip(' ') +';'
                                        EQuranyConcepts =
B.strip(';')
                                        QuranyConcepts =
ET.SubElement(verse,'QuranyConcepts')

    QuranyConcepts.set('lang','English' )

    QuranyConcepts.set('tree',EQuranyConcepts)
```

```
#write the tree
        tree = ET.ElementTree(root)
        tree.write(output,encoding='utf-8')


if __name__ == '__main__':
        main()
```

Appendix F: Code of Converting Quranic Arabic Corpus TXT file to XML file (QACtoXML.py)

```
# -*- coding: utf-8 -*-
import re
import xml.etree.ElementTree as ET
from xml.etree import ElementTree

root = ET.Element('Quran')
xmlQAC = open('QAC.xml','w')


def main():

        segmentid=0
        prev_chpater_id=0
        prev_verse_id=0
        prev_seg_id=0
        verse_INDX=0



        QAC = open('quranic-corpus-morphology-0.4.txt').readlines()
        QACLinesLength = len(QAC)

        for i in range(QACLinesLength):
                if re.search("LOCATION\tFORM\tTAG\tFEATURES",QAC[i]):
                        index=i+1

        #chapter/verse/segment info
        for line in range(index,QACLinesLength):

                QACLine = QAC[line].split("\t")
                Location = QACLine[0].replace('\n','')
                Location = Location.strip(')')
                Location = Location.strip('(')

                form = QACLine[1]
                postag = QACLine[2]
                features = QACLine[3].replace('\n','')
                features = features.strip()
                chapterid,verseid,tokenid= Location_extractor(Location)
                segmentid+=1


                if      prev_chpater_id != chapterid:
                        chapter = ET.SubElement(root,'chapter')
                        chapter.set('id', str(chapterid))
                        prev_chpater_id = chapterid

                if prev_verse_id != verseid:
                        verse_INDX += 1
                        verse = ET.SubElement(chapter,'verse')
                        verse.set('id', str(verse_INDX))
                        prev_verse_id = verseid

                if prev_seg_id != segmentid:
                        seg = ET.SubElement(verse,'seg')
                        seg.set('id', str(segmentid))
                        prev_seg_id = segmentid
                seg.set('token_id', str(tokenid))
                seg.text=form
                seg.set('POSTag',postag)
                Features_extractor(features,seg)


        tree = ET.ElementTree(root)
        tree.write(xmlQAC)
        xmlQAC.close()


def Location_extractor(Location):
        Location = Location.split(":")
        chapterid = Location[0]
        verseid = Location[1]
        tokenid=Location[3]
        return chapterid,verseid,tokenid



def     Features_extractor(features,seg):
        features = features.split("|")
        for feature in features:
                feature = feature.strip()

                #if feature is like PREFIX|A:INTG+ or PREFIX|ka+
                if feature == 'PREFIX':
```

```
                PREFIX_Tager(features[0],features[1],seg)
                break

        #if feature has a colon : , such as  PRON:3MP or ROOT:qwl:
        elif ":" in feature:
                Twinsfeatures(feature.split(":")[0],feature.split(":")[1],seg)

        else:
                #if the feature has a single word such as 2MP, M ...
                Singlefeature(feature,seg)



def PREFIX_Tager(f1,f2,seg):
        #if the Prefix has this form PREFIX|A:INTG+
        if ":" in f2:
            seg.set('PREFIX',f2.split(":")[0])

            #put the Prefix in the Grammar node
            seg.set('Grammar',f1)

        else:
        #if the Prefix has this form PREFIX|ka+
            seg.set('Grammar', f1)
            seg.set('Prefix_features',f2)




def Twinsfeatures(f1,f2,seg):
    # 'POS' feature is deleted since it's repeated:
    if f1=='ROOT':
        seg.set('Root',f2)

    elif f1=='PRON':
        #NO need for this peace of info, output.write('\tgrammar2="'+ f1 + '"')
        Singlefeature(f2,seg)   #PRON has an additional features and need to be labeled
using Singlefeature

    elif f1=='SP':
        seg.set('special',f2)

    elif f1=='MOOD':
        seg.set('MOOD',f2)
    elif f1=='LEM':
        seg.set('Lemma', f2.encode('utf-8'))



def Singlefeature(f1,seg):

    if re.match(r'STEM|SUFFIX',f1):
        seg.set('Grammar', f1)


    elif
re.match(r'3MS|2MS|1MS|3MD|2MD|1MD|3MP|2MP|1MP|3FS|2FS|1FS|3FD|2FD|1FD|3FP|2FP
|1FP',f1):
            seg.set('Person', f1[0] + 'P')
            seg.set('Gender',f1[1])
            seg.set('Number', f1[2])

    elif  re.match(r'MS|MD|MP|FS|FD|FP',f1):

            seg.set('Gender', f1[0])
            seg.set('Number', f1[1])

    elif  re.match(r'1P|1D|1S|2P|2D|2S|3P|3D|3S',f1):
            seg.set('Person',f1[0] + 'P')
            seg.set('Number', f1[1])

    elif  re.match(r'/[123][MF]',f1):
            seg.set('Person', f1[0]+ 'P')
            seg.set('Gender', f1[1])


    elif  f1=='M' or f1=='F': #re.match(r'[MF]',f1):
            seg.set('Gender',f1)

    elif  f1=="P" or f1=="D" or f1=="S": #re.match(r'[PDS]',f1):
            seg.set('Number',f1)

    elif  f1=='PERF' or f1=='IMPF' or f1=='IMPV':
            seg.set('Aspect',f1)


    elif  f1=='ACT' or f1=='PASS':
            seg.set('Voice', f1)
    elif  re.match(r'\(([IVX]+\)',f1):
            f1=f1.strip(")")
            f1=f1.strip("(")
            seg.set('Verb_Form',f1)


    elif  f1=="PCPL" or f1=="VN": #re.match(r'PCPL|VN',f1):
            seg.set('Derivation',f1)


    elif  f1=='DEF' or f1=='INDEF':
            seg.set('State',f1)


    elif  f1=='NOM' or f1=='ACC' or f1=='GEN':
            seg.set('Case',f1)


    elif  f1 =='SUBJ' or f1 =='JUS' or f1 =='ENG' or f1 =='IND':
            seg.set('Mood',f1)




if __name__ == '__main__':
    main()
```

Appendix G: Code of Convert QurAna XML files to XML file (QurAnatoXML.py)

```
# -*- coding: utf-8 -*-
import xml.etree.ElementTree as ET
from xml.etree import ElementTree
import glob

conceptfile='Quran-pron/concepts.xml'
_concepts = ET.parse(conceptfile).getroot()
concept = open(conceptfile,'r')
xmlfile = open('QurAna.xml','w')
verse_INDX=0
prev_chapter_id=1
QurAna_files = glob.glob('Quran-pron/*')
root = ET.Element('Quran')

for i in range(1,115):

    file = 'Quran-pron/pronxml-'+str(i)+'.xml'
    if file in  QurAna_files:
        #chapter id:
        _chapter = ET.parse(file).getroot()
        chapterid = _chapter.get('id')
        chapter = ET.SubElement(root,'chapter')
        chapter.set('id',chapterid)

        for node in _chapter.iter():
            if node.tag=='verse':
                verse_INDX += 1
                verseid = node.get('id')
                verse = ET.SubElement(chapter,'verse')
                verse.set('id',str(verse_INDX))



            elif node.tag=='seg':
                segid=node.get('id')
                seg = ET.SubElement(verse,'seg')
                seg.set('id',segid)
                seg.text =  node.text

            if node.tag=='pron':
                for e in node.getchildren():
                    if node.get('con'):
                        for concept in _concepts.iter():
                            if concept.get('id') == '2054':
                                concept.set('id','1054')
```

```
                                                         if concept.get('id') == node.get('con'):
                                                                         for x in
concept.getchildren():
                                                                                 if x.tag=='arabic':
                                                                                         ArTXT =
x.text

        seg.set('Arconcept',ArTXT.strip())
                                                                                 if x.tag=='english':
                                                                                         EnTXT =
x.text

        seg.set('Enconcept',EnTXT.strip())
                                         if  node.get('id'):
                                                                 seg.set('PRON_id',node.get('id'))
                                         if  node.get('ant'):
                                                                 seg.set('ant',node.get('ant'))




tree= ET.ElementTree(root)
tree.write(xmlfile, encoding="utf-8")
xmlfile.close()
```

Appendix H: Code of Merging the three XML datasets in one XML file (merge.py)

```
 # -*- coding: utf-8 -*-
#import  The ElementTree XML API
import xml.etree.ElementTree as ET
from xml.etree import ElementTree
from xml.dom import minidom
import re



#Sketch engine POS filter format
def  SKEPosFilter(pos):

            if pos == 'N' or pos=='PN' or pos=='IMPN':
                  return "Noun|Noun::"+pos
            elif pos=='PRON' or pos=='DEM' or pos=='REL':
                  return "Pronouns|Pronouns::"+pos

            elif pos=='ADJ' or pos=='NUM':
                  return "Nominals|Nominals::"+pos

            elif pos=='T' or pos=='LOC':
                  return "Adverbs|Adverbs::"+pos

            elif pos=='V':
                  return "Verb|Verb::"+pos

            elif pos=='P':
                  return "Prepositions|Prepositions::"+pos

            elif pos=='EMPH' or pos=='IMPV' or pos=='PRP':
                  return "lam prefixes|lam prefixes::"+pos

            elif pos=='CONJ' or pos=='SUB':
                  return "Conjunctions|Conjunctions::"+pos

            elif pos=='ACC' or pos=='CIRC' or pos=='COM' or pos=='RSLT' or pos=='AMD'or
pos=='ANS' or pos=='AVR'or pos=='CAUS' or pos=='CERT' or pos=='COND' or pos=='EQ' or pos=='EXH'
or pos=='EXL' or pos=='EXP' or pos=='FUT' or pos=='INC' or pos=='INTG' or pos=='NEG' or
pos=='PREV' or pos=='PRO' or pos=='REM' or pos=='RES' or pos=='RET' or pos=='SUP' or pos=='SUR'
or pos=='VOC':
                  return "Particles|Particles::"+pos
```

```
        elif pos=='INL':
            return "Disconnected Letters|Disconnected Letters::"+pos
        else:
            return pos+'|'+pos+'::'+pos

#Sketch engine hierarchal elements format
def SketchEngineFormat(conceptline):
    conceptline=conceptline.strip()
    conceptline = conceptline.split(';')
    return '|'.join('::'.join(conceptline[:INDX]) for INDX in range(1,len(conceptline)+1)).strip()


#Function to remove XML Entity References
def removeXMLReference(filename):
    newfile = filename.split('.')[0] + '_v1.xml'
    output = open(newfile,'w')
    filename = open(filename).readlines()
    for line in filename:
        line = line.replace('&quot;','')
        line = line.replace('&amp;','')
        line = line.replace('&apos;','')
        line = line.replace('&lt;','')
        line = line.replace('&gt;','')
        output.write(line)
    output.close()

def SKEformat(file):

    root = ET.parse(file).getroot()
    for node in root.iter():
        if node.get('POSTag'):
            node.attrib['POSTag']=SKEPosFilter(node.get('POSTag'))
        if node.get('Enconcept'):
            node.attrib['Enconcept']='Prnoun Reference(English)|Prnoun
Reference(English)::'+node.get('Enconcept').strip()
        if node.get('Arconcept'):
            node.attrib['Arconcept']='Prnoun Reference(Arabic)|Prnoun
Reference(Arabic)::'+node.get('Arconcept').strip()
    tree = ET.ElementTree(root)
    SKE_filename='SKE_Unfied_Quranic_Corpus_Latin.xml'
    SKE_output = open(SKE_filename,'w')
    converttostring = ElementTree.tostring(root,encoding='utf-8')
    parsingTree = minidom.parseString(converttostring)
    convertedtree= parsingTree.toprettyxml(indent="",encoding='utf-8')
    SKE_output.write(convertedtree)
    SKE_output.close()
    removeXMLReference(SKE_filename)


#Defining the output files
```

```
filename='Unified_Quranic_Corpus.xml'
output = open(filename,'w')

SKE_filename='SKE_Unfied_Quranic_Corpus_VLatin.xml'
SKE_output = open(SKE_filename,'w')

#Parsing the three XML files
qac_tree= ET.parse("QAC.xml").getroot()
qurana_tree = ET.parse("QurAna.xml").getroot()
qurany_tree = ET.parse("Qurany.xml").getroot()

#initialise chapter,verse and segment counters to zero
Prev_chid=0
Prev_vid=0
segid=0

#Define the XML root Element in the two outputs
root = ET.Element('Quran')
SKEroot =  ET.Element('Quran')


#compare the chapters ids
for ch_qurana in qurana_tree:
    for ch_qac in qac_tree:
        for ch_qurany in qurany_tree:
                        if ch_qurana.get('id') == ch_qac.get('id') == ch_qurany.get('id'):
                                if Prev_chid!=ch_qurana.get('id'):
                                        chapter_MT = ET.SubElement(root,'chapter')
                                        chapter_MT.set('id',ch_qurana.get('id') )
                                        chapter_SKE = ET.SubElement(SKEroot,'chapter')
                                        chapter_SKE.set('id',ch_qurana.get('id') )
                                        Prev_chid=chapter_MT.get('id')

                                for verse_qurana in ch_qurana.getchildren():
                                        for verse_qac in ch_qac.getchildren():
                                                for verse_qurany in ch_qurany.getchildren():
                                                        if
verse_qurany.get('id')==verse_qurana.get('id') == verse_qac.get('id'):
                                                                verse_MT =
ET.SubElement(chapter_MT,'verse')

        verse_MT.set('id',verse_qurana.get('id') )
                                                                verse_SKE =
ET.SubElement(chapter_SKE,'verse')

        verse_SKE.set('id',verse_qurana.get('id') )

                                                                #Merge Qurany concepts
                  #retive Qurany verse English concepts using the XPath
                                                                if
verse_qurany.findall('.//QuranyConcepts/[@lang="English"]'):
        Econcepts= verse_qurany.findall('QuranyConcepts/[@lang="English"]')
                                                                                                    for
c in Econcepts:

        ENcurrent = ET.SubElement(verse_MT, 'QuranyConcepts')

        ENcurrent_SKE = ET.SubElement(verse_SKE, 'ENQuranyConcepts')

        ENcurrent_SKE.set('tree', SketchEngineFormat(c.get('tree')))


        for concept in c.get('tree').split(';'):

                concept=concept.strip()

                concept=re.sub('\W+','_',concept)

                if concept=='12SonsofJacob':

                        concept='twelveSonsofJacob'

                ENcurrent = ET.SubElement(ENcurrent,concept)




                                                                        #Add the Arabic Qurany Concepts
to Sketch engine corpus only.
                                                                        if
verse_qurany.findall('.//QuranyConcepts/[@lang="Arabic"]'):

        Econcepts= verse_qurany.findall('QuranyConcepts/[@lang="Arabic"]')
                                                                                                    for
c in Econcepts:

        ENcurrent_SKE = ET.SubElement(verse_SKE, 'ARQuranyConcepts')

        ENcurrent_SKE.set('tree', SketchEngineFormat(c.get('tree')))

                                                                        #Merge QAC with QurAna
                                                                        for seg_qac in
verse_qac.getchildren():


        #FIND POS TAG IF IT IS PRON TAG

        pos = seg_qac.get('POSTag')

        if pos=='PRON':
```

```
    for v in verse_qurana.getchildren():

        if v.get('PRON_id'):

            if v.get('Enconcept'):

                    seg_qac.set("Arconcept",v.get('Arconcept'))

                    seg_qac.set("Enconcept",v.get('Enconcept'))

                    del v.attrib['PRON_id']

                    break



seg_MT = ET.SubElement(verse_MT,'seg')

seg_MT.attrib = seg_qac.attrib


seg_SKE = ET.SubElement(verse_SKE,'seg')

seg_SKE.attrib = seg_qac.attrib



#Add Sketch engine Glue tag

if seg_SKE.get('token_id') > '1':

    glue = ET.SubElement(verse_SKE,'g')



text = str(seg_qac.text)

text = text.replace('\n','')



if  seg_SKE.get('Lemma'):
```

```
                Lemma = seg_SKE.get('Lemma')


        else:

                Lemma=text




        #Write the vertical line attributes

        seg_SKE.text = str(text.decode('utf-8')) +'\t'+ str(Lemma.decode('utf-8')) + '\t' + str(pos)

        seg_MT.text = text.decode('utf-8')


 #close the tree and write it in the output file
tree = ET.ElementTree(root)
#no need for pretty print
converttostring = ElementTree.tostring(root,encoding='utf-8')
parsingTree = minidom.parseString(converttostring)
convertedtree= parsingTree.toprettyxml(indent="",encoding='utf-8')
output.write(convertedtree)
output.close()


#sketch engine file

SKE_tree = ET.ElementTree(SKEroot)
SKE_tree.write(SKE_output)
SKE_output.close()
SKEformat(SKE_filename)
#remove XML entity reference characters
removeXMLReference(filename)
```

Appendix I: Code of adding Arabic and Latin text to the merged dataset (AddQuranicText.py)

```
import xml.etree.ElementTree as ET
from xml.etree import ElementTree
from xml.dom import minidom
import re



def GetText(input, output):
    XML_root= ET.parse("SKE_Unified_Quranic_Corpus_Latin_v1.xml").getroot()
    segments = XML_root.findall('.//seg')
    segINDX = len(segments)
    INDX =0
    for line in input:
        if  not re.match('<seg id',line) and not re.match('<chapter id',line) and not
re.match('<verse id',line) and not re.match('</',line) and not re.match('<g/>',line):
            if line != '\n' or line != '\t':
                #print line
                text = line.split('\t')[0].strip('\n')
                lemma = line.split('\t')[2].strip('\n')
                pos = line.split('\t')[1].strip('\n')
                if INDX >= segINDX:
                    break
                else:
                    segments[INDX].text = text.decode('utf-8') + '\t'+
lemma.decode('utf-8') + '\t'+ pos
                    INDX = INDX+1

    tree = ET.ElementTree(XML_root)
    #no need for pretty print
    converttostring = ElementTree.tostring(XML_root,encoding='utf-8')
    parsingTree = minidom.parseString(converttostring)
    convertedtree= parsingTree.toprettyxml(indent="",encoding='utf-8')
    output.write(convertedtree)
    #tree.write(output)
    output.close()
```

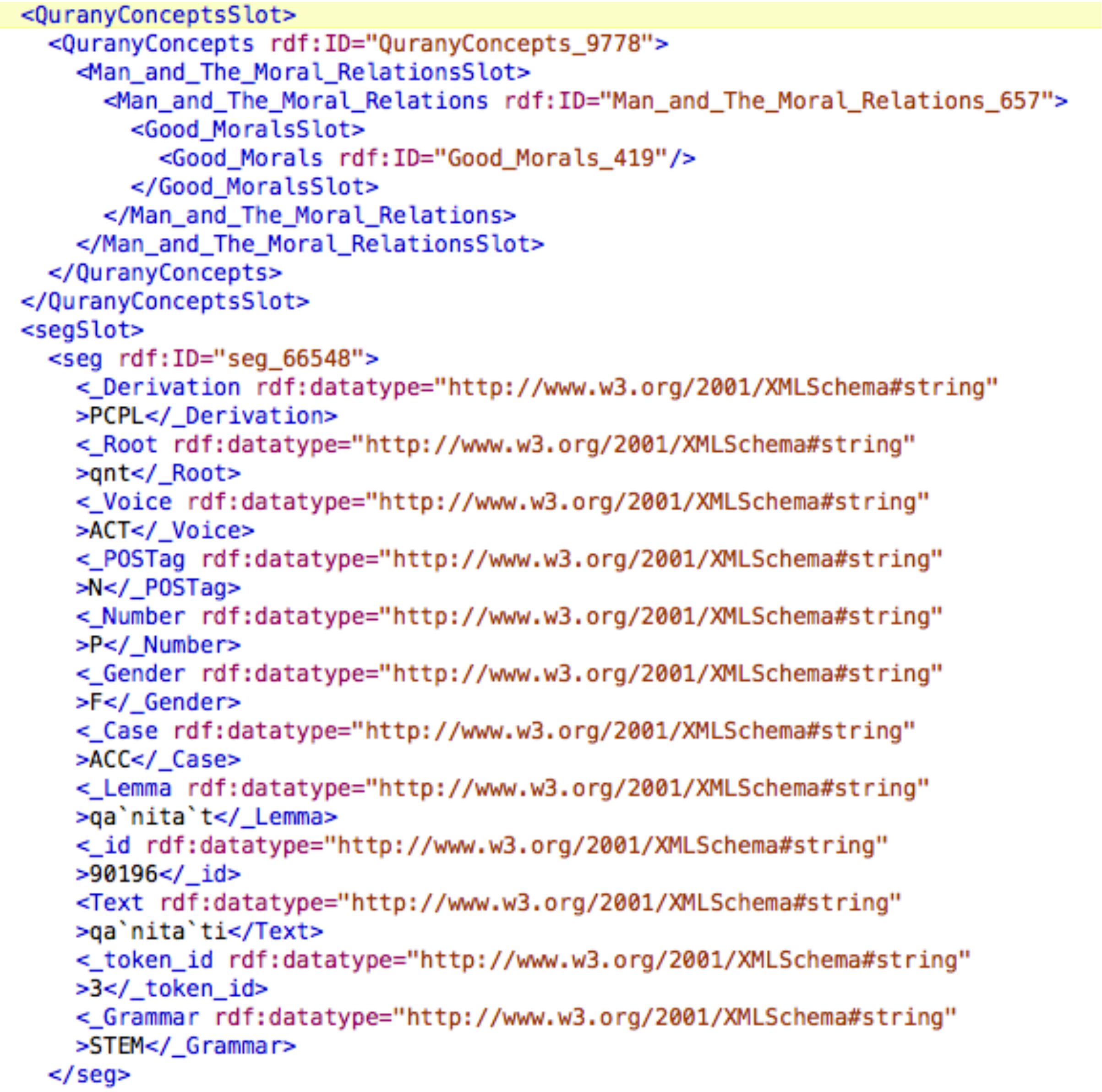
```
AUVSKE = open('UnifiedSKE_AU.xml','w')
AVSKE = open('UnifiedSKE_AV.xml','w')
LUSKE = open('UnifiedSKE_LU.xml','w')

AUV = open('IC/integrateAU.txt').readlines()
AV = open('IC/integrateAV.txt').readlines()
LUV = open('IC/integrateLU.txt').readlines()

GetText(AUV,AUVSKE)
GetText(AV,AVSKE)
GetText(LUV,LUSKE)


AUVSKE.close()
AVSKE.close()
LUSKE.close()
```

Appendix J: Sketch engine configuration file

```
NAME "Unified Quranic Annotations (Arabic Vowelled)"
PATH   "/corpora/ca/user_data/zainabAlquassem/manatee/ UnifiedSKE_AU/"
ENCODING     "UTF-8"
WPOSLIST       ",noun,N|PN, pronoun, PRON|DEM|REL, adverb,LOC|T, Derived nominals,ADJ|
IMPN, verb,V, Preposition,P, l?m Prefixes,EMPH| IMPV| PRP, Conjunction,CONJ|SUB,
Particle,ACC| AMD| ANS| AVR| CAUS| CERT| CIRC| COM| COND | EQ | EXH | EXL| EXP| FUT|
INC| INT| INTG| NE"
LPOSLIST        ",noun,N|PN, pronoun, PRON|DEM|REL, adverb,LOC|T, Derived nominals,ADJ|
IMPN, verb,V, Preposition,P, l?m Prefixes,EMPH| IMPV| PRP, Conjunction,CONJ|SUB,
Particle,ACC| AMD| ANS| AVR| CAUS| CERT| CIRC| COM| COND | EQ | EXH | EXL| EXP| FUT|
INC| INT| INTG| NE"
TAGSETDOC   "http://corpus.quran.com/documentation/tagset.jsp"
FILESTRUCTURE        ""
RIGHTTOLEFT           "1"
ALIGNED        ""
SUBCDEF        ""
SUBCBASE     "/corpora/ca/user_data/zainabAlquassem/manatee/ UnifiedSKE_AU/subcorp"
DOCSTRUCTURE        ""

ATTRIBUTE    "word"
ATTRIBUTE    "lemma"
ATTRIBUTE    "tag"
ATTRIBUTE    "lemma_lc" {
            DYNAMIC      "utf8lowercase"
            DYNLIB      "internal"
            ARG1        "C"
            FUNTYPE     "s"
            FROMATTR    "lemma"
            TYPE        "index"
            TRANSQUERY  "yes"
}
ATTRIBUTE    "lc" {
            DYNAMIC      "utf8lowercase"
            DYNLIB      "internal"
            ARG1        "C"
            FUNTYPE     "s"
            FROMATTR    "word"
            TYPE        "index"
            TRANSQUERY  "yes"
}

STRUCTURE   "chapter" {
ATTRIBUTE    "id"{
LABEL "Chapter ID"
}
}
STRUCTURE   "verse" {
ATTRIBUTE    "id"{
LABEL "Verse ID"
}
```

```
}
STRUCTURE   "seg"{
ATTRIBUTE "id"{
LABEL "Segment ID"
}
ATTRIBUTE "POSTAG"{
     MULTIVALUE  "1"
     MULTISEP "|"
     HIERARCHICAL   "::"
    LABEL "POS Tag"
}
ATTRIBUTE "State"{
LABEL "Segment State (INDEF=indefinite)"
}
ATTRIBUTE "Morpheme"{
LABEL "Morpheme"
}
ATTRIBUTE "Verb_Form"{
LABEL "Verb Form"
}
ATTRIBUTE "Derivation"{
LABEL "Derivation"
}
ATTRIBUTE "special"{
LABEL "Special Grammar (كان و إن و أخواتها)"
}
ATTRIBUTE "Case"{
LABEL "Case"
}
ATTRIBUTE "MOOD"{
LABEL "MOOD"
}
ATTRIBUTE "Number"{
LABEL "Number(S=Singular, D=Dual, P=Plural)"
}
ATTRIBUTE "Person"{
LABEL "Person"
}
ATTRIBUTE "Gender"{
LABEL "Gender (M=Masculine, F= feminine)"
}
ATTRIBUTE "Root"{
LABEL "Root"
}
ATTRIBUTE "PREFIX"{
LABEL "Prefix Letter"
}

ATTRIBUTE "Prefix_features"{
LABEL "Prefix Features"
}
ATTRIBUTE "ARconcept"{
LABEL "Arabic QurAna Concepts"
MULTIVALUE  "1"
     MULTISEP "|"
    HIERARCHICAL   "::"

}
ATTRIBUTE "ENconcept"{
 LABEL "English QurAna Concepts"
MULTIVALUE  "1"
    MULTISEP "|"
    HIERARCHICAL   "::"
}
}

STRUCTURE   "ENQuranyConcepts " {
     ATTRIBUTE       "tree"{
    LABEL "Qurany Concepts Tree"
    MULTIVALUE  "1"
    MULTISEP "|"
    HIERARCHICAL   "::"
}

}

STRUCTURE   "ARQuranyConcepts " {
     ATTRIBUTE       "tree"{
    LABEL "Qurany Concepts Tree"
    MULTIVALUE  "1"
    MULTISEP "|"
    HIERARCHICAL   "::"
}

}

STRUCTURE "QuranyConcepts"{
ATTRIBUTE "tree"{
LABEL "Qurany Concepts"
    MULTIVALUE  "1"
    MULTISEP "|"

}



}

STRUCTURE   "g" {
        DISPLAYTAG   0
        DISPLAYBEGIN "_EMPTY_"
}


WSDEF           "/corpora/ca/user_data/zainabAlquassem/sg/grammar_12.txt"
WSPOSLIST    ","
```

Appendix K: Part of Unified Quranic Ontology (UnifiedQuranicOntology.owl)



```
<QuranyConceptsSlot>
  <QuranyConcepts rdf:ID="QuranyConcepts_9778">
    <Man_and_The_Moral_RelationsSlot>
      <Man_and_The_Moral_Relations rdf:ID="Man_and_The_Moral_Relations_657">
        <Good_MoralsSlot>
          <Good_Morals rdf:ID="Good_Morals_419"/>
        </Good_MoralsSlot>
      </Man_and_The_Moral_Relations>
    </Man_and_The_Moral_RelationsSlot>
  </QuranyConcepts>
</QuranyConceptsSlot>
<segSlot>
  <seg rdf:ID="seg_66548">
    <_Derivation rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >PCPL</_Derivation>
    <_Root rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >qnt</_Root>
    <_Voice rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ACT</_Voice>
    <_POSTag rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >N</_POSTag>
    <_Number rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >P</_Number>
    <_Gender rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >F</_Gender>
    <_Case rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ACC</_Case>
    <_Lemma rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >qa`nita`t</_Lemma>
    <_id rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >90196</_id>
    <Text rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >qa`nita`ti</Text>
    <_token_id rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >3</_token_id>
    <_Grammar rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >STEM</_Grammar>
  </seg>
```

Appendix L: Abdullah Alfaifi evaluation

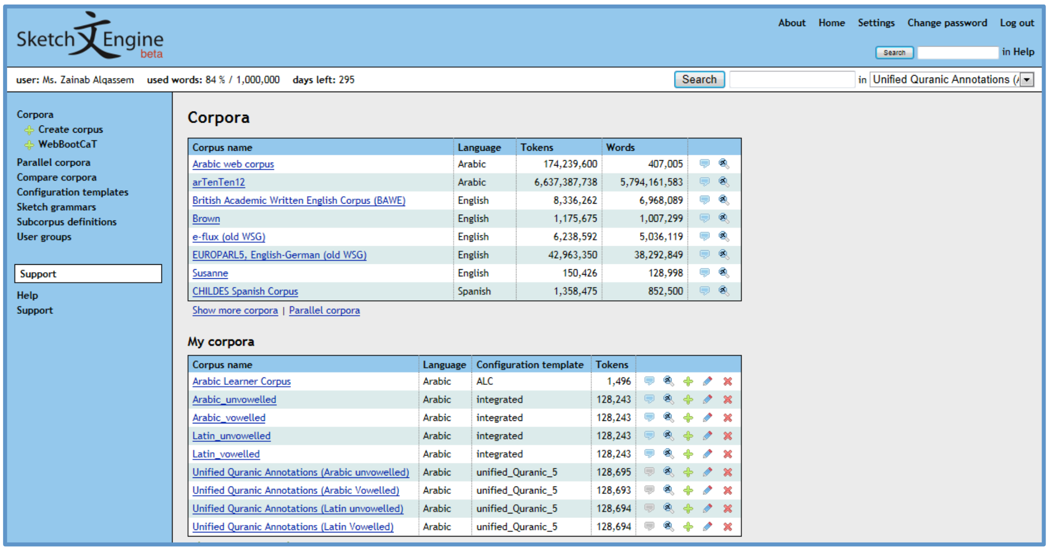
Evaluation of Luluh Aldhubayi’s Project

Abdullah Alfaifi

Introduction

This is a brief evaluation of the **Unified Quranic Annotations and Ontologies** project of **Luluh Aldhubayi**. I didn’t have full information about the project except a PowerPoint presentation file (.pptx) which explains (1) how to login to the project account on the SketchEngine website, (2) some features that can be used for searching the corpus (PoS, Syntactic, and Semantic features), but it can be seen that the project idea is to provide users with a unified method of annotation which can be used for morphological, syntactic, and semantic search and analysis in the Quranic text.

As a specialist in Arabic, my evaluation will focus on searching the Arabic text, unvowelled in particular (**Error! Reference source not found.**), which may clearly show how we can apply the corpus features to our search and analysis.



user: Ms. Zainab Alqassem used words: 84 % / 1,000,000 days left: 295

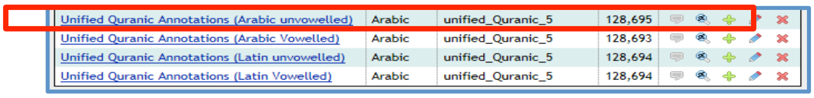
Corpora

Corpus name	Language	Tokens	Words
Arabic web corpus	Arabic	174,239,600	407,005
arTenTen12	Arabic	6,637,387,738	5,794,161,583
British Academic Written English Corpus (BAWE)	English	8,336,262	6,968,089
Brown	English	1,175,675	1,007,299
e-flux (old WSG)	English	6,238,592	5,036,119
EUROPARL5, English-German (old WSG)	English	42,963,350	38,292,849
Susanne	English	150,426	128,998
CHILDES Spanish Corpus	Spanish	1,358,475	852,500

My corpora

Corpus name	Language	Configuration template	Tokens
Arabic Learner Corpus	Arabic	ALC	1,496
Arabic_unvowelled	Arabic	integrated	128,243
Arabic_vowelled	Arabic	integrated	128,243
Latin_unvowelled	Arabic	integrated	128,243
Latin_vowelled	Arabic	integrated	128,243
Unified Quranic Annotations (Arabic unvowelled)	Arabic	unified_Quranic_5	128,695
Unified Quranic Annotations (Arabic Vowelled)	Arabic	unified_Quranic_5	128,693
Unified Quranic Annotations (Latin unvowelled)	Arabic	unified_Quranic_5	128,694
Unified Quranic Annotations (Latin Vowelled)	Arabic	unified_Quranic_5	128,694

Figure 1: The main screen of the project on Sketch Engine

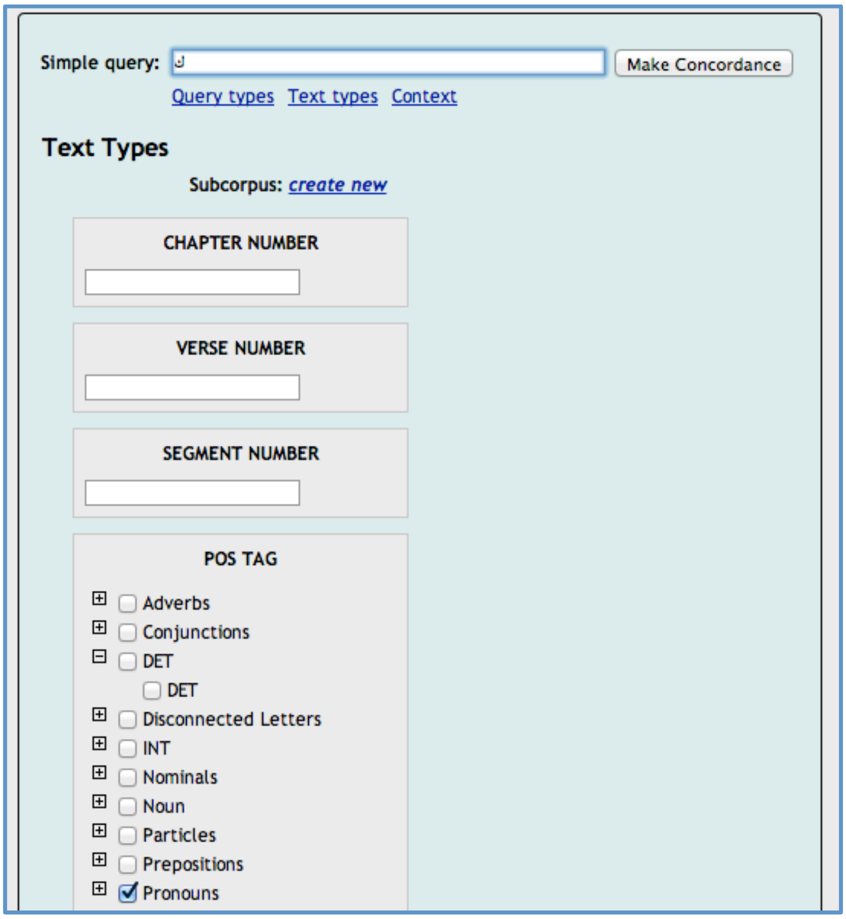


Unified Quranic Annotations (Arabic unvowelled)	Arabic	unified_Quranic_5	128,695
Unified Quranic Annotations (Arabic Vowelled)	Arabic	unified_Quranic_5	128,693
Unified Quranic Annotations (Latin unvowelled)	Arabic	unified_Quranic_5	128,694
Unified Quranic Annotations (Latin Vowelled)	Arabic	unified_Quranic_5	128,694

Figure 2: The “Arabic_unvowelled” corpus was selected for the search

Evaluation

The evaluation started by searching for the pronoun "ك". I typed "ك" in the *Simple query* text-field, and selected *Pronoun* from the list *POS TAG* to generate concordances that include this specific pronoun (Figure 3). 1,166 instances were shown (Figure 4).



Simple query: ك Make Concordance

Query types Text types Context

Text Types

Subcorpus: *create new*

CHAPTER NUMBER

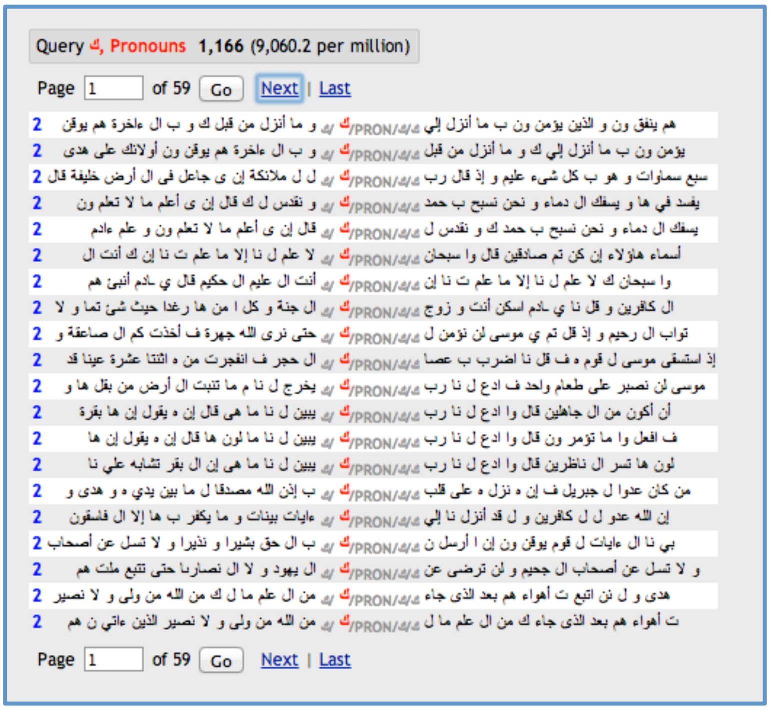
VERSE NUMBER

SEGMENT NUMBER

POS TAG

⊞ □ Adverbs
⊞ □ Conjunctions
⊟ □ DET
□ DET
⊞ □ Disconnected Letters
⊞ □ INT
⊞ □ Nominals
⊞ □ Noun
⊞ □ Particles
⊞ □ Prepositions
⊞ ☑ Pronouns

Figure 3: Searching for the pronoun "ك"



Query ك, Pronouns 1,166 (9,060.2 per million)

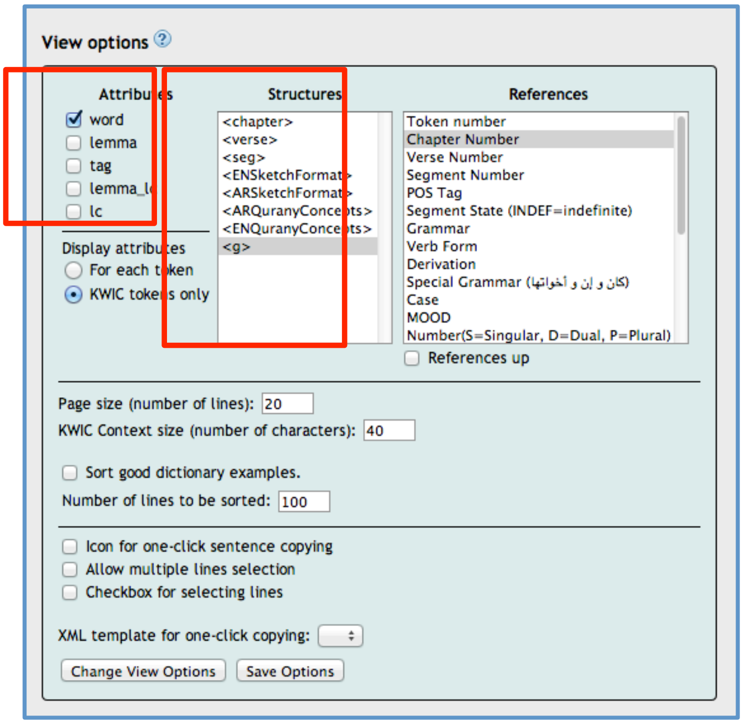
Page 1 of 59 Go Next | Last

```
2  هم ينفق ون و الذين يؤمن ون ب ما أنزل إلي ك/PRON/ك/ك و ما أنزل من قبل ك و ب ال ءاخرة هم يوقن
2  يؤمن ون ب ما أنزل إلي ك و ما أنزل من قبل ك/PRON/ك/ك و ب ال ءاخرة هم يوقن ون أولائك على هدى
2  سبع سماوات و هو ب كل شيء عليم و إذ قال رب ك/PRON/ك/ك ل ل ملائكة إن ى جاعل فى ال أرض خليفة قال
2  يفسد في ها و يسفك ال دماء و نحن نسبح ب حمد ك/PRON/ك/ك و نقدس ل ك قال إن ى أعلم ما لا تعلم ون
2  يسفك ال دماء و نحن نسبح ب حمد ك و نقدس ل ك/PRON/ك/ك قال إن ى أعلم ما لا تعلم ون و علم ءادم
2  أسماء هاؤلاء إن كن تم صادقين قال وا سبحان ك/PRON/ك/ك لا علم ل نا إلا ما علم ت نا إن ك أنت ال
2  وا سبحان ك لا علم ل نا إلا ما علم ت نا إن ك/PRON/ك/ك أنت ال عليم ال حكيم قال ي ـادم أنبئ هم
2  ال كافرين و قل نا ي ـادم اسكن أنت و زوج ك/PRON/ك/ك ال جنة و كل ا من ها رغدا حيث شئ تما و لا
2  تواب ال رحيم و إذ قل تم ي موسى لن نؤمن ل ك/PRON/ك/ك حتى نرى الله جهرة ف أخذت كم ال صاعقة و
2  إذ استسقى موسى ل قوم ه ف قل نا اضرب ب عصا ك/PRON/ك/ك ال حجر ف انفجرت من ه اثنتا عشرة عينا قد
2  موسى لن نصبر على طعام واحد ف ادع ل نا رب ك/PRON/ك/ك يخرج ل نا م ما تنبت ال أرض من بقل ها و
2  أن أكون من ال جاهلين قال وا ادع ل نا رب ك/PRON/ك/ك يبين ل نا ما هى قال إن ه يقول إن ها بقرة
2  ف افعل وا ما تؤمر ون قال وا ادع ل نا رب ك/PRON/ك/ك يبين ل نا ما لون ها قال إن ه يقول إن ها
2  لون ها تسر ال ناظرين قال وا ادع ل نا رب ك/PRON/ك/ك يبين ل نا ما هى إن ال بقر تشابه علي نا
2  من كان عدوا ل جبريل ف إن ه نزل ه على قلب ك/PRON/ك/ك ب إذن الله مصدقا ل ما بين يدي ه و هدى و
2  إن الله عدو ل ل كافرين و ل قد أنزل نا إلي ك/PRON/ك/ك ءايات بينات و ما يكفر ب ها إلا ال فاسقون
2  بي نا ال ءايات ل قوم يوقن ون إن ا أرسل ن ك/PRON/ك/ك ب ال حق بشيرا و نذيرا و لا تسل عن أصحاب
2  و لا تسل عن أصحاب ال جحيم و لن ترضى عن ك/PRON/ك/ك ال يهود و لا ال نصارىا حتى تتبع ملت هم
2  هدى و ل ئن اتبع ت أهواء هم بعد الذى جاء ك/PRON/ك/ك من ال علم ما ل ك من الله من ولى و لا نصير
2  ت أهواء هم بعد الذى جاء ك من ال علم ما ل ك/PRON/ك/ك من الله من ولى و لا نصير الذين ءاتي ن هم
```

Page 1 of 59 Go Next | Last

Figure 4: Results of searching for "ك"

I then clicked on the ***View options*** on the left hand side of the screen, and in order to hide the metadata I unselected all attributes check boxes except *word*, then selected the tag <g> from *Structures* list to remove spaces between segmented morphemes and shows Arabic words in a correct form (e.g. "ينفق ون" to be "ينفقون"), then clicked on the "**Change View Option**" button (Figure 5), this showed the concordances of all instances of the pronoun "ك" with no metadata (Figure 6).



View options

Attributes
☑ word
□ lemma
□ tag
□ lemma_lc
□ lc

Display attributes
○ For each token
● KWIC tokens only

Structures
<chapter>
<verse>
<seg>
<ENSketchFormat>
<ARSketchFormat>
<ARQuranyConcepts>
<ENQuranyConcepts>
<g>

References
Token number
Chapter Number
Verse Number
Segment Number
POS Tag
Segment State (INDEF=indefinite)
Grammar
Verb Form
Derivation
Special Grammar (كان و إن و أخواتها)
Case
MOOD
Number(S=Singular, D=Dual, P=Plural)

□ References up

Page size (number of lines): 20
KWIC Context size (number of characters): 40

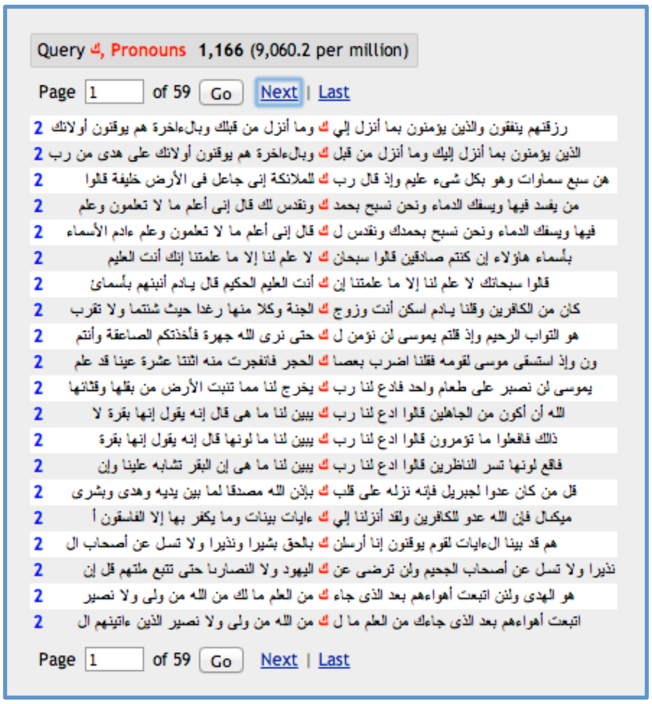
□ Sort good dictionary examples.
Number of lines to be sorted: 100

□ Icon for one-click sentence copying
□ Allow multiple lines selection
□ Checkbox for selecting lines

XML template for one-click copying:

Change View Options | Save Options

Figure 5: Changing the view options to hide the metadata



Query ك, Pronouns 1,166 (9,060.2 per million)

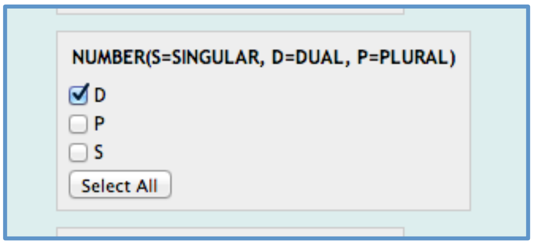
Page 1 of 59 Go Next | Last

2 رزقنهم ينفقون والذين يؤمنون بما أنزل إلي ك وما أنزل من قبلك وبالءاخرة هم يوقنون أولائك
2 الذين يؤمنون بما أنزل إليك وما أنزل من قبل ك وبالءاخرة هم يوقنون أولائك على هدى من رب
2 هن سبع سماوات وهو بكل شيء عليم وإذ قال رب ك للملائكة إني جاعل فى الأرض خليفة قالوا
2 من يفسد فيها ويسفك الدماء ونحن نسبح بحمد ك ونقدس لك قال إنى أعلم ما لا تعلمون وعلم
2 فيها ويسفك الدماء ونحن نسبح بحمدك ونقدس ل ك قال إنى أعلم ما لا تعلمون وعلم ءادم الأسماء
2 بأسماء هاؤلاء إن كنتم صادقين قالوا سبحان ك لا علم لنا إلا ما علمتنا إنك أنت العليم
2 قالوا سبحانك لا علم لنا إلا ما علمتنا إن ك أنت العليم الحكيم قال يـادم أنبئهم بأسمائ
2 كان من الكافرين وقلنا يـادم اسكن أنت وزوج ك الجنة وكلا منها رغدا حيث شئتما ولا تقرب
2 هو التواب الرحيم وإذ قلتم يموسى لن نؤمن ل ك حتى نرى الله جهرة فأخذتكم الصاعقة وأنتم
2 ون وإذ استسقى موسى لقومه فقلنا اضرب بعصا ك الحجر فانفجرت منه اثنتا عشرة عينا قد علم
2 يموسى لن نصبر على طعام واحد فادع لنا رب ك يخرج لنا مما تنبت الأرض من بقلها وقثائها
2 الله أن أكون من الجاهلين قالوا ادع لنا رب ك يبين لنا ما هى قال إنه يقول إنها بقرة لا
2 ذالك فافعلوا ما تؤمرون قالوا ادع لنا رب ك يبين لنا ما لونها قال إنه يقول إنها بقرة
2 فاقع لونها تسر الناظرين قالوا ادع لنا رب ك يبين لنا ما هى إن البقر تشابه علينا وإن
2 قل من كان عدوا لجبريل فإنه نزله على قلب ك بإذن الله مصدقا لما بين يديه وهدى وبشرى
2 ميكـال فإن الله عدو للكافرين ولقد أنزلنا إلي ك ءايات بينات وما يكفر بها إلا الفاسقون أ
2 هم قد بينا الءايات لقوم يوقنون إنا أرسلن ك بالحق بشيرا ونذيرا ولا تسل عن أصحاب ال
2 نذيرا ولا تسل عن أصحاب الجحيم ولن ترضى عن ك اليهود ولا النصـارىا حتى تتبع ملتهم قل إن
2 هو الهدى ولئن اتبعت أهواءهم بعد الذى جاء ك من العلم ما لك من الله من ولى ولا نصير
2 اتبعت أهواءهم بعد الذى جاءك من العلم ما ل ك من الله من ولى ولا نصير الذين ءاتينهم ال

Page 1 of 59 Go Next | Last

Figure 6: Results of the pronoun "ك" after hiding the metadata

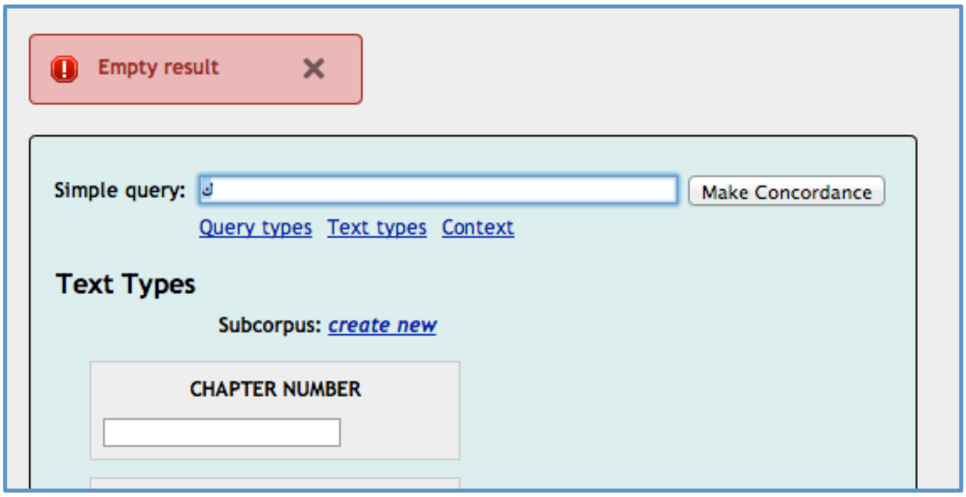
In order to have concordances of the dual form of the same pronoun "ك", the feature *D* was selected from the list *NUMBER(S=SINGULAR, D=DUAL, P=PLURAL)* (Figure 7). However, a message of “Empty result” was shown (Figure 8).



NUMBER(S=SINGULAR, D=DUAL, P=PLURAL)

☑ D
□ P
□ S

Figure 7: How to search using CQL in Sketch Engine



Empty result ✕

Simple query: ك

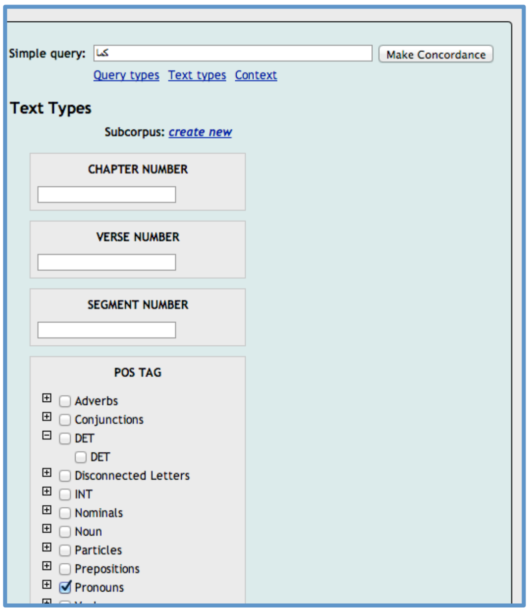
Text Types

Subcorpus: *create new*

CHAPTER NUMBER

Figure 8: Empty result when selecting the dual feature with the pronoun "ك"

The solution was to enter the dual form "كما" (Figure 9) with unselecting the feature *D*, this gave the correct result (Figure 10).


Simple query: کما Make Concordance

Query types Text types Context

Text Types

Subcorpus: *create new*

CHAPTER NUMBER

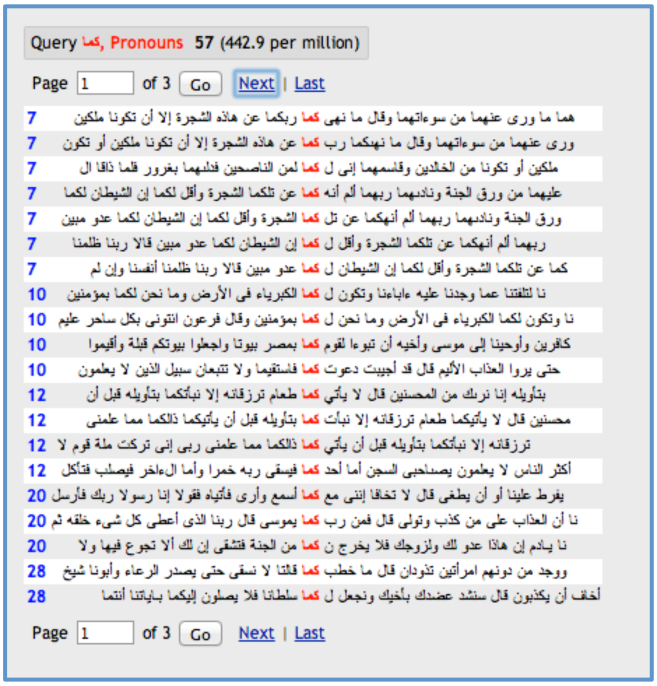
VERSE NUMBER

SEGMENT NUMBER

POS TAG

- ⊞ □ Adverbs
- ⊞ □ Conjunctions
- ⊟ □ DET
  - □ DET
- ⊞ □ Disconnected Letters
- ⊞ □ INT
- ⊞ □ Nominals
- ⊞ □ Noun
- ⊞ □ Particles
- ⊞ □ Prepositions
- ⊞ ☑ Pronouns

Figure 9: Searching for the dual form of the pronoun "ك"


Query كما, Pronouns 57 (442.9 per million)

Page 1 of 3 Go Next | Last

7	هما ما ورى عنهما من سوءاتهما وقال ما نهى كما ربكما عن هاذه الشجرة إلا أن تكونا ملكين
7	ورى عنهما من سوءاتهما وقال ما نهيكما رب كما عن هاذه الشجرة إلا أن تكونا ملكين أو تكون
7	ملكين أو تكونا من الخالدين وقاسمهما إنى ل كما لمن الناصحين فدلىهما بغرور فلما ذاقا ال
7	عليهما من ورق الجنة ونادىهما ربهما ألم أنه كما عن تلكما الشجرة وأقل لكما إن الشيطان لكما
7	ورق الجنة ونادىهما ربهما ألم أنهكما عن تل كما الشجرة وأقل لكما إن الشيطان لكما عدو مبين
7	ربهما ألم أنهكما عن تلكما الشجرة وأقل ل كما إن الشيطان لكما عدو مبين قالا ربنا ظلمنا
7	كما عن تلكما الشجرة وأقل لكما إن الشيطان ل كما عدو مبين قالا ربنا ظلمنا أنفسنا وإن لم
10	نا لتلفتنا عما وجدنا عليه ءاباءنا وتكون ل كما الكبرياء فى الأرض وما نحن لكما بمؤمنين
10	نا وتكون لكما الكبرياء فى الأرض وما نحن ل كما بمؤمنين وقال فرعون انتونى بكل ساحر عليم
10	كافرين وأوحينا إلى موسى وأخيه أن تبوءا لقوم كما بمصر بيوتا واجعلوا بيوتكم قبلة وأقيموا
10	حتى يروا العذاب الأليم قال قد أجيبت دعوت كما فاستقيما ولا تتبعان سبيل الذين لا يعلمون
12	بتأويله إنا نريك من المحسنين قال لا يأتي كما طعام ترزقانه إلا نبأتكما بتأويله قبل أن
12	محسنين قال لا يأتيكما طعام ترزقانه إلا نبأت كما بتأويله قبل أن يأتيكما ذالكما مما علمنى
12	ترزقانه إلا نبأتكما بتأويله قبل أن يأتي كما ذالكما مما علمنى ربى إنى تركت ملة قوم لا
12	أكثر الناس لا يعلمون يصاحبى السجن أما أحد كما فيسقى ربه خمرا وأما الءاخر فيصلب فتأكل
20	يفرط علينا أو أن يطغى قال لا تخافا إننى مع كما أسمع وأرى فأتياه فقولا إنا رسولا ربك فأرسل
20	نا أن العذاب على من كذب وتولى قال فمن رب كما يموسى قال ربنا الذى أعطى كل شىء خلقه ثم
20	نا يـادم إن هاذا عدو لك ولزوجك فلا يخرج ن كما من الجنة فتشقى إن لك ألا تجوع فيها ولا
28	ووجد من دونهم امرأتين تذودان قال ما خطب كما قالتا لا نسقى حتى يصدر الرعاء وأبونا شيخ
28	أخاف أن يكذبون قال سنشد عضدك بأخيك ونجعل ل كما سلطانا فلا يصلون إليكما بـاياتنا أنتما

Page 1 of 3 Go Next | Last

Figure 10: The result appears when using the form "كما"

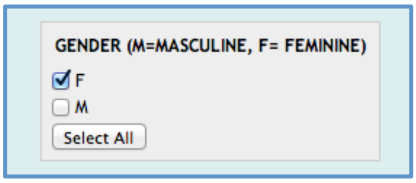
I noticed that using the feature *D* from the list *NUMBER(S=SINGULAR, D=DUAL, P=PLURAL)*, would be useless if I still need to change the form itself. In other words, I can search for the form "كما" directly with no need to select the dual feature check box, as I already did.

It is known in traditional Arabic linguistics that some pronouns have one lemma for their forms. For instance, the pronoun [Hā’ ’lġā’ib] "هاء الغائب" has five different forms:

1. "ه" (3rd person, **Masculine**, *Singular*)
2. "ها" (3rd person, **Feminine**, *Singular*)
3. "هما" (3rd person, **Masculine** or **Feminine**, *Dual*)
4. "هم" (3rd person, **Masculine**, *Plural*)
5. "هن" (3rd person, **Feminine**, *Plural*)

They share the same lemma "ه" which is the same form as the 3_M_S. The additional characters then (e.g. "ا", "ما", "م", "ن") indicate the features of gender and number, so some linguists consider the pronoun "هما", for example, as two morphemes ("ه" and "ما"). This is also applied to the pronoun "ك" (with a sole different in the 3_F_S form which has a short vowel (ـِ) instead of the long vowel "ا").

On the other hand, if I search for two pronouns that have the same form, in such case features work perfectly. For instance, searching for "كَ" (3_M_S) and "كِ" (3_F_S) can be done in the unvowelled corpus using the gender feature (Figure 11) which shows good results (Figure 12 and Figure 13).

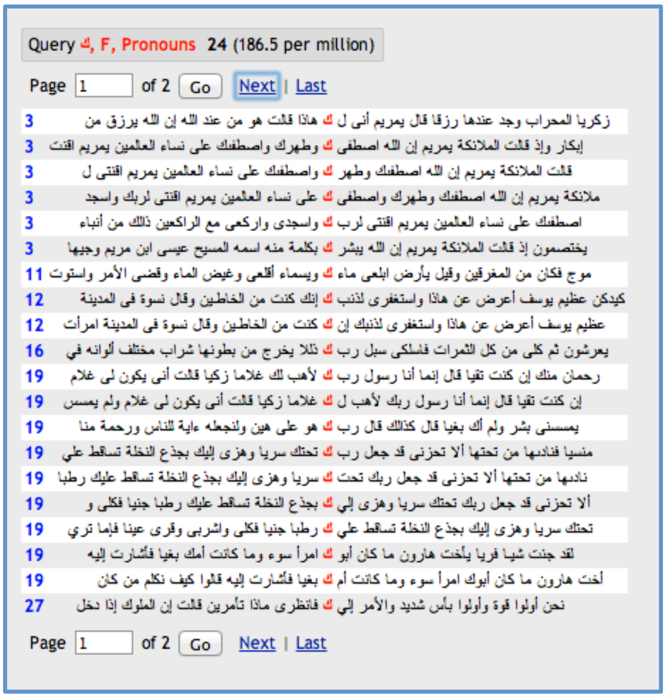


GENDER (M=MASCULINE, F= FEMININE)

☑ F

□ M

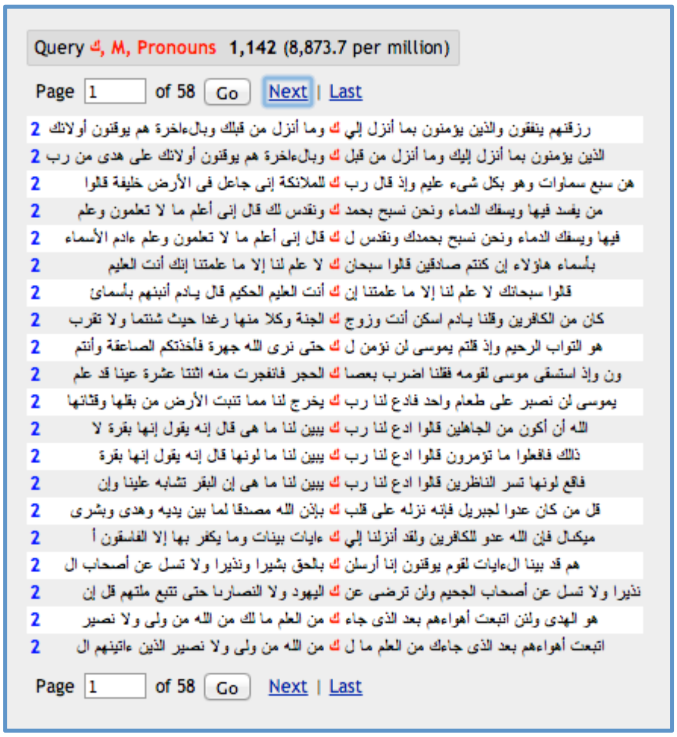
Figure 11: Using the gender feature to distinguish between different pronouns



Query ك, F, Pronouns 24 (186.5 per million)

3 زكريا المحراب وجد عندها رزقا قال يمريم أنى ل ك هاذا قالت هو من عند الله إن الله يرزق من
3 إبكار وإذ قالت الملائكة يمريم إن الله اصطفى ك وطهرك واصطفك على نساء العلمين يمريم اقنت
3 قالت الملائكة يمريم إن الله اصطفك وطهر ك واصطفك على نساء العلمين يمريم اقنتى ل
3 ملائكة يمريم إن الله اصطفك وطهرك واصطفى ك على نساء العلمين يمريم اقنتى لربك واسجد
3 اصطفك على نساء العلمين يمريم اقنتى لرب ك واسجدى واركعى مع الراكعين ذالك من أنباء
3 يختصمون إذ قالت الملائكة يمريم إن الله يبشر ك بكلمة منه اسمه المسيح عيسى ابن مريم وجيها
11 موج فكان من المغرقين وقيل يأرض ابلعى ماء ك ويسماء أقلعى وغيض الماء وقضى الأمر واستوت
12 كيدكن عظيم يوسف أعرض عن هاذا واستغفرى لذنب ك إنك كنت من الخاطين وقال نسوة فى المدينة
12 عظيم يوسف أعرض عن هاذا واستغفرى لذنبك إن ك كنت من الخاطين وقال نسوة فى المدينة امرأت
16 يعرشون ثم كلى من كل الثمرات فاسلكى سبل رب ك ذللا يخرج من بطونها شراب مختلف ألوانه في
19 رحمان منك إن كنت تقيا قال إنما أنا رسول رب ك لأهب لك غلاما زكيا قالت أنى يكون لى غلام
19 إن كنت تقيا قال إنما أنا رسول ربك لأهب ل ك غلاما زكيا قالت أنى يكون لى غلام ولم يمسس
19 يمسسنى بشر ولم أك بغيا قال كذالك قال رب ك هو على هين ولنجعله ءاية للناس ورحمة منا
19 منسيا فنادىها من تحتها ألا تحزنى قد جعل رب ك تحتك سريا وهزى إليك بجذع النخلة تساقط علي
19 نادىها من تحتها ألا تحزنى قد جعل ربك تحت ك سريا وهزى إليك بجذع النخلة تساقط عليك رطبا
19 ألا تحزنى قد جعل ربك تحتك سريا وهزى إلي ك بجذع النخلة تساقط عليك رطبا جنيا فكلى و
19 تحتك سريا وهزى إليك بجذع النخلة تساقط علي ك رطبا جنيا فكلى واشربى وقرى عينا فإما ترى
19 لقد جئت شيا فريا يأخت هارون ما كان أبو ك امرأ سوء وما كانت أمك بغيا فأشارت إليه
19 أخت هارون ما كان أبوك امرأ سوء وما كانت أم ك بغيا فأشارت إليه قالوا كيف نكلم من كان
27 نحن أولوا قوة وأولوا بأس شديد والأمر إلي ك فانظرى ماذا تأمرين قالت إن الملوك إذا دخل

Figure 12: Result of searching for the feminine form of the pronoun "ك"



Query ك, M, Pronouns 1,142 (8,873.7 per million)

Page 1 of 58 Go Next | Last

رزقنهم ينفقون والذين يؤمنون بما أنزل إلي ك وما أنزل من قبلك وبالءاخرة هم يوقنون أولائك 2
الذين يؤمنون بما أنزل إليك وما أنزل من قبل ك وبالءاخرة هم يوقنون أولائك على هدى من رب 2
هن سبع سماوات وهو بكل شيء عليم وإذ قال رب ك للملائكة إنى جاعل فى الأرض خليفة قالوا 2
من يفسد فيها ويسفك الدماء ونحن نسبح بحمد ك ونقدس لك قال إنى أعلم ما لا تعلمون وعلم 2
فيها ويسفك الدماء ونحن نسبح بحمدك ونقدس ل ك قال إنى أعلم ما لا تعلمون وعلم ءادم الأسماء 2
بأسماء هاؤلاء إن كنتم صادقين قالوا سبحان ك لا علم لنا إلا ما علمتنا إنك أنت العليم 2
قالوا سبحانك لا علم لنا إلا ما علمتنا إن ك أنت العليم الحكيم قال يـادم أنبئهم بأسمائ 2
كان من الكافرين وقلنا يـادم اسكن أنت وزوج ك الجنة وكلا منها رغدا حيث شئتما ولا تقرب 2
هو التواب الرحيم وإذ قلتم يموسى لن نؤمن ل ك حتى نرى الله جهرة فأخذتكم الصاعقة وأنتم 2
ون وإذ استسقى موسى لقومه فقلنا اضرب بعصا ك الحجر فانفجرت منه اثنتا عشرة عينا قد علم 2
يموسى لن نصبر على طعام واحد فادع لنا رب ك يخرج لنا مما تنبت الأرض من بقلها وقثائها 2
الله أن أكون من الجاهلين قالوا ادع لنا رب ك يبين لنا ما هى قال إنه يقول إنها بقرة لا 2
ذالك فافعلوا ما تؤمرون قالوا ادع لنا رب ك يبين لنا ما لونها قال إنه يقول إنها بقرة 2
فاقع لونها تسر الناظرين قالوا ادع لنا رب ك يبين لنا ما هى إن البقر تشابه علينا وإن 2
قل من كان عدوا لجبريل فإنه نزله على قلب ك بإذن الله مصدقا لما بين يديه وهدى وبشرى 2
ميكال فإن الله عدو للكافرين ولقد أنزلنا إلي ك ءايات بينات وما يكفر بها إلا الفاسقون أ 2
هم قد بينا الءايات لقوم يوقنون إنا أرسلن ك بالحق بشيرا ونذيرا ولا تسل عن أصحاب ال 2
نذيرا ولا تسل عن أصحاب الجحيم ولن ترضى عن ك اليهود ولا النصارىا حتى تتبع ملتهم قل إن 2
هو الهدى ولئن اتبعت أهواءهم بعد الذى جاء ك من العلم ما لك من الله من ولى ولا نصير 2
اتبعت أهواءهم بعد الذى جاءك من العلم ما ل ك من الله من ولى ولا نصير الذين ءاتينهم ال 2

Page 1 of 58 Go Next | Last

Figure 13: Result of searching for the masculine form of the pronoun "ك"

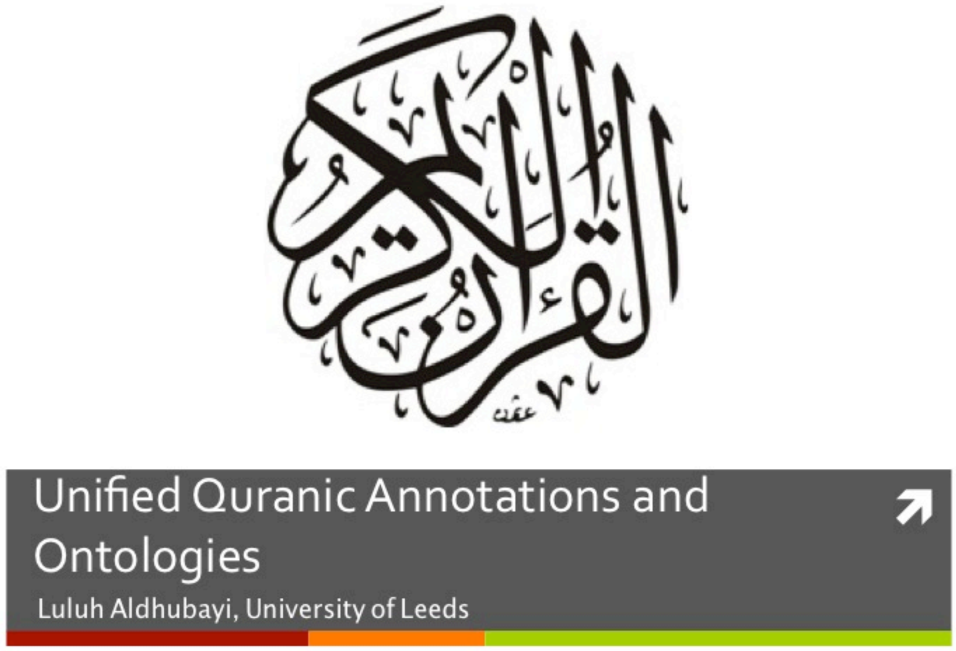
My suggestion for improving the use of the features included is to add more attention to the method used for extracting lemmas, which is linguistically important in some cases of corpus analysis.

Conclusion

It can be concluded from this brief evaluation that the **Unified Quranic Annotations and Ontologies** corpus seems to have well-structured design which provide useful features (PoS, Syntactic, and Semantic) that enable users to search in the Quranic text for different purposes, Quranic and linguistic studies.

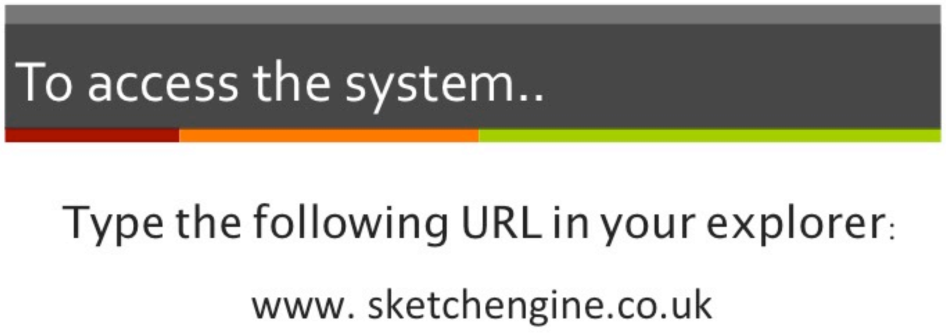
I believe that a corpus with such number of features can be beneficial for many researchers in different domains, so I recommend Luluh to develop a brief guideline that explains in details how to benefit from each feature with examples illustrating to the potential users the advantages of this corpus. Another recommendation is to publish a short research paper –collaborating with some academic members– in order to introduce this corpus to a broader audience of researchers.

Appendix M: Sketch engine system manual provided to evaluators:



Unified Quranic Annotations and Ontologies

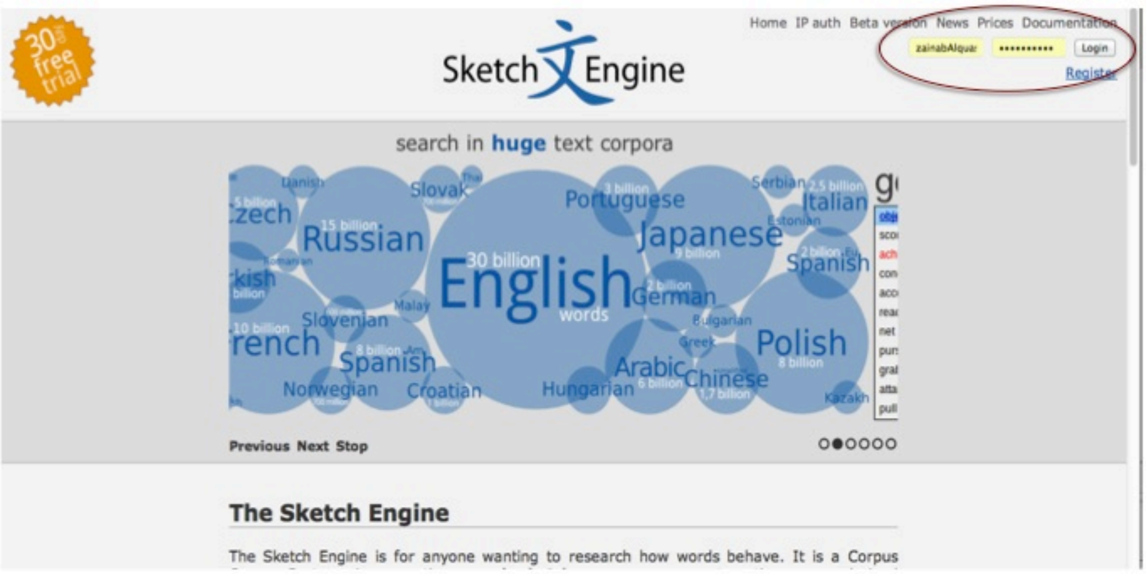
Luluh Aldhubayi, University of Leeds



To access the system..

Type the following URL in your explorer:

www. sketchengine.co.uk



The Sketch Engine

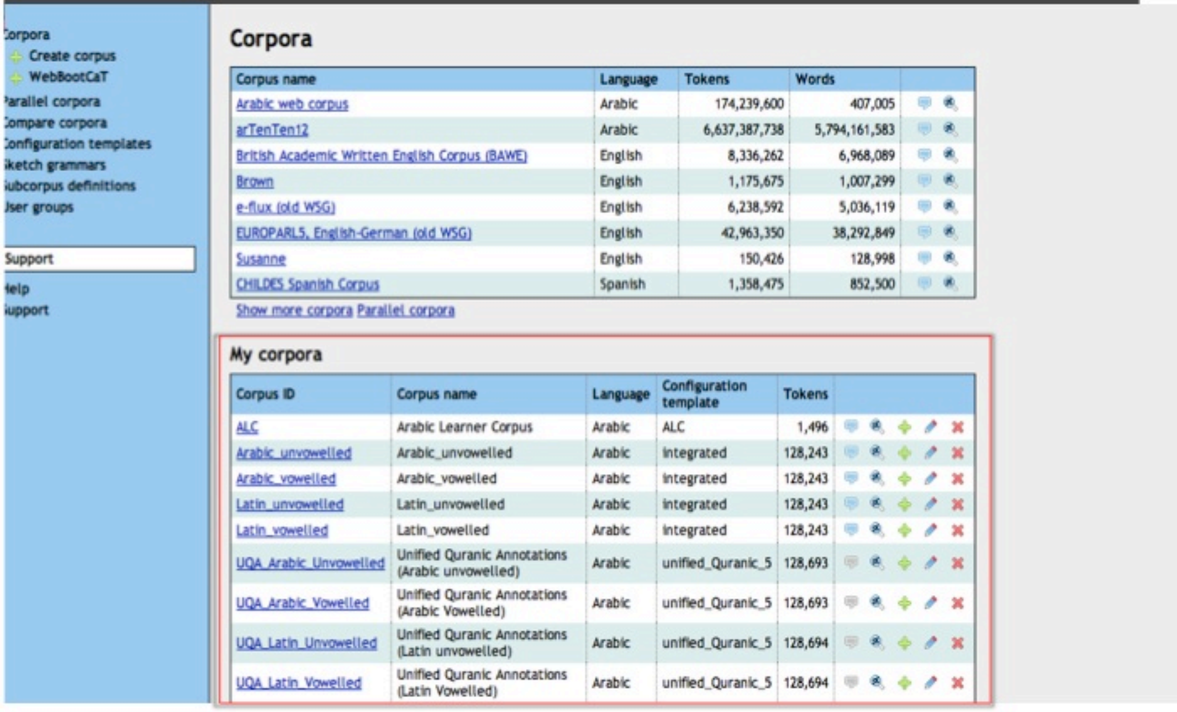
The Sketch Engine is for anyone wanting to research how words behave. It is a Corpus

Use the following inforamtion to login:

User Name : zainabAlquassem

Password: HMuP4xTAqE

My corpora



orpora
Create corpus
WebBootCaT
arallel corpora
ompare corpora
onfiguration templates
ketch grammars
ubcorpus definitions
ser groups

Support

elp
upport

Corpora

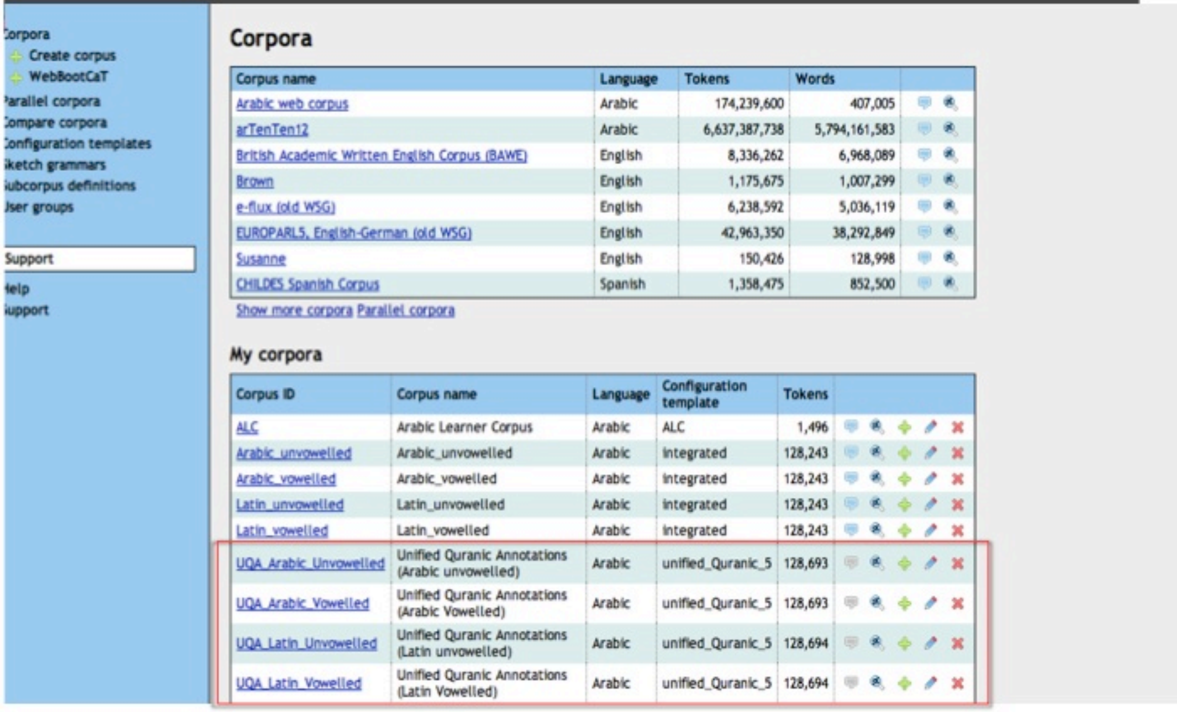
Corpus name	Language	Tokens	Words
Arabic web corpus	Arabic	174,239,600	407,005
arTenTen12	Arabic	6,637,387,738	5,794,161,583
British Academic Written English Corpus (BAWE)	English	8,336,262	6,968,089
Brown	English	1,175,675	1,007,299
e-flux (old WSG)	English	6,238,592	5,036,119
EUROPARL5, English-German (old WSG)	English	42,963,350	38,292,849
Susanne	English	150,426	128,998
CHILDES Spanish Corpus	Spanish	1,358,475	852,500

Show more corpora Parallel corpora

My corpora

Corpus ID	Corpus name	Language	Configuration template	Tokens
ALC	Arabic Learner Corpus	Arabic	ALC	1,496
Arabic_unvowelled	Arabic_unvowelled	Arabic	integrated	128,243
Arabic_vowelled	Arabic_vowelled	Arabic	integrated	128,243
Latin_unvowelled	Latin_unvowelled	Arabic	integrated	128,243
Latin_vowelled	Latin_vowelled	Arabic	integrated	128,243
UQA_Arabic_Unvowelled	Unified Quranic Annotations (Arabic unvowelled)	Arabic	unified_Quranic_5	128,693
UQA_Arabic_Vowelled	Unified Quranic Annotations (Arabic Vowelled)	Arabic	unified_Quranic_5	128,693
UQA_Latin_Unvowelled	Unified Quranic Annotations (Latin unvowelled)	Arabic	unified_Quranic_5	128,694
UQA_Latin_Vowelled	Unified Quranic Annotations (Latin Vowelled)	Arabic	unified_Quranic_5	128,694

Choose one of the last four corpora:



orpora
Create corpus
WebBootCaT
arallel corpora
ompare corpora
onfiguration templates
ketch grammars
ubcorpus definitions
ser groups

Support

elp
upport

Corpora

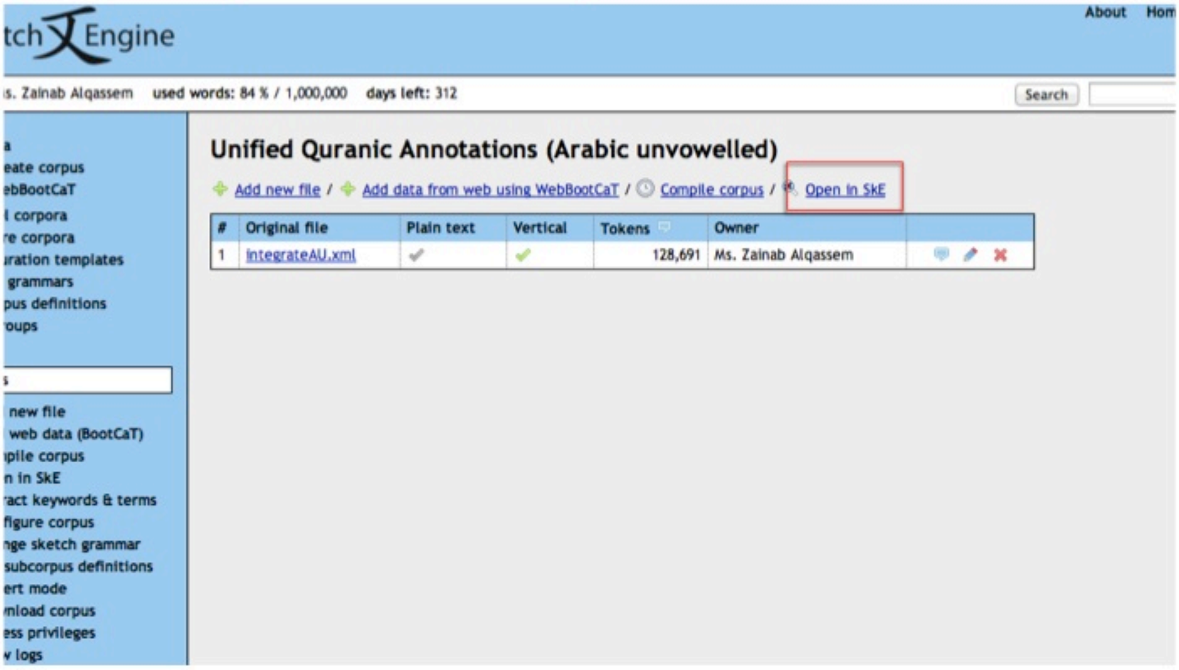
Corpus name	Language	Tokens	Words
Arabic web corpus	Arabic	174,239,600	407,005
arTenTen12	Arabic	6,637,387,738	5,794,161,583
British Academic Written English Corpus (BAWE)	English	8,336,262	6,968,089
Brown	English	1,175,675	1,007,299
e-flux (old WSG)	English	6,238,592	5,036,119
EUROPARL5, English-German (old WSG)	English	42,963,350	38,292,849
Susanne	English	150,426	128,998
CHILDES Spanish Corpus	Spanish	1,358,475	852,500

Show more corpora Parallel corpora

My corpora

Corpus ID	Corpus name	Language	Configuration template	Tokens
ALC	Arabic Learner Corpus	Arabic	ALC	1,496
Arabic_unvowelled	Arabic_unvowelled	Arabic	integrated	128,243
Arabic_vowelled	Arabic_vowelled	Arabic	integrated	128,243
Latin_unvowelled	Latin_unvowelled	Arabic	integrated	128,243
Latin_vowelled	Latin_vowelled	Arabic	integrated	128,243
UQA_Arabic_Unvowelled	Unified Quranic Annotations (Arabic unvowelled)	Arabic	unified_Quranic_5	128,693
UQA_Arabic_Vowelled	Unified Quranic Annotations (Arabic Vowelled)	Arabic	unified_Quranic_5	128,693
UQA_Latin_Unvowelled	Unified Quranic Annotations (Latin unvowelled)	Arabic	unified_Quranic_5	128,694
UQA_Latin_Vowelled	Unified Quranic Annotations (Latin Vowelled)	Arabic	unified_Quranic_5	128,694

Open in SKE

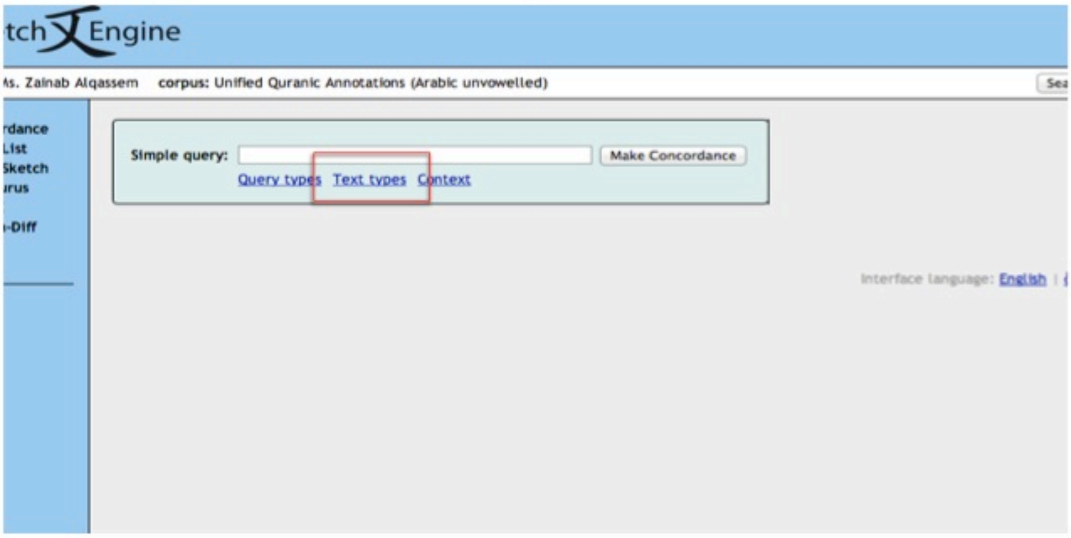


Unified Quranic Annotations (Arabic unvowelled)

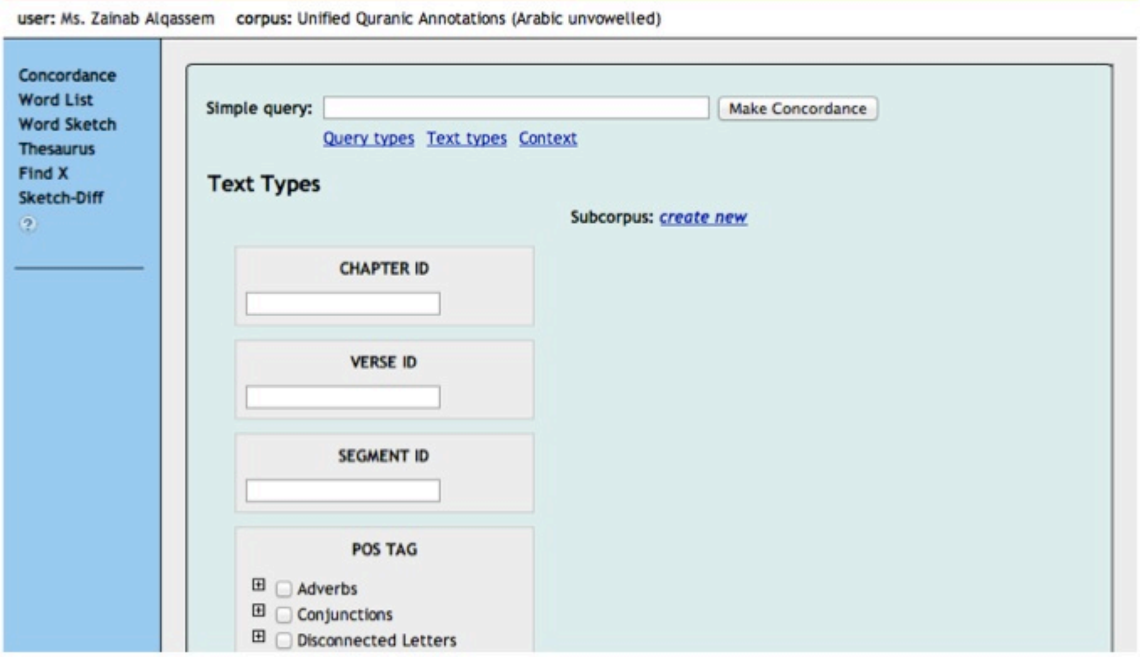
Add new file / Add data from web using WebBootCaT / Compile corpus / Open in SkE

#	Original file	Plain text	Vertical	Tokens	Owner	
1	integrateAU.xml	✓	✓	128,691	Ms. Zainab Alqassem	

Text types



Scroll down to get more options



user: Ms. Zainab Alqassem corpus: Unified Quranic Annotations (Arabic unvowelled)

Text Types

Subcorpus: *create new*

CHAPTER ID

VERSE ID

SEGMENT ID

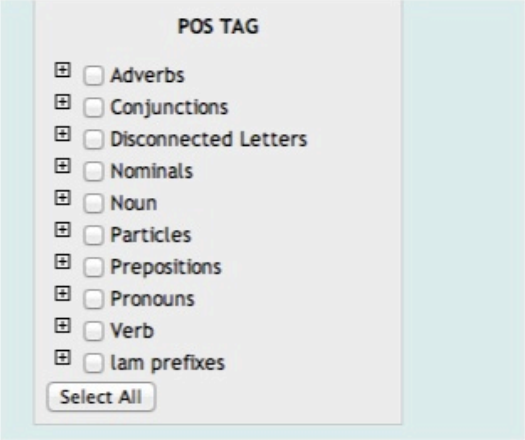
POS TAG

- ⊞ ☐ Adverbs
- ⊞ ☐ Conjunctions
- ⊞ ☐ Disconnected Letters

From the previous screen you can make the following search levels:

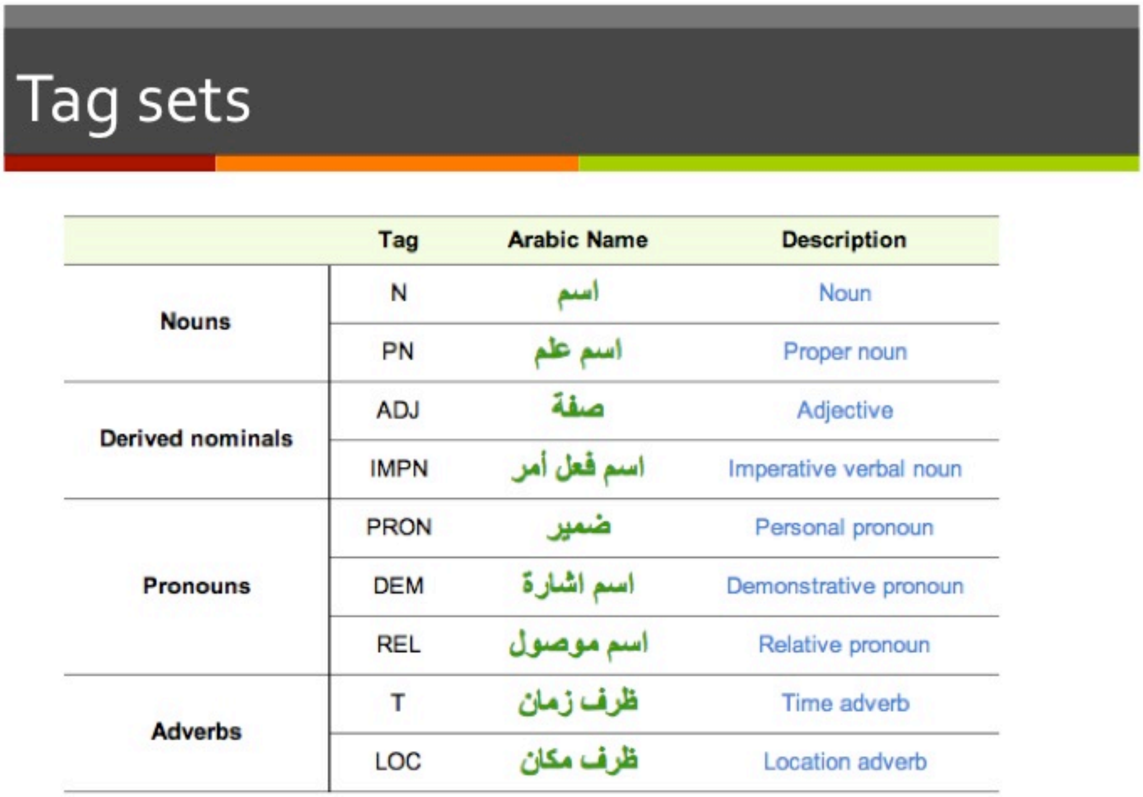
Fisrt:

Part of speech analysis in Quran



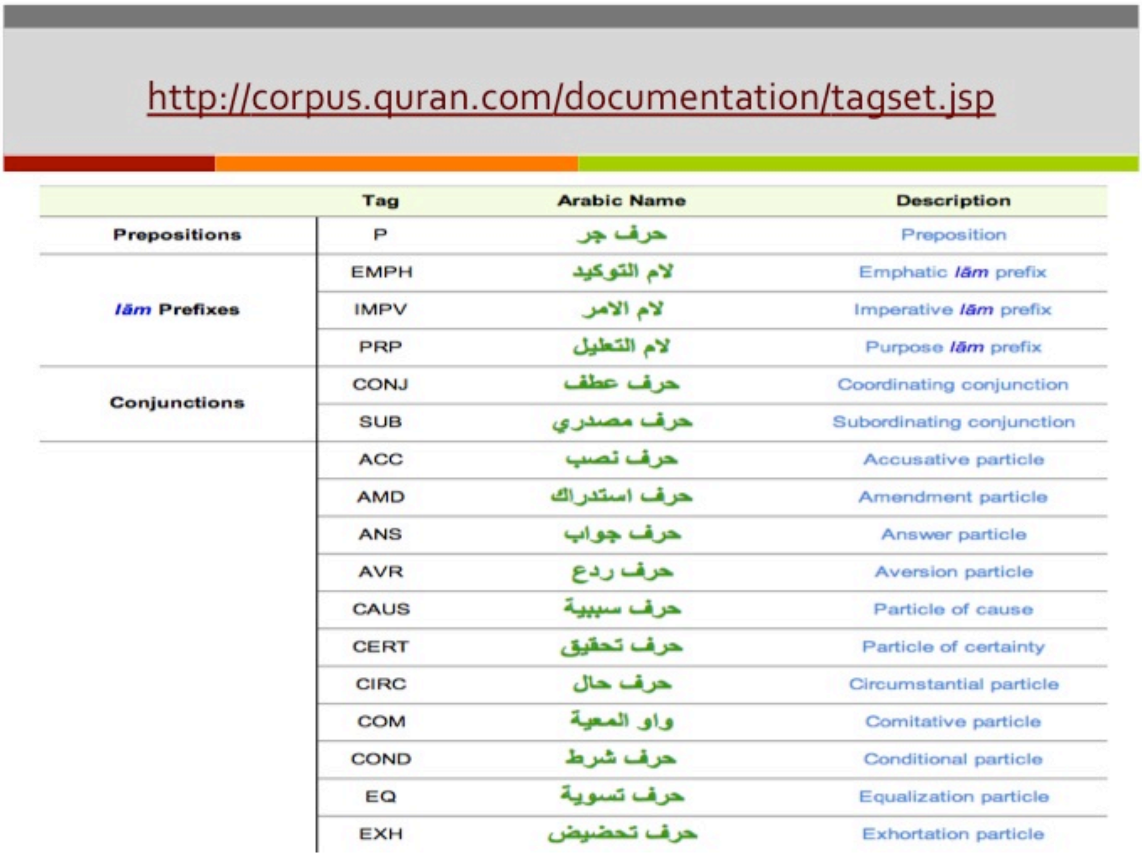
POS TAG

- ⊞ ☐ Adverbs
- ⊞ ☐ Conjunctions
- ⊞ ☐ Disconnected Letters
- ⊞ ☐ Nominals
- ⊞ ☐ Noun
- ⊞ ☐ Particles
- ⊞ ☐ Prepositions
- ⊞ ☐ Pronouns
- ⊞ ☐ Verb
- ⊞ ☐ lam prefixes



Tag sets

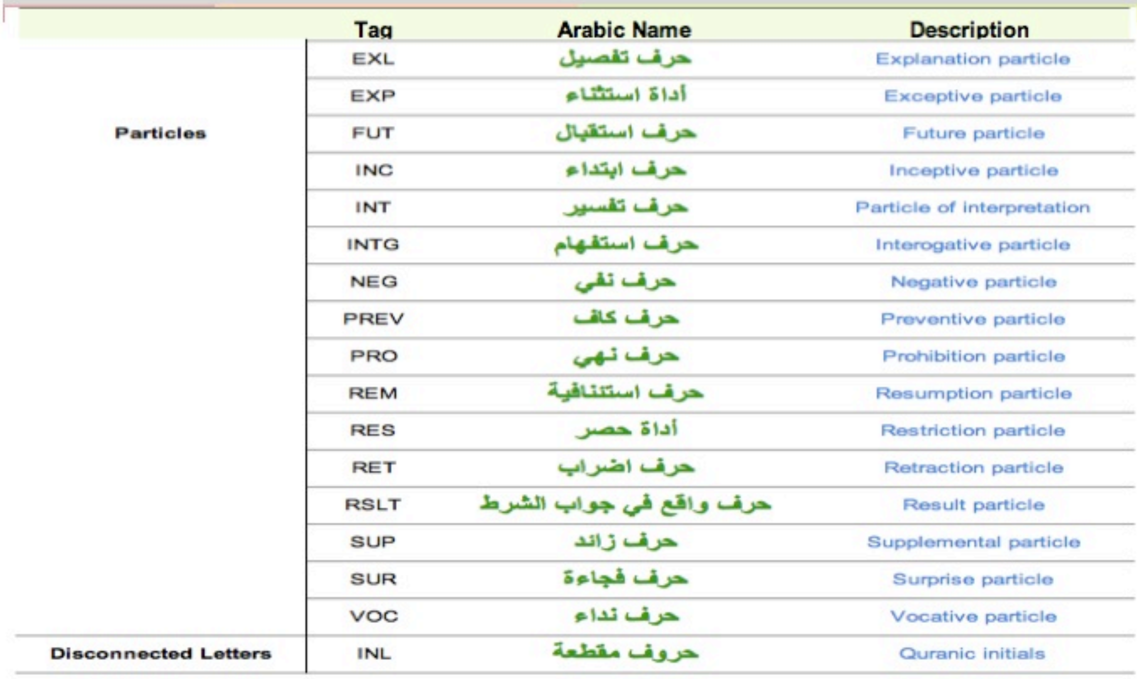
	Tag	Arabic Name	Description
Nouns	N	اسم	Noun
	PN	اسم علم	Proper noun
Derived nominals	ADJ	صفة	Adjective
	IMPN	اسم فعل أمر	Imperative verbal noun
Pronouns	PRON	ضمير	Personal pronoun
	DEM	اسم اشارة	Demonstrative pronoun
	REL	اسم موصول	Relative pronoun
Adverbs	T	ظرف زمان	Time adverb
	LOC	ظرف مكان	Location adverb



http://corpus.quran.com/documentation/tagset.jsp

	Tag	Arabic Name	Description
Prepositions	P	حرف جر	Preposition
***lām* Prefixes**	EMPH	لام التوكيد	Emphatic *lām* prefix
	IMPV	لام الامر	Imperative *lām* prefix
	PRP	لام التعليل	Purpose *lām* prefix
Conjunctions	CONJ	حرف عطف	Coordinating conjunction
	SUB	حرف مصدري	Subordinating conjunction
	ACC	حرف نصب	Accusative particle
	AMD	حرف استدراك	Amendment particle
	ANS	حرف جواب	Answer particle
	AVR	حرف ردع	Aversion particle
	CAUS	حرف سببية	Particle of cause
	CERT	حرف تحقيق	Particle of certainty
	CIRC	حرف حال	Circumstantial particle
	COM	واو المعية	Comitative particle
	COND	حرف شرط	Conditional particle
	EQ	حرف تسوية	Equalization particle
	EXH	حرف تحضيض	Exhortation particle

http://corpus.quran.com/documentation/tagset.jsp



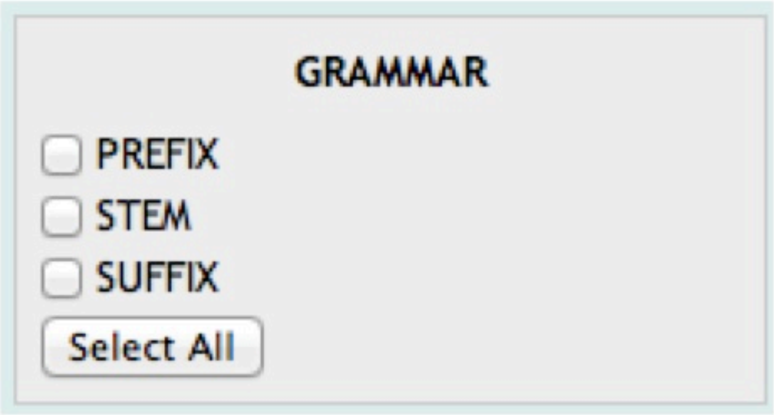
	Tag	Arabic Name	Description
Particles	EXL	حرف تفصيل	Explanation particle
	EXP	أداة استثناء	Exceptive particle
	FUT	حرف استقبال	Future particle
	INC	حرف ابتداء	Inceptive particle
	INT	حرف تفسير	Particle of interpretation
	INTG	حرف استفهام	Interogative particle
	NEG	حرف نفي	Negative particle
	PREV	حرف كاف	Preventive particle
	PRO	حرف نهي	Prohibition particle
	REM	حرف استئنافية	Resumption particle
	RES	أداة حصر	Restriction particle
	RET	حرف اضراب	Retraction particle
	RSLT	حرف واقع في جواب الشرط	Result particle
	SUP	حرف زائد	Supplemental particle
	SUR	حرف فجاءة	Surprise particle
	VOC	حرف نداء	Vocative particle
Disconnected Letters	INL	حروف مقطعة	Quranic initials

Fig 3. Part-of-speech tagset for particles and the Quranic initials.

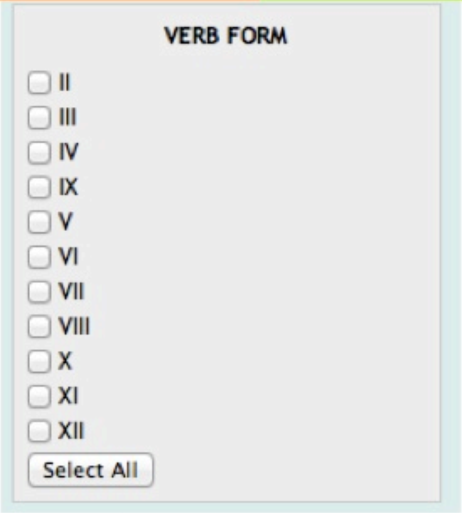
Second:
Syntactic analysis for each word in Quran

Which includes the following options:

Prefix, Stem or Suffix



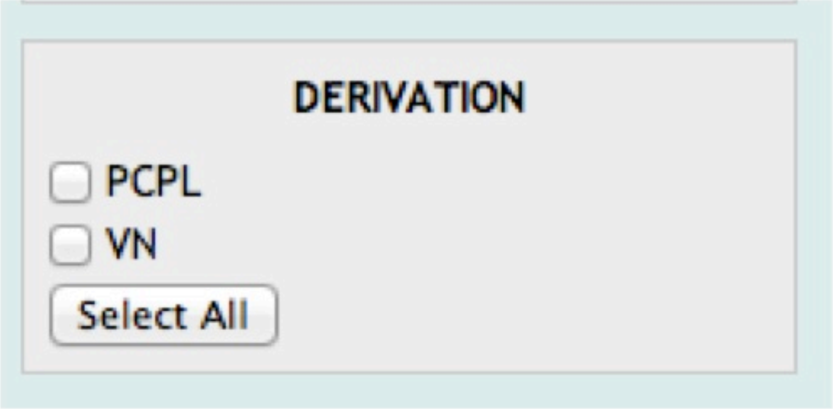
Verb form:



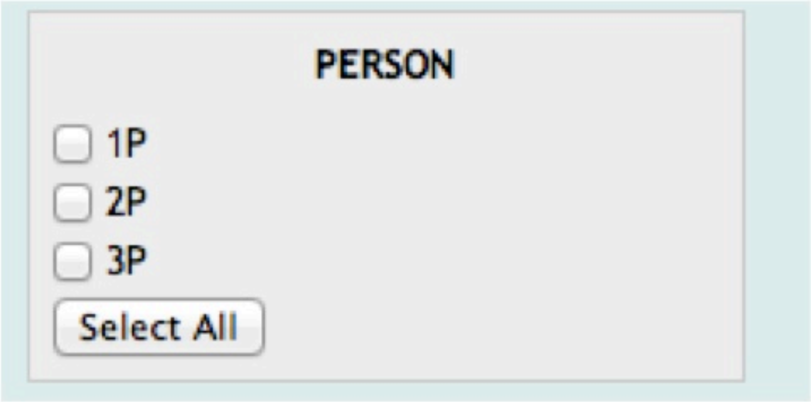
To know more about these abbreviations, please click on the URL below

http://corpus.quran.com/documentation/verbforms.jsp

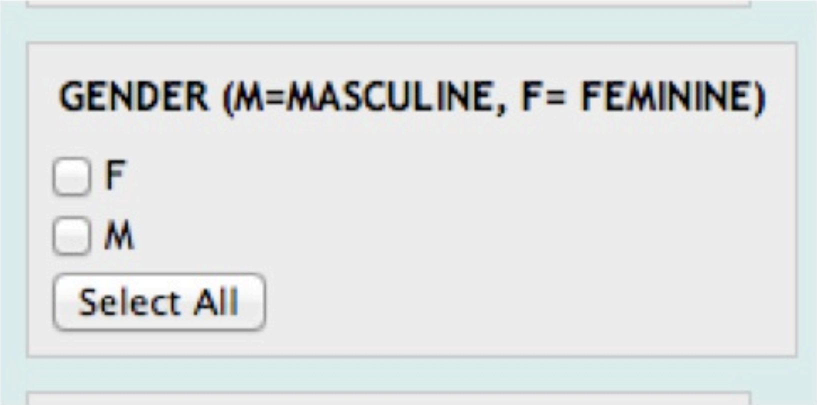
Words derivations:



First ,second and third person:



Masculine and feminine words:

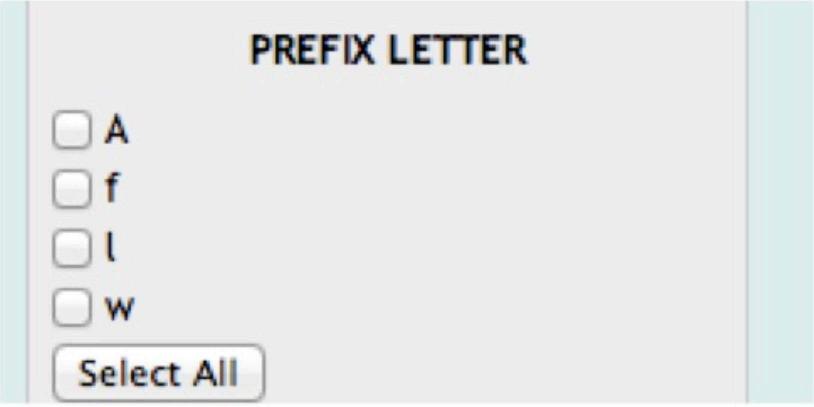


GENDER (M=MASCULINE, F= FEMININE)

□ F

□ M

Prefix letter



PREFIX LETTER

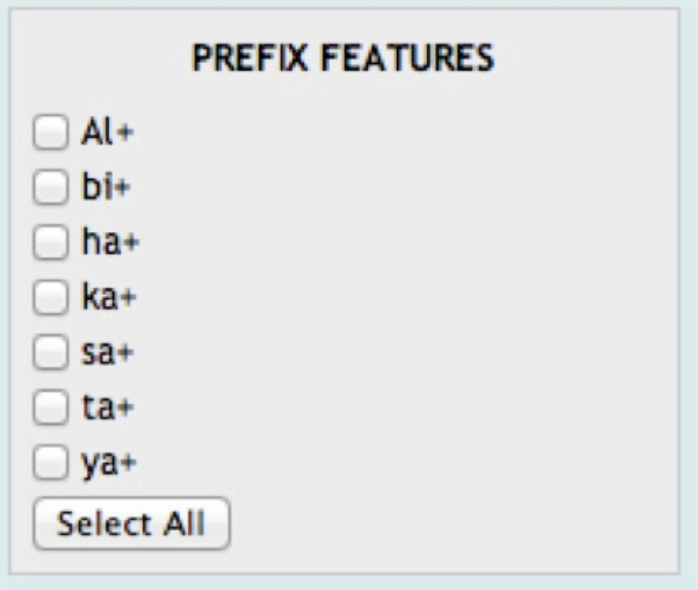
□ A

□ f

□ l

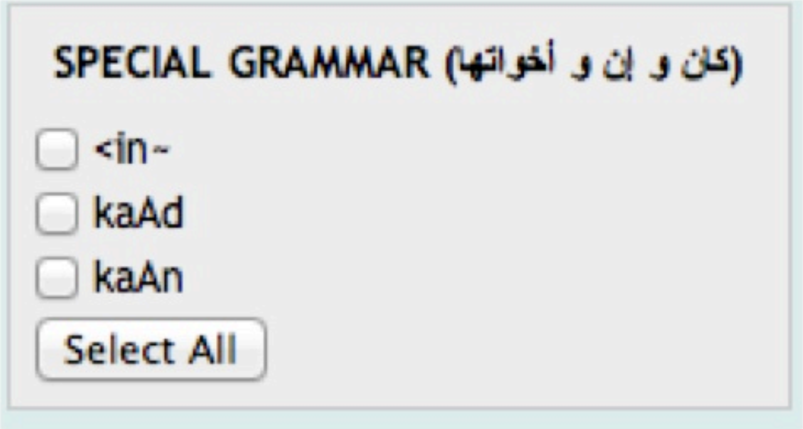
□ w

Prefix letters features


PREFIX FEATURES

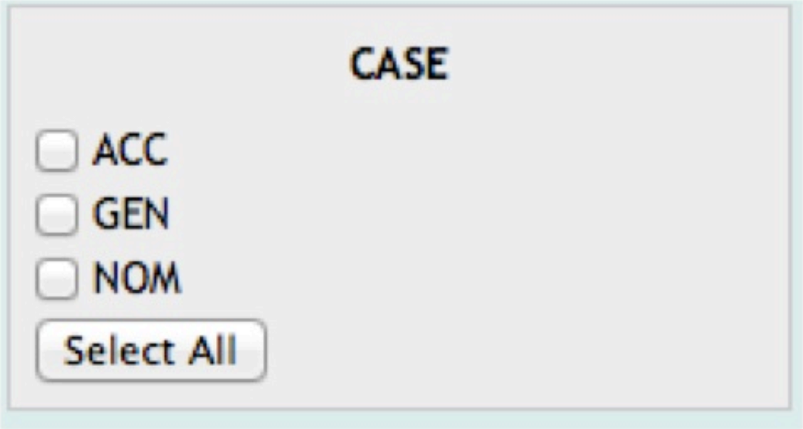
□ Al+
□ bi+
□ ha+
□ ka+
□ sa+
□ ta+
□ ya+

KaAn , KaAd and <in words


SPECIAL GRAMMAR (كان و إن و أخواتها)

□ <in~
□ kaAd
□ kaAn

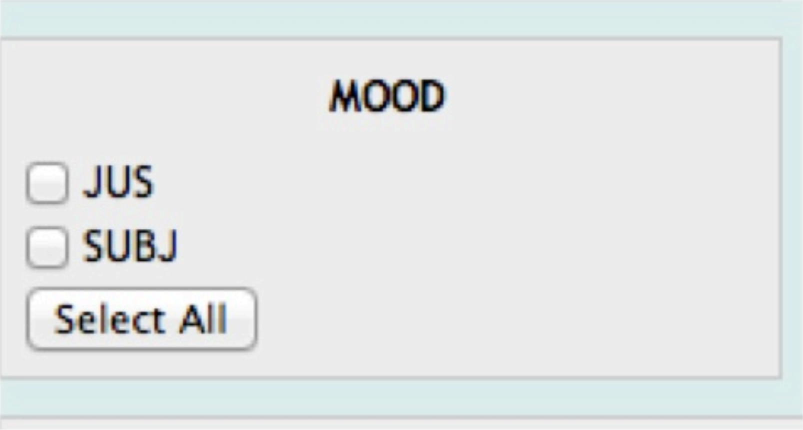
NOMinative, ACCusative and GENitive



CASE

□ ACC
□ GEN
□ NOM

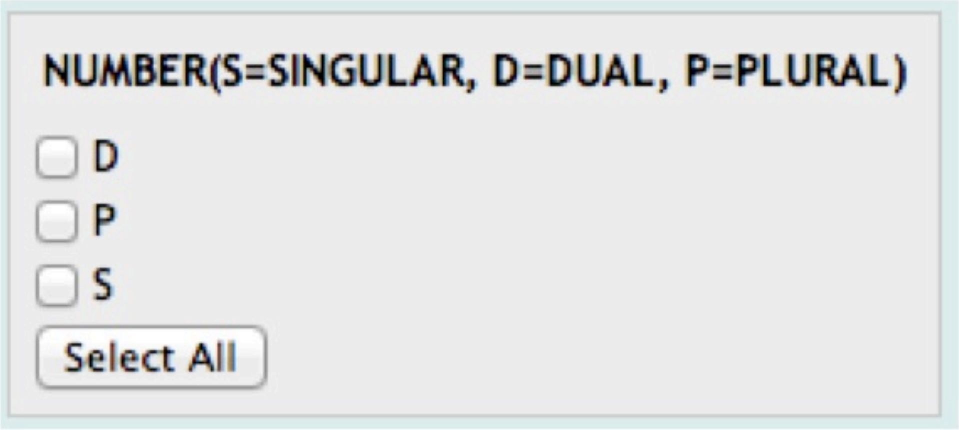
Subjunctive and jussive:



MOOD

□ JUS
□ SUBJ

Singular, Plural and dual words:



NUMBER(S=SINGULAR, D=DUAL, P=PLURAL)

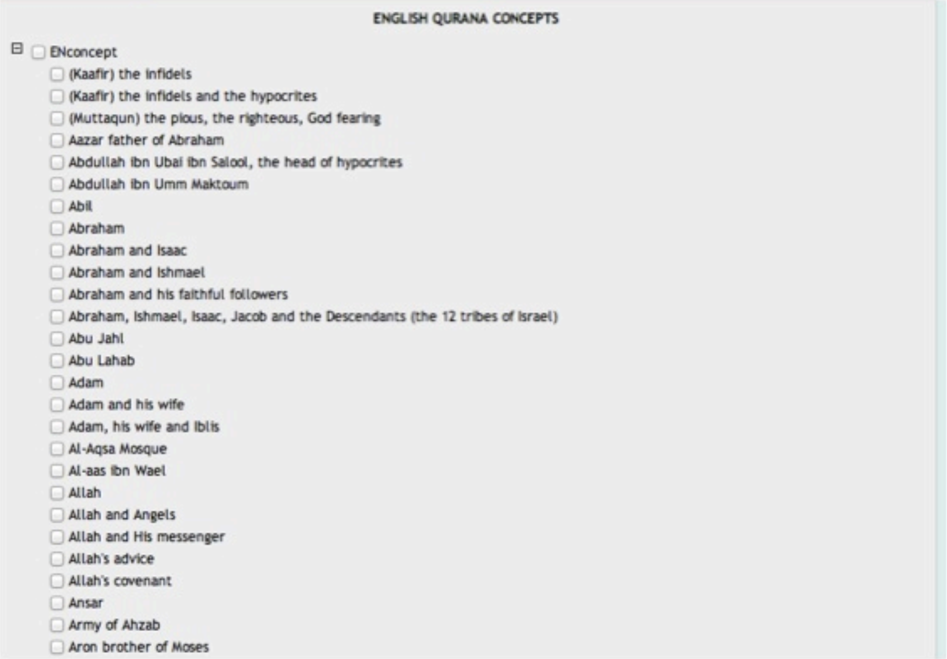
□ D

□ P

□ S

Third:
Semantic analysis for Pronouns and verses in Quran

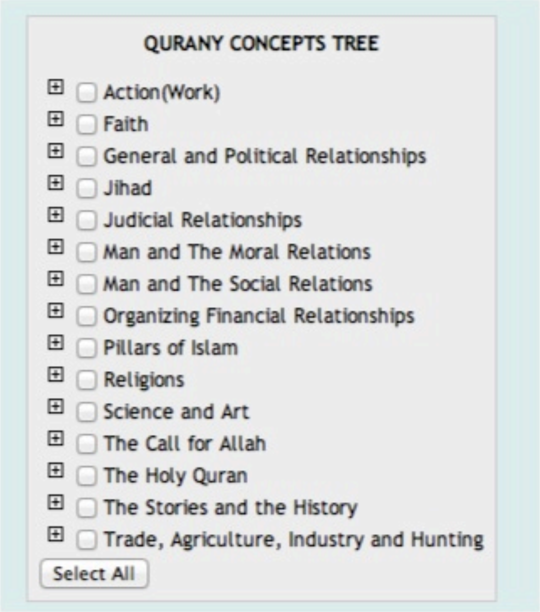
Semantic analysis for each Pronouns (to whome it refers)?



ENGLISH QURANA CONCEPTS

⊟ □ ENconcept
- □ (Kaafir) the infidels
- □ (Kaafir) the infidels and the hypocrites
- □ (Muttaqun) the pious, the righteous, God fearing
- □ Aazar father of Abraham
- □ Abdullah ibn Ubai ibn Salool, the head of hypocrites
- □ Abdullah ibn Umm Maktoum
- □ Abil
- □ Abraham
- □ Abraham and Isaac
- □ Abraham and Ishmael
- □ Abraham and his faithful followers
- □ Abraham, Ishmael, Isaac, Jacob and the Descendants (the 12 tribes of Israel)
- □ Abu Jahl
- □ Abu Lahab
- □ Adam
- □ Adam and his wife
- □ Adam, his wife and Iblis
- □ Al-Aqsa Mosque
- □ Al-aas ibn Wael
- □ Allah
- □ Allah and Angels
- □ Allah and His messenger
- □ Allah's advice
- □ Allah's covenant
- □ Ansar
- □ Army of Ahzab
- □ Aron brother of Moses

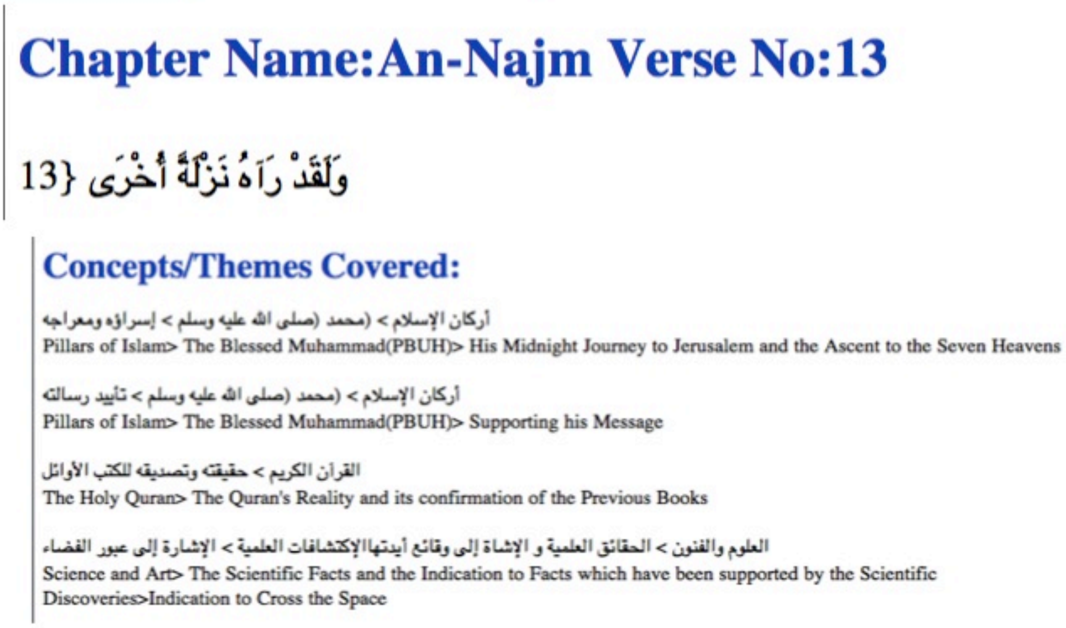
Semantic analysis for each verse in Quran



QURANY CONCEPTS TREE

- ⊞ □ Action(Work)
- ⊞ □ Faith
- ⊞ □ General and Political Relationships
- ⊞ □ Jihad
- ⊞ □ Judicial Relationships
- ⊞ □ Man and The Moral Relations
- ⊞ □ Man and The Social Relations
- ⊞ □ Organizing Financial Relationships
- ⊞ □ Pillars of Islam
- ⊞ □ Religions
- ⊞ □ Science and Art
- ⊞ □ The Call for Allah
- ⊞ □ The Holy Quran
- ⊞ □ The Stories and the History
- ⊞ □ Trade, Agriculture, Industry and Hunting

Example of semantic analysis for verse in Qurany system



Chapter Name:An-Najm Verse No:13

وَلَقَدْ رَآهُ نَزْلَةً أُخْرَى {13

Concepts/Themes Covered:

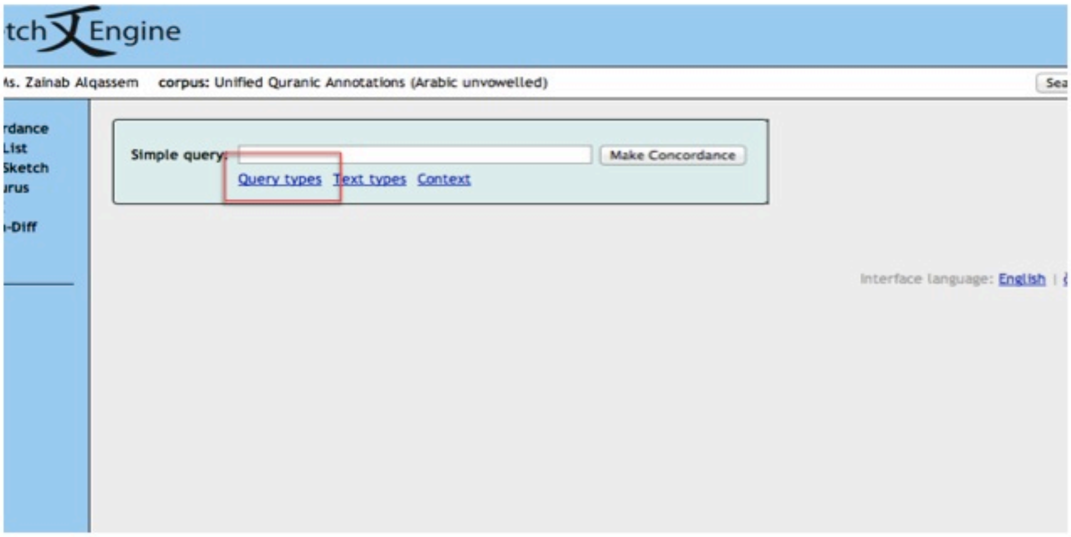
أركان الإسلام > (محمد (صلى الله عليه وسلم > إسراؤه ومعراجه
Pillars of Islam> The Blessed Muhammad(PBUH)> His Midnight Journey to Jerusalem and the Ascent to the Seven Heavens

أركان الإسلام > (محمد (صلى الله عليه وسلم > تأييد رسالته
Pillars of Islam> The Blessed Muhammad(PBUH)> Supporting his Message

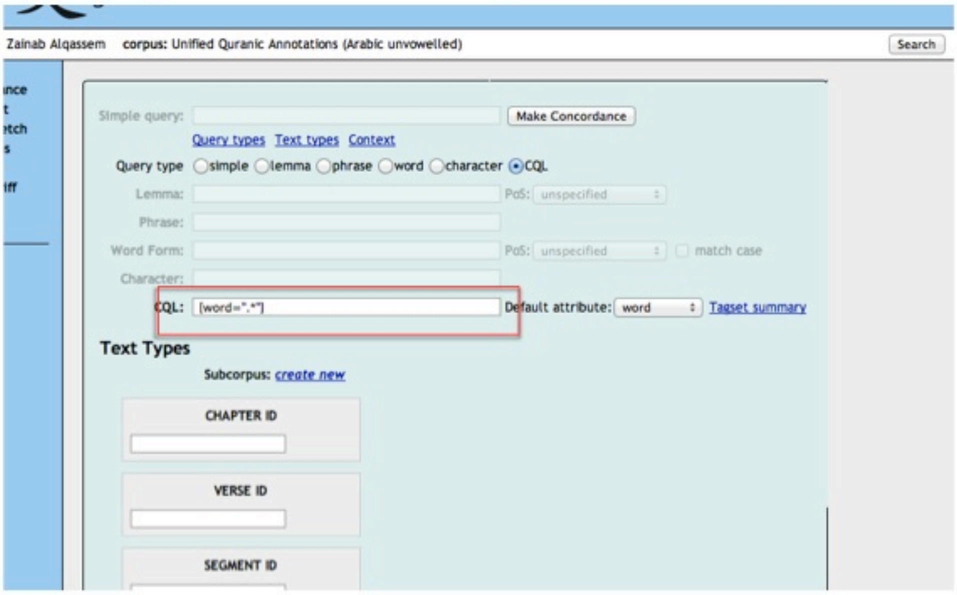
القرآن الكريم > حقيقته وتصديقه للكتب الأوائل
The Holy Quran> The Quran's Reality and its confirmation of the Previous Books

العلوم والفنون > الحقائق العلمية و الإشاة إلى وقائع أيدتها الإكتشافات العلمية > الإشارة إلى عبور الفضاء
Science and Art> The Scientific Facts and the Indication to Facts which have been supported by the Scientific Discoveries>Indication to Cross the Space

1- To extact ALL Quranic words with F gender type



In CQL, type the expression [word=".*"]

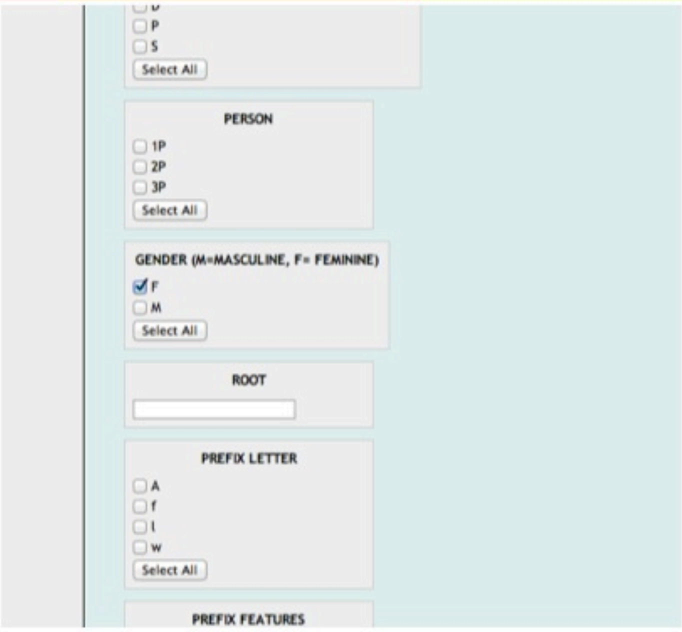


Zainab Alqassem   corpus: Unified Quranic Annotations (Arabic unvowelled)

Query type ○simple ○lemma ○phrase ○word ○character ●CQL

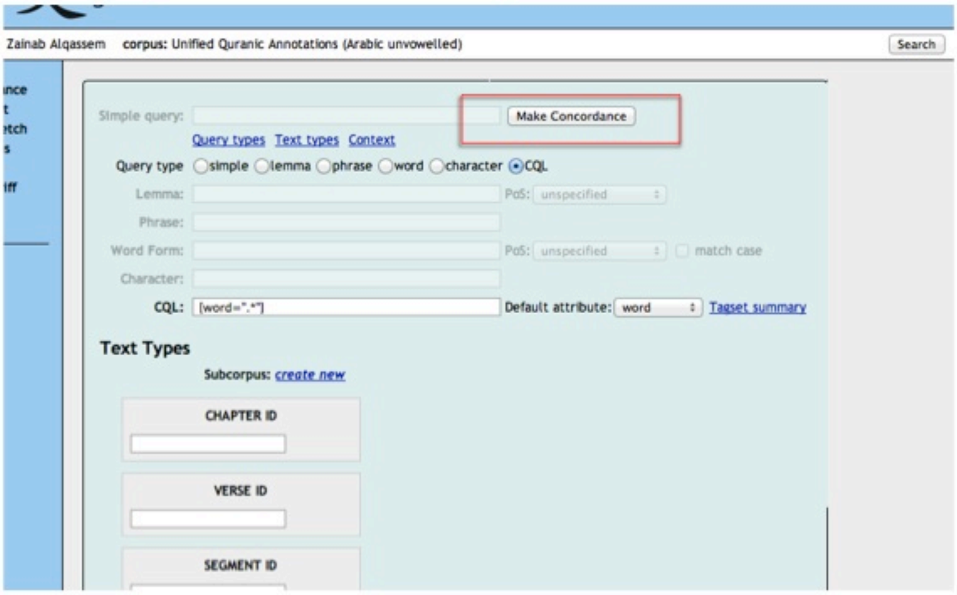
CQL: [word=".*"]   Default attribute: word

From Text type options, choose F in Gender options



GENDER (M=MASCULINE, F= FEMININE)
☑ F
□ M

Then click on Make Concordance



Zainab Alqassem corpus: Unified Quranic Annotations (Arabic unvowelled)

Simple query:

Query type ○simple ○lemma ○phrase ○word ○character ●CQL

Lemma: PoS: unspecified

Phrase:

Word Form: PoS: unspecified □ match case

Character:

CQL: [word=".*"] Default attribute: word

Text Types

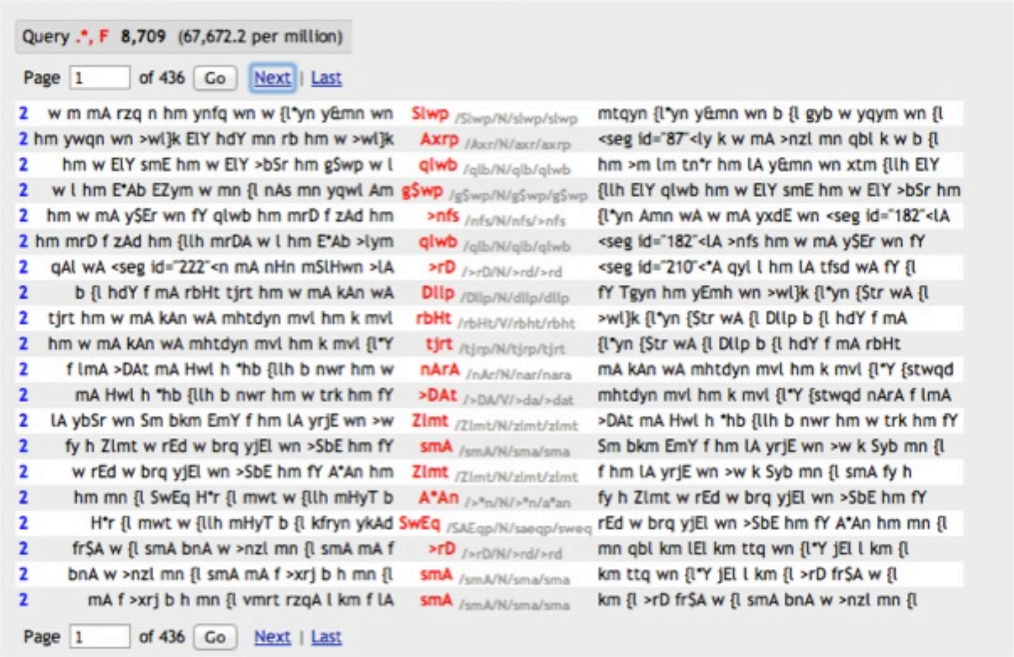
Subcorpus: *create new*

CHAPTER ID

VERSE ID

SEGMENT ID

At the top, you can see the number of feminine words in Quran.



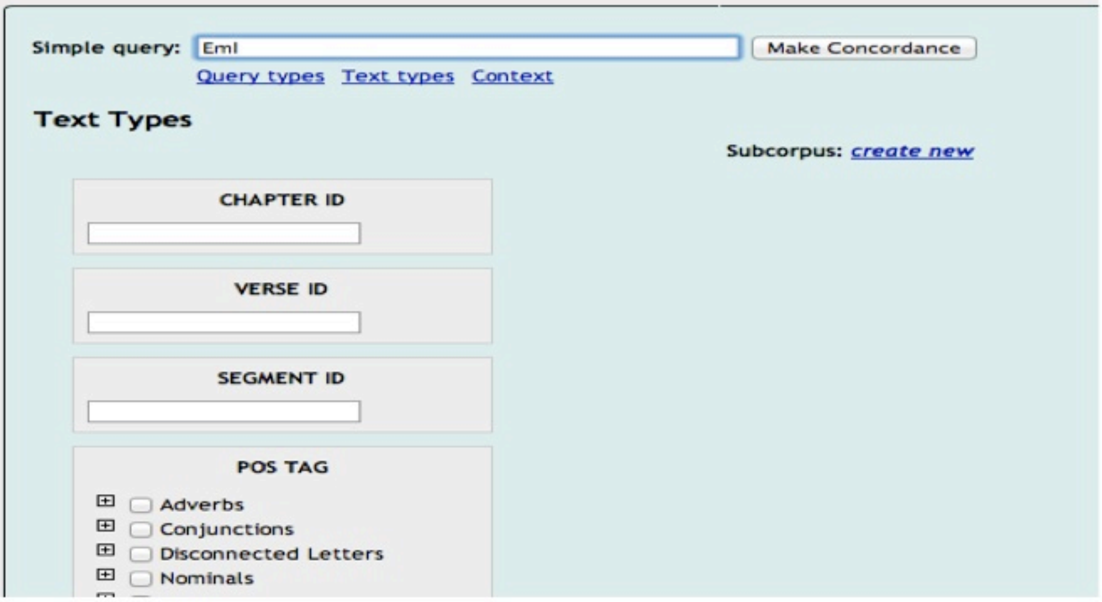
Query .*, F 8,709 (67,672.2 per million)

Page 1 of 436 Go Next | Last

2	w m mA rzq n hm ynfq wn w {l*yn y&mn wn	Slwp	/Slwp/N/slwp/slwp	mtqyn {l*yn y&mn wn b {l gyb w yqym wn {l
2	hm ywqn wn >wl}k ElY hdY mn rb hm w >wl}k	Axrp	/Axr/N/axr/axrp	<seg id="87"<ly k w mA >nzl mn qbl k w b {l
2	hm w ElY smE hm w ElY >bSr hm g$wp w l	qlwb	/qlb/N/qlb/qlwb	hm >m lm tn*r hm lA y&mn wn xtm {llh ElY
2	w l hm E*Ab EZym w mn {l nAs mn yqwl Am	g$wp	/g$wp/N/g$wp/g$wp	{llh ElY qlwb hm w ElY smE hm w ElY >bSr hm
2	hm w mA y$Er wn fY qlwb hm mrD f zAd hm	>nfs	/nfs/N/nfs/>nfs	{l*yn Amn wA w mA yxdE wn <seg id="182"<lA
2	hm mrD f zAd hm {llh mrDA w l hm E*Ab >lym	qlwb	/qlb/N/qlb/qlwb	<seg id="182"<lA >nfs hm w mA y$Er wn fY
2	qAl wA <seg id="222"<n mA nHn mSlHwn >lA	>rD	/>rD/N/>rd/>rd	<seg id="210"<*A qyl l hm lA tfsd wA fY {l
2	b {l hdY f mA rbHt tjrt hm w mA kAn wA	Dllp	/Dllp/N/dllp/dllp	fY Tgyn hm yEmh wn >wl}k {l*yn {$tr wA {l
2	tjrt hm w mA kAn wA mhtdyn mvl hm k mvl	rbHt	/rbHt/V/rbht/rbht	>wl}k {l*yn {$tr wA {l Dllp b {l hdY f mA
2	hm w mA kAn wA mhtdyn mvl hm k mvl {l*Y	tjrt	/tjrp/N/tjrp/tjrt	{l*yn {$tr wA {l Dllp b {l hdY f mA rbHt
2	f lmA >DAt mA Hwl h *hb {llh b nwr hm w	nArA	/nAr/N/nar/nara	mA kAn wA mhtdyn mvl hm k mvl {l*Y {stwqd
2	mA Hwl h *hb {llh b nwr hm w trk hm fY	>DAt	/>DA/V/>da/>dat	mhtdyn mvl hm k mvl {l*Y {stwqd nArA f lmA
2	lA ybSr wn Sm bkm EmY f hm lA yrjE wn >w	Zlmt	/Zlmt/N/zlmt/zlmt	>DAt mA Hwl h *hb {llh b nwr hm w trk hm fY
2	fy h Zlmt w rEd w brq yjEl wn >SbE hm fY	smA	/smA/N/sma/sma	Sm bkm EmY f hm lA yrjE wn >w k Syb mn {l
2	w rEd w brq yjEl wn >SbE hm fY A*An hm	Zlmt	/Zlmt/N/zlmt/zlmt	f hm lA yrjE wn >w k Syb mn {l smA fy h
2	hm mn {l SwEq H*r {l mwt w {llh mHyT b	A*An	/>*n/N/>*n/a*an	fy h Zlmt w rEd w brq yjEl wn >SbE hm fY
2	H*r {l mwt w {llh mHyT b {l kfryn ykAd	SwEq	/SAEqp/N/saeqp/sweq	rEd w brq yjEl wn >SbE hm fY A*An hm mn {l
2	fr$A w {l smA bnA w >nzl mn {l smA mA f	>rD	/>rD/N/>rd/>rd	mn qbl km lEl km ttq wn {l*Y jEl l km {l
2	bnA w >nzl mn {l smA mA f >xrj b h mn {l	smA	/smA/N/sma/sma	km ttq wn {l*Y jEl l km {l >rD fr$A w {l
2	mA f >xrj b h mn {l vmrt rzqA l km f lA	smA	/smA/N/sma/sma	km {l >rD fr$A w {l smA bnA w >nzl mn {l

Page 1 of 436 Go Next | Last

Also, you can search by any word by typing it in Simple query box. But, you have to make sure that you type the word without any prefix or suffix.



Simple query: Eml [Make Concordance]

Query types Text types Context

Text Types

Subcorpus: *create new*

CHAPTER ID

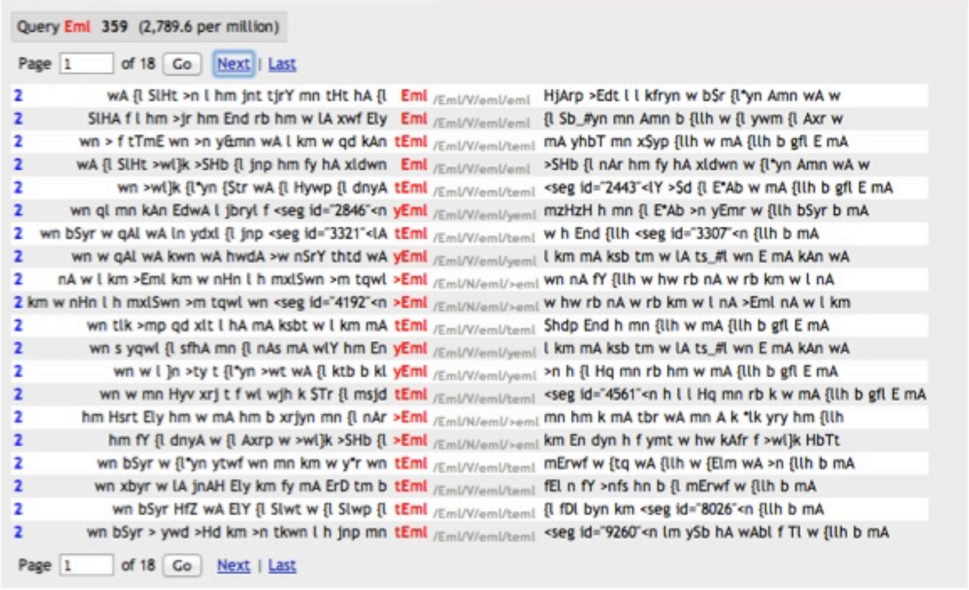
VERSE ID

SEGMENT ID

POS TAG

- ⊞ □ Adverbs
- ⊞ □ Conjunctions
- ⊞ □ Disconnected Letters
- ⊞ □ Nominals

Results:



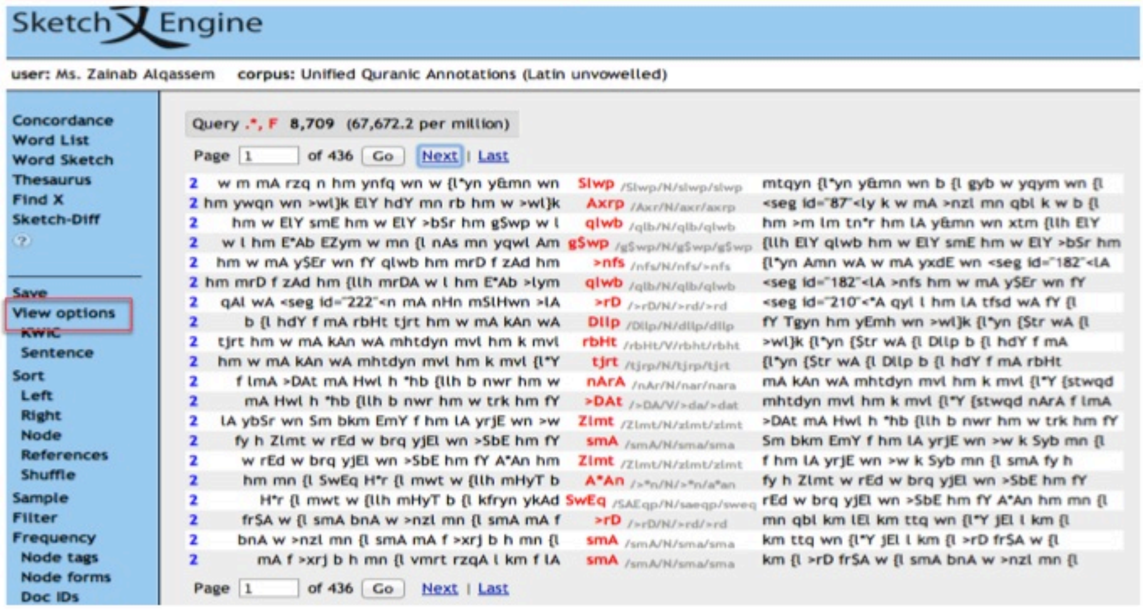
Query Eml 359 (2,789.6 per million)

Page 1 of 18 Go Next | Last

2	wA {l SlHt >n l hm jnt tjrY mn tHt hA {l	Eml	/Eml/V/eml/eml	HjArp >Edt l l kfryn w b$r {l*yn Amn wA w
2	SlHA f l hm >jr hm End rb hm w lA xwf Ely	Eml	/Eml/V/eml/eml	{l Sb_#yn mn Amn b {llh w {l ywm {l Axr w
2	wn > f tTmE wn >n y&mn wA l km w qd kAn	tEml	/Eml/V/eml/teml	mA yhbT mn x$yp {llh w mA {llh b gfl E mA
2	wA {l SlHt >wl}k >SHb {l jnp hm fy hA xldwn	Eml	/Eml/V/eml/eml	>SHb {l nAr hm fy hA xldwn w {l*yn Amn wA w
2	wn >wl}k {l*yn {Str wA {l Hywp {l dnyA	tEml	/Eml/V/eml/teml	<seg id="2443"<lY >$d {l E*Ab w mA {llh b gfl E mA
2	wn ql mn kAn EdwA l jbryl f <seg id="2846"<n	yEml	/Eml/V/eml/yeml	mzHzH h mn {l E*Ab >n yEmr w {llh bSyr b mA
2	wn bSyr w qAl wA ln ydxl {l jnp <seg id="3321"<lA	tEml	/Eml/V/eml/teml	w h End {llh <seg id="3307"<n {llh b mA
2	wn w qAl wA kwn wA hwdA >w nSrY thtd wA	yEml	/Eml/V/eml/yeml	l km mA ksb tm w lA ts_#l wn E mA kAn wA
2	nA w l km >Eml km w nHn l h mxlSwn >m tqwl	>Eml	/Eml/N/eml/>eml	wn nA fY {llh w hw rb nA w rb km w l nA
2	km w nHn l h mxlSwn >m tqwl wn <seg id="4192"<n	>Eml	/Eml/N/eml/>eml	w hw rb nA w rb km w l nA >Eml nA w l km
2	wn tlk >mp qd xlt l hA mA ksbt w l km mA	tEml	/Eml/V/eml/teml	$hdp End h mn {llh w mA {llh b gfl E mA
2	wn s yqwl {l sfhA mn {l nAs mA wlY hm En	yEml	/Eml/V/eml/yeml	l km mA ksb tm w lA ts_#l wn E mA kAn wA
2	wn w l }n >ty t {l*yn >wt wA {l ktb b kl	yEml	/Eml/V/eml/yeml	>n h {l Hq mn rb hm w mA {llh b gfl E mA
2	wn w mn Hyv xrj t f wl wjh k $Tr {l msjd	tEml	/Eml/V/eml/teml	<seg id="4561"<n h l l Hq mn rb k w mA {llh b gfl E mA
2	hm Hsrt Ely hm w mA hm b xrjyn mn {l nAr	>Eml	/Eml/N/eml/>eml	mn hm k mA tbr wA mn A k *lk yry hm {llh
2	hm fY {l dnyA w {l Axrp w >wl}k >SHb {l	>Eml	/Eml/N/eml/>eml	km En dyn h f ymt w hw kAfr f >wl}k HbTt
2	wn bSyr w {l*yn ytwf wn mn km w y*r wn	tEml	/Eml/V/eml/teml	mErwf w {tq wA {llh w {Elm wA >n {llh b mA
2	wn xbyr w lA jnAH Ely km fy mA ErD tm b	tEml	/Eml/V/eml/teml	fEl n fY >nfs hn b {l mErwf w {llh b mA
2	wn bSyr HfZ wA ElY {l Slwt w {l Slwp {l	tEml	/Eml/V/eml/teml	{l fDl byn km <seg id="8026"<n {llh b mA
2	wn bSyr > ywd >Hd km >n tkwn l h jnp mn	tEml	/Eml/V/eml/teml	<seg id="9260"<n lm ySb hA wAbl f Tl w {llh b mA

Page 1 of 18 Go Next | Last

To view the results with chapter and verse id:



user: Ms. Zainab Alqassem   corpus: Unified Quranic Annotations (Latin unvowelled)

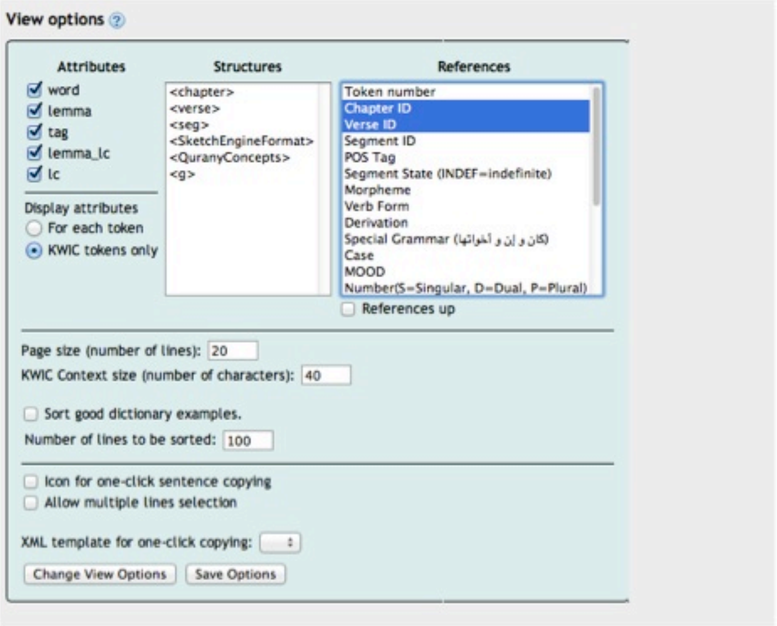
Query .*, F 8,709 (67,672.2 per million)

Page 1 of 436 Go Next | Last

```
2   w m mA rzq n hm ynfq wn w {l*yn y&mn wn   Slwp /Slwp/N/slwp/slwp   mtqyn {l*yn y&mn wn b {l gyb w yqym wn {l
2 hm ywqn wn >wl}k ElY hdY mn rb hm w >wl}k   Axrp /Axr/N/axr/axrp   <seg id="87"<ly k w mA >nzl mn qbl k w b {l
2   hm w ElY smE hm w ElY >bSr hm g$wp w l   qlwb /qlb/N/qlb/qlwb   hm >m lm tn*r hm lA y&mn wn xtm {llh ElY
2   w l hm E*Ab EZym w mn {l nAs mn yqwl Am   g$wp /g$wp/N/g$wp/g$wp   {llh ElY qlwb hm w ElY smE hm w ElY >bSr hm
2   hm w mA y$Er wn fY qlwb hm mrD f zAd hm   >nfs /nfs/N/nfs/>nfs   {l*yn Amn wA w mA yxdE wn <seg id="182"<lA
2 hm mrD f zAd hm {llh mrDA w l hm E*Ab >lym   qlwb /qlb/N/qlb/qlwb   <seg id="182"<lA >nfs hm w mA y$Er wn fY
2   qAl wA <seg id="222"<n mA nHn mSlHwn >lA   >rD />rD/N/>rd/>rd   <seg id="210"<*A qyl l hm lA tfsd wA fY {l
2   b {l hdY f mA rbHt tjrt hm w mA kAn wA   Dllp /Dllp/N/dllp/dllp   fY Tgyn hm yEmh wn >wl}k {l*yn {$tr wA {l
2   tjrt hm w mA kAn wA mhtdyn mvl hm k mvl   rbHt /rbHt/V/rbht/rbht   >wl}k {l*yn {$tr wA {l Dllp b {l hdY f mA
2   hm w mA kAn wA mhtdyn mvl hm k mvl {l*Y   tjrt /tjrp/N/tjrp/tjrt   {l*yn {$tr wA {l Dllp b {l hdY f mA rbHt
2   f lmA >DAt mA Hwl h *hb {llh b nwr hm w   nArA /nAr/N/nar/nara   mA kAn wA mhtdyn mvl hm k mvl {l*Y {stwqd
2   mA Hwl h *hb {llh b nwr hm w trk hm fY   >DAt />DA/V/>da/>dat   mhtdyn mvl hm k mvl {l*Y {stwqd nArA f lmA
2   lA ybSr wn Sm bkm EmY f hm lA yrjE wn >w   Zlmt /Zlmt/N/zlmt/zlmt   >DAt mA Hwl h *hb {llh b nwr hm w trk hm fY
2   fy h Zlmt w rEd w brq yjEl wn >SbE hm fY   smA /smA/N/sma/sma   Sm bkm EmY f hm lA yrjE wn >w k Syb mn {l
2   w rEd w brq yjEl wn >SbE hm fY A*An hm   Zlmt /Zlmt/N/zlmt/zlmt   f hm lA yrjE wn >w k Syb mn {l smA fy h
2   hm mn {l SwEq H*r {l mwt w {llh mHyT b   A*An />*n/N/>*n/a*an   fy h Zlmt w rEd w brq yjEl wn >SbE hm fY
2   H*r {l mwt w {llh mHyT b {l kfryn ykAd   SwEq /SAEqp/N/saeqp/sweq   rEd w brq yjEl wn >SbE hm fY A*An hm mn {l
2   fr$A w {l smA bnA w >nzl mn {l smA mA f   >rD />rD/N/>rd/>rd   mn qbl km lEl km ttq wn {l*Y jEl l km {l
2   bnA w >nzl mn {l smA mA f >xrj b h mn {l   smA /smA/N/sma/sma   km ttq wn {l*Y jEl l km {l >rD fr$A w {l
2   mA f >xrj b h mn {l vmrt rzqA l km f lA   smA /smA/N/sma/sma   km {l >rD fr$A w {l smA bnA w >nzl mn {l
```

Page 1 of 436 Go Next | Last

From (Reference) you can choose more than one choice



View options

Attributes
- ☑ word
- ☑ lemma
- ☑ tag
- ☑ lemma_lc
- ☑ lc

Display attributes
- ○ For each token
- ● KWIC tokens only

Structures
- <chapter>
- <verse>
- <seg>
- <SketchEngineFormat>
- <QuranyConcepts>
- <g>

References
- Token number
- Chapter ID
- Verse ID
- Segment ID
- POS Tag
- Segment State (INDEF=indefinite)
- Morpheme
- Verb Form
- Derivation
- Special Grammar (كان و إن و أخواتها)
- Case
- MOOD
- Number(S=Singular, D=Dual, P=Plural)

□ References up

Page size (number of lines): 20

KWIC Context size (number of characters): 40

□ Sort good dictionary examples.

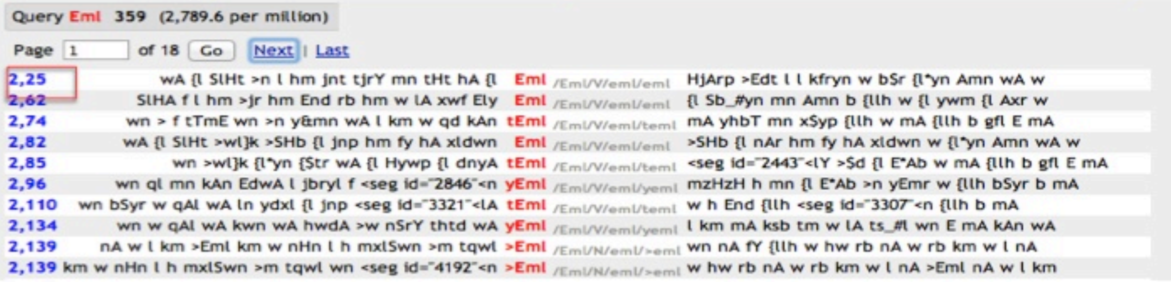
Number of lines to be sorted: 100

□ Icon for one-click sentence copying

□ Allow multiple lines selection

XML template for one-click copying:

For example, word (Eml) is in chapter 2 and verse 25.

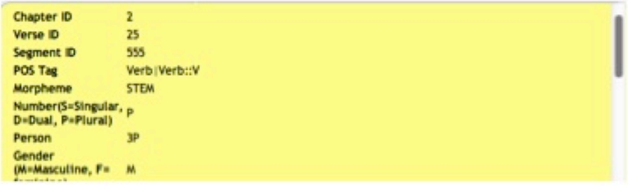


Query Eml 359 (2,789.6 per million)

Page 1 of 18 Go Next | Last

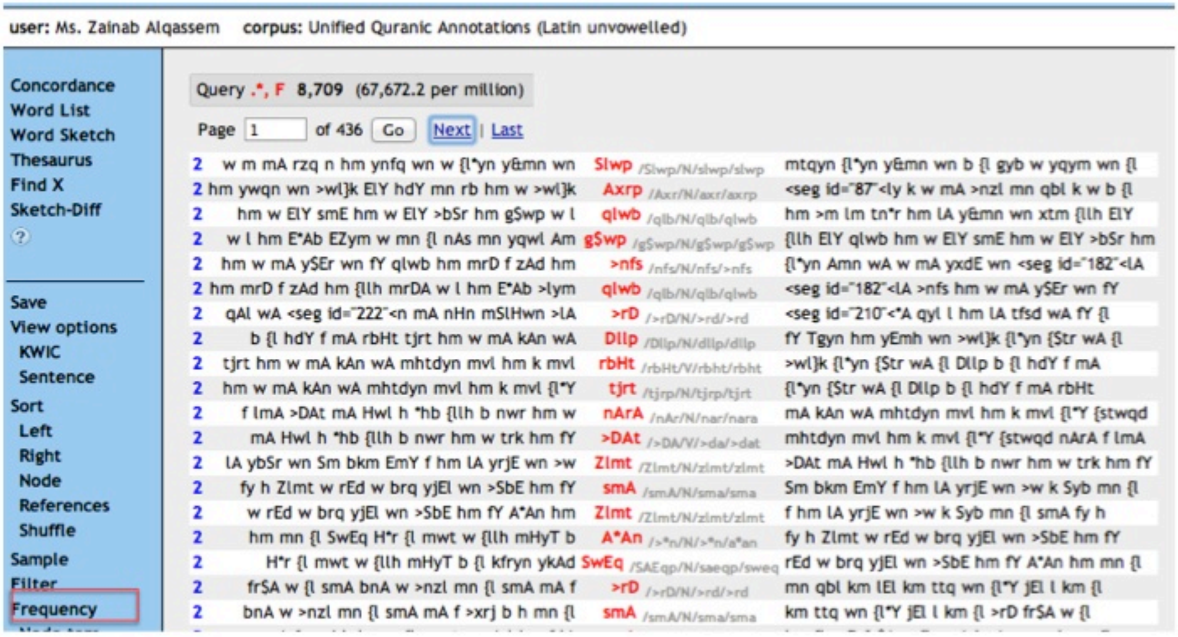
2,25	wA {l SlHt >n l hm jnt tjrY mn tHt hA {l	Eml	/Eml/V/eml/eml	HjArp >Edt l l kfryn w bSr {l*yn Amn wA w
2,62	SlHA f l hm >jr hm End rb hm w lA xwf Ely	Eml	/Eml/V/eml/eml	{l Sb_#yn mn Amn b {llh w {l ywm {l Axr w
2,74	wn > f tTmE wn >n y&mn wA l km w qd kAn	tEml	/Eml/V/eml/teml	mA yhbT mn x$yp {llh w mA {llh b gfl E mA
2,82	wA {l SlHt >wl}k >SHb {l jnp hm fy hA xldwn	Eml	/Eml/V/eml/eml	>SHb {l nAr hm fy hA xldwn w {l*yn Amn wA w
2,85	wn >wl}k {l*yn {$tr wA {l Hywp {l dnyA	tEml	/Eml/V/eml/teml	<seg id="2443"<lY >$d {l E*Ab w mA {llh b gfl E mA
2,96	wn ql mn kAn EdwA l jbryl f <seg id="2846"<n	yEml	/Eml/V/eml/yeml	mzHzH h mn {l E*Ab >n yEmr w {llh bSyr b mA
2,110	wn bSyr w qAl wA ln ydxl {l jnp <seg id="3321"<lA	tEml	/Eml/V/eml/teml	w h End {llh <seg id="3307"<n {llh b mA
2,134	wn w qAl wA kwn wA hwdA >w nSrY thtd wA	yEml	/Eml/V/eml/yeml	l km mA ksb tm w lA ts_#l wn E mA kAn wA
2,139	nA w l km >Eml km w nHn l h mxlSwn >m tqwl	>Eml	/Eml/N/eml/>eml	wn nA fY {llh w hw rb nA w rb km w l nA
2,139	km w nHn l h mxlSwn >m tqwl wn <seg id="4192"<n	>Eml	/Eml/N/eml/>eml	w hw rb nA w rb km w l nA >Eml nA w l km

By clicking on the blue colour, you will be able to see all inforamtion related to that word



Chapter ID	2
Verse ID	25
Segment ID	555
POS Tag	Verb|Verb::V
Morpheme	STEM
Number(S=Singular, D=Dual, P=Plural)	P
Person	3P
Gender (M=Masculine, F=	M

By clicking on (Frequency) you can see how many times this word appears as Noun and Verb



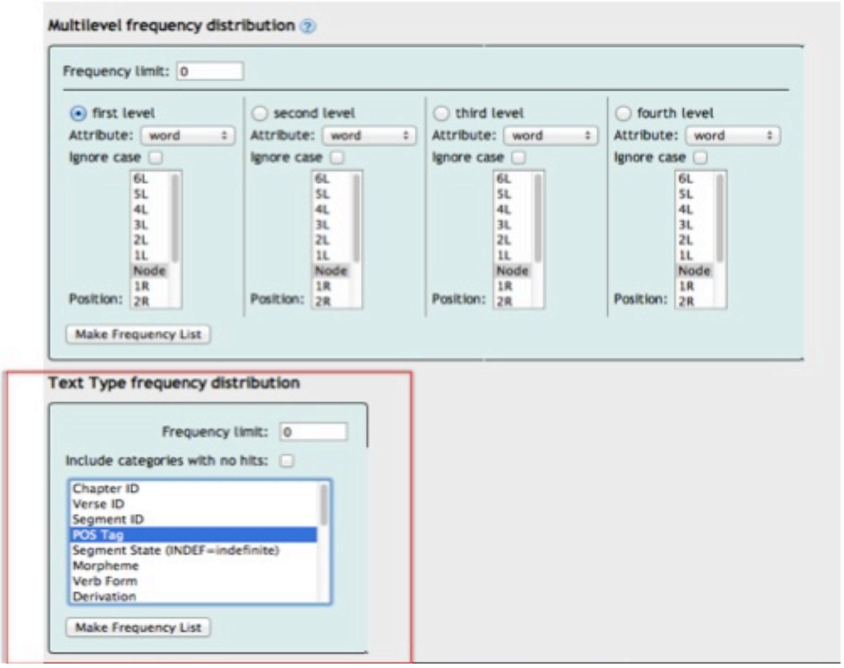
user: Ms. Zainab Alqassem corpus: Unified Quranic Annotations (Latin unvowelled)

Query .*, F 8,709 (67,672.2 per million)

Page 1 of 436 Go Next | Last

2	w m mA rzq n hm ynfq wn w {l*yn y&mn wn	Slwp	/Slwp/N/slwp/slwp	mtqyn {l*yn y&mn wn b {l gyb w yqym wn {l
2	hm ywqn wn >wl}k ElY hdY mn rb hm w >wl}k	Axrp	/Axr/N/axr/axrp	<seg id="87"<ly k w mA >nzl mn qbl k w b {l
2	hm w ElY smE hm w ElY >bSr hm g$wp w l	qlwb	/qlb/N/qlb/qlwb	hm >m lm tn*r hm lA y&mn wn xtm {llh ElY
2	w l hm E*Ab EZym w mn {l nAs mn yqwl Am	g$wp	/g$wp/N/g$wp/g$wp	{llh ElY qlwb hm w ElY smE hm w ElY >bSr hm
2	hm w mA y$Er wn fY qlwb hm mrD f zAd hm	>nfs	/nfs/N/nfs/>nfs	{l*yn Amn wA w mA yxdE wn <seg id="182"<lA
2	hm mrD f zAd hm {llh mrDA w l hm E*Ab >lym	qlwb	/qlb/N/qlb/qlwb	<seg id="182"<lA >nfs hm w mA y$Er wn fY
2	qAl wA <seg id="222"<n mA nHn mSlHwn >lA	>rD	/>rD/N/>rd/>rd	<seg id="210"<*A qyl l hm lA tfsd wA fY {l
2	b {l hdY f mA rbHt tjrt hm w mA kAn wA	Dllp	/Dllp/N/dllp/dllp	fY Tgyn hm yEmh wn >wl}k {l*yn {$tr wA {l
2	tjrt hm w mA kAn wA mhtdyn mvl hm k mvl	rbHt	/rbHt/V/rbht/rbht	>wl}k {l*yn {$tr wA {l Dllp b {l hdY f mA
2	hm w mA kAn wA mhtdyn mvl hm k mvl {l*Y	tjrt	/tjrp/N/tjrp/tjrt	{l*yn {$tr wA {l Dllp b {l hdY f mA rbHt
2	f lmA >DAt mA Hwl h *hb {llh b nwr hm w	nArA	/nAr/N/nar/nara	mA kAn wA mhtdyn mvl hm k mvl {l*Y {stwqd
2	mA Hwl h *hb {llh b nwr hm w trk hm fY	>DAt	/>DA/V/>da/>dat	mhtdyn mvl hm k mvl {l*Y {stwqd nArA f lmA
2	lA ybSr wn Sm bkm EmY f hm lA yrjE wn >w	Zlmt	/Zlmt/N/zlmt/zlmt	>DAt mA Hwl h *hb {llh b nwr hm w trk hm fY
2	fy h Zlmt w rEd w brq yjEl wn >SbE hm fY	smA	/smA/N/sma/sma	Sm bkm EmY f hm lA yrjE wn >w k Syb mn {l
2	w rEd w brq yjEl wn >SbE hm fY A*An hm	Zlmt	/Zlmt/N/zlmt/zlmt	f hm lA yrjE wn >w k Syb mn {l smA fy h
2	hm mn {l SwEq H*r {l mwt w {llh mHyT b	A*An	/>*n/N/>*n/a*an	fy h Zlmt w rEd w brq yjEl wn >SbE hm fY
2	H*r {l mwt w {llh mHyT b {l kfryn ykAd	SwEq	/SAEqp/N/saeqp/sweq	rEd w brq yjEl wn >SbE hm fY A*An hm mn {l
2	fr$A w {l smA bnA w >nzl mn {l smA mA f	>rD	/>rD/N/>rd/>rd	mn qbl km lEl km ttq wn {l*Y jEl l km {l
2	bnA w >nzl mn {l smA mA f >xrj b h mn {l	smA	/smA/N/sma/sma	km ttq wn {l*Y jEl l km {l >rD fr$A w {l

Choose your frequency type



Multilevel frequency distribution

Frequency limit: 0

● first level	○ second level	○ third level	○ fourth level
Attribute: word	Attribute: word	Attribute: word	Attribute: word
Ignore case □	Ignore case □	Ignore case □	Ignore case □
Position: Node	Position: Node	Position: Node	Position: Node

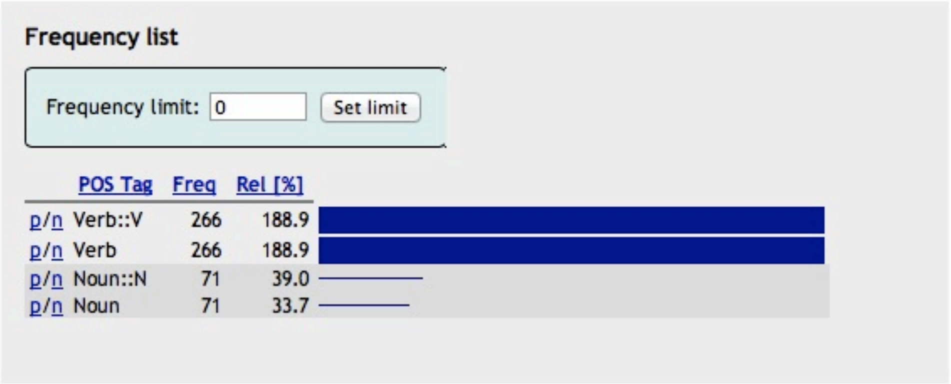
Text Type frequency distribution

Frequency limit: 0

Include categories with no hits: □

- Chapter ID
- Verse ID
- Segment ID
- POS Tag
- Segment State (INDEF=indefinite)
- Morpheme
- Verb Form
- Derivation

Results:

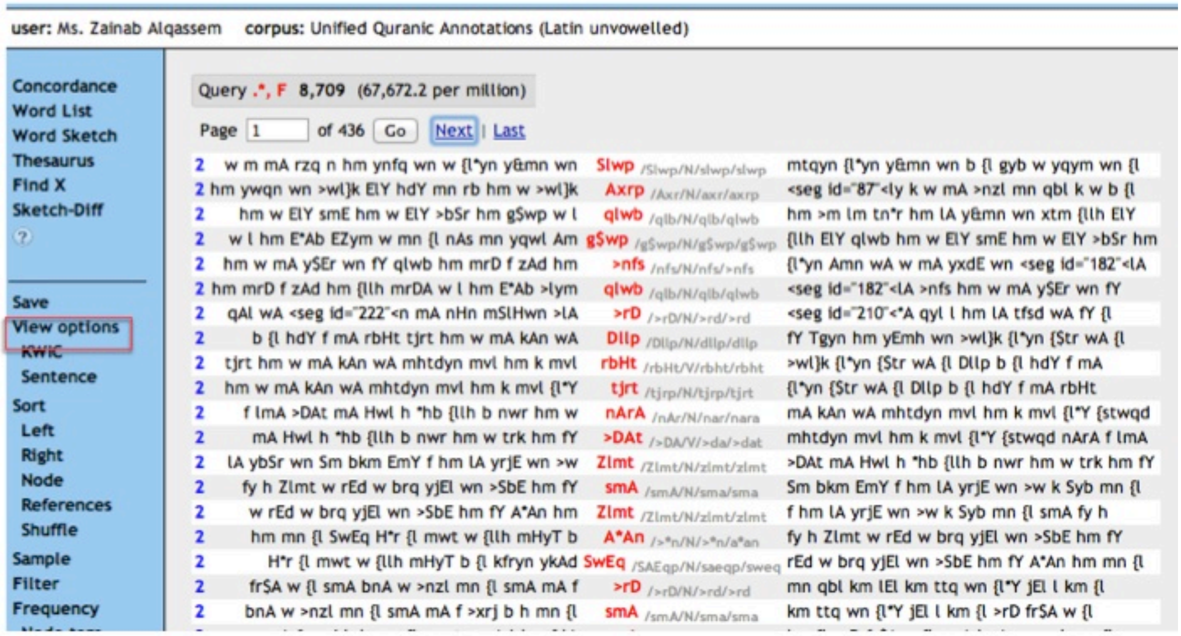


Frequency list

Frequency limit: 0

	POS Tag	Freq	Rel [%]
p/n	Verb::V	266	188.9
p/n	Verb	266	188.9
p/n	Noun::N	71	39.0
p/n	Noun	71	33.7

To show the semantic meaning for word(Eml), click again on View options



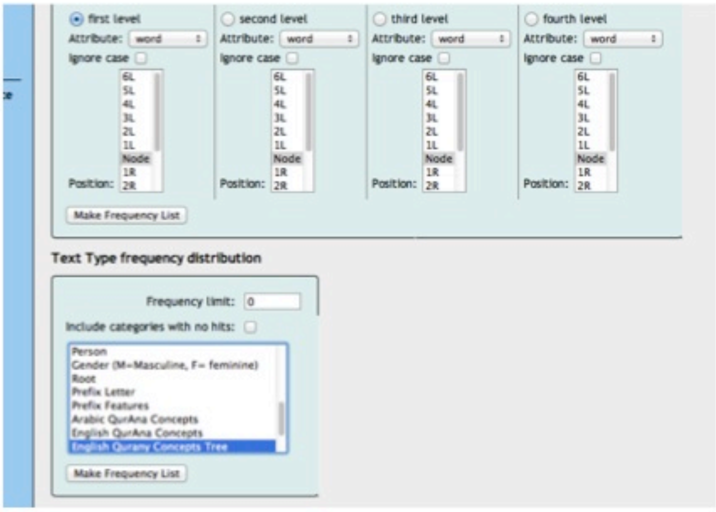
user: Ms. Zainab Alqassem corpus: Unified Quranic Annotations (Latin unvowelled)

Query .*, F 8,709 (67,672.2 per million)

Page 1 of 436

```
2    w m mA rzq n hm ynfq wn w {l*yn y&mn wn   Slwp /Slwp/N/slwp/slwp     mtqyn {l*yn y&mn wn b {l gyb w yqym wn {l
2 hm ywqn wn >wl}k ElY hdY mn rb hm w >wl}k    Axrp /Axr/N/axr/axrp       <seg id="87"<ly k w mA >nzl mn qbl k w b {l
2      hm w ElY smE hm w ElY >bSr hm g$wp w l  qlwb /qlb/N/qlb/qlwb       hm >m lm tn*r hm lA y&mn wn xtm {llh ElY
2   w l hm E*Ab EZym w mn {l nAs mn yqwl Am g$wp /g$wp/N/g$wp/g$wp       {llh ElY qlwb hm w ElY smE hm w ElY >bSr hm
2   hm w mA y$Er wn fY qlwb hm mrD f zAd hm    >nfs /nfs/N/nfs/>nfs       {l*yn Amn wA w mA yxdE wn <seg id="182"<lA
2 hm mrD f zAd hm {llh mrDA w l hm E*Ab >lym   qlwb /qlb/N/qlb/qlwb       <seg id="182"<lA >nfs hm w mA y$Er wn fY
2    qAl wA <seg id="222"<n mA nHn mSlHwn >lA  >rD />rD/N/>rd/>rd         <seg id="210"<*A qyl l hm lA tfsd wA fY {l
2        b {l hdY f mA rbHt tjrt hm w mA kAn wA Dllp /Dllp/N/dllp/dllp     fY Tgyn hm yEmh wn >wl}k {l*yn {$tr wA {l
2  tjrt hm w mA kAn wA mhtdyn mvl hm k mvl      rbHt /rbHt/V/rbht/rbht     >wl}k {l*yn {$tr wA {l Dllp b {l hdY f mA
2  hm w mA kAn wA mhtdyn mvl hm k mvl {l*Y     tjrt /tjrp/N/tjrp/tjrt     {l*yn {$tr wA {l Dllp b {l hdY f mA rbHt
2      f lmA >DAt mA Hwl h *hb {llh b nwr hm w nArA /nAr/N/nar/nara       mA kAn wA mhtdyn mvl hm k mvl {l*Y {stwqd
2       mA Hwl h *hb {llh b nwr hm w trk hm fY >DAt />DA/V/>da/>dat       mhtdyn mvl hm k mvl {l*Y {stwqd nArA f lmA
2  lA ybSr wn Sm bkm EmY f hm lA yrjE wn >w    Zlmt /Zlmt/N/zlmt/zlmt     >DAt mA Hwl h *hb {llh b nwr hm w trk hm fY
2     fy h Zlmt w rEd w brq yjEl wn >SbE hm fY smA /smA/N/sma/sma         Sm bkm EmY f hm lA yrjE wn >w k Syb mn {l
2      w rEd w brq yjEl wn >SbE hm fY A*An hm  Zlmt /Zlmt/N/zlmt/zlmt     f hm lA yrjE wn >w k Syb mn {l smA fy h
2      hm mn {l SwEq H*r {l mwt w {llh mHyT b  A*An />*n/N/>*n/a*an       fy h Zlmt w rEd w brq yjEl wn >SbE hm fY
2         H*r {l mwt w {llh mHyT b {l kfryn ykAd SwEq /SAEqp/N/saeqp/sweq rEd w brq yjEl wn >SbE hm fY A*An hm mn {l
2      fr$A w {l smA bnA w >nzl mn {l smA mA f >rD />rD/N/>rd/>rd         mn qbl km lEl km ttq wn {l*Y jEl l km {l
2     bnA w >nzl mn {l smA mA f >xrj b h mn {l smA /smA/N/sma/sma         km ttq wn {l*Y jEl l km {l >rD fr$A w {l
```

Choose English Qurany Concepts



● first level Attribute: word Ignore case □ Position: Node

○ second level Attribute: word Ignore case □ Position: Node

○ third level Attribute: word Ignore case □ Position: Node

○ fourth level Attribute: word Ignore case □ Position: Node

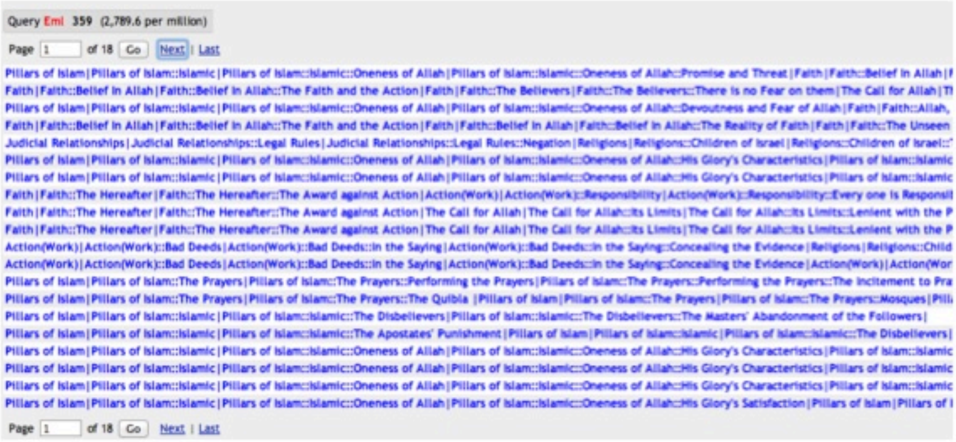
Text Type frequency distribution

Frequency limit: 0

Include categories with no hits: □

English Qurany Concepts Tree

Results are shown in Tree structure



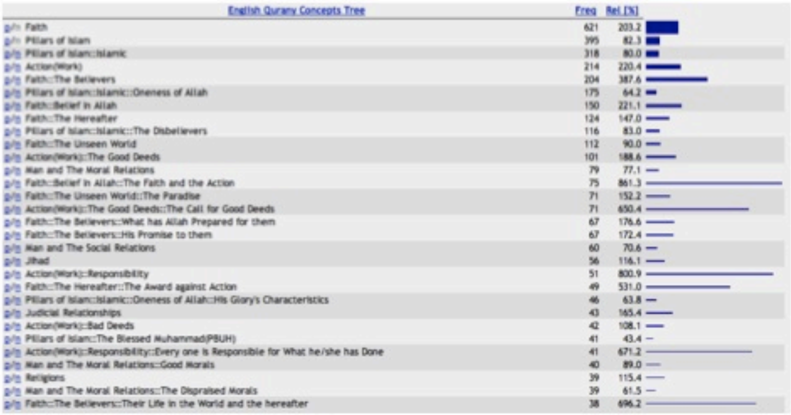
Query Eml 359 (2,789.6 per million)

Page 1 of 18 Go Next | Last

```
Pillars of Islam|Pillars of Islam::Islamic|Pillars of Islam::Islamic::Oneness of Allah|Pillars of Islam::Islamic::Oneness of Allah::Promise and Threat|Faith|Faith::Belief in Allah|F
Faith|Faith::Belief in Allah|Faith::Belief in Allah::The Faith and the Action|Faith|Faith::The Believers|Faith::The Believers::There is no Fear on them|The Call for Allah|Th
Pillars of Islam|Pillars of Islam::Islamic|Pillars of Islam::Islamic::Oneness of Allah|Pillars of Islam::Islamic::Oneness of Allah::Devoutness and Fear of Allah|Faith|Faith::Allah,
Faith|Faith::Belief in Allah|Faith::Belief in Allah::The Faith and the Action|Faith|Faith::Belief in Allah|Faith::Belief in Allah::The Reality of Faith|Faith|Faith::The Unseen
Judicial Relationships|Judicial Relationships::Legal Rules|Judicial Relationships::Legal Rules::Negation|Religions|Religions::Children of Israel|Religions::Children of Israel::
Pillars of Islam|Pillars of Islam::Islamic|Pillars of Islam::Islamic::Oneness of Allah|Pillars of Islam::Islamic::Oneness of Allah::His Glory's Characteristics|Pillars of Islam::Islamic
Pillars of Islam|Pillars of Islam::Islamic|Pillars of Islam::Islamic::Oneness of Allah|Pillars of Islam::Islamic::Oneness of Allah::His Glory's Characteristics|Pillars of Islam::Islamic
Faith|Faith::The Hereafter|Faith::The Hereafter::The Award against Action|Action(Work)|Action(Work)::Responsibility|Action(Work)::Responsibility::Every one is Responsi
Faith|Faith::The Hereafter|Faith::The Hereafter::The Award against Action|The Call for Allah|The Call for Allah::Its Limits|The Call for Allah::Its Limits::Lenient with the P
Faith|Faith::The Hereafter|Faith::The Hereafter::The Award against Action|The Call for Allah|The Call for Allah::Its Limits|The Call for Allah::Its Limits::Lenient with the P
Action(Work)|Action(Work)::Bad Deeds|Action(Work)::Bad Deeds::In the Saying|Action(Work)::Bad Deeds::In the Saying::Concealing the Evidence|Religions|Religions::Child
Action(Work)|Action(Work)::Bad Deeds|Action(Work)::Bad Deeds::In the Saying|Action(Work)::Bad Deeds::In the Saying::Concealing the Evidence|Action(Work)|Action(Wor
Pillars of Islam|Pillars of Islam::The Prayers|Pillars of Islam::The Prayers::Performing the Prayers|Pillars of Islam::The Prayers::Performing the Prayers::The Incitement to Pra
Pillars of Islam|Pillars of Islam::The Prayers|Pillars of Islam::The Prayers::The Quibla |Pillars of Islam|Pillars of Islam::The Prayers|Pillars of Islam::The Prayers::Mosques|Pill
Pillars of Islam|Pillars of Islam::Islamic|Pillars of Islam::Islamic::The Disbelievers|Pillars of Islam::Islamic::The Disbelievers::The Masters' Abandonment of the Followers|
Pillars of Islam|Pillars of Islam::Islamic|Pillars of Islam::Islamic::The Apostates' Punishment|Pillars of Islam|Pillars of Islam::Islamic|Pillars of Islam::Islamic::The Disbelievers|
Pillars of Islam|Pillars of Islam::Islamic|Pillars of Islam::Islamic::Oneness of Allah|Pillars of Islam::Islamic::Oneness of Allah::His Glory's Characteristics|Pillars of Islam::Islamic
Pillars of Islam|Pillars of Islam::Islamic|Pillars of Islam::Islamic::Oneness of Allah|Pillars of Islam::Islamic::Oneness of Allah::His Glory's Characteristics|Pillars of Islam::Islamic
Pillars of Islam|Pillars of Islam::Islamic|Pillars of Islam::Islamic::Oneness of Allah|Pillars of Islam::Islamic::Oneness of Allah::His Glory's Characteristics|Pillars of Islam::Islamic
Pillars of Islam|Pillars of Islam::Islamic|Pillars of Islam::Islamic::Oneness of Allah|Pillars of Islam::Islamic::Oneness of Allah::His Glory's Satisfaction|Pillars of Islam|Pillars of I
```

Page 1 of 18 Go Next | Last

To classify the results based on the verse concepts

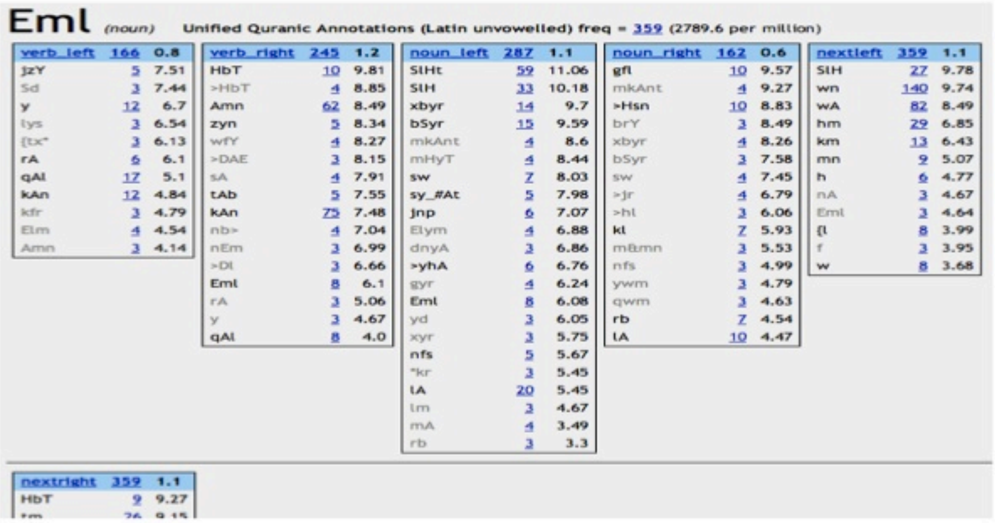


English Qurany Concepts Tree	Freq	Rel [%]
Faith	621	203.2
Pillars of Islam	395	82.3
Pillars of Islam::Islamic	318	80.0
Action(Work)	214	220.4
Faith::The Believers	204	387.6
Pillars of Islam::Islamic::Oneness of Allah	175	64.2
Faith::Belief in Allah	150	221.1
Faith::The Hereafter	124	147.0
Pillars of Islam::Islamic::The Disbelievers	116	83.0
Faith::The Unseen World	112	90.0
Action(Work)::The Good Deeds	101	188.6
Man and The Moral Relations	79	77.1
Faith::Belief in Allah::The Faith and the Action	75	861.3
Faith::The Unseen World::The Paradise	71	152.2
Action(Work)::The Good Deeds::The Call for Good Deeds	71	650.4
Faith::The Believers::What has Allah Prepared for them	67	176.6
Faith::The Believers::His Promise to them	67	172.4
Man and The Social Relations	60	70.6
Jihad	56	116.1
Action(Work)::Responsibility	51	800.9
Faith::The Hereafter::The Award against Action	49	531.0
Pillars of Islam::Islamic::Oneness of Allah::His Glory's Characteristics	46	63.8
Judicial Relationships	43	165.4
Action(Work)::Bad Deeds	42	108.1
Pillars of Islam::The Blessed Muhammad(PBUH)	41	43.4
Action(Work)::Responsibility::Every one is Responsible for What he/she has Done	41	671.2
Man and The Moral Relations::Good Morals	40	89.0
Religions	39	115.4
Man and The Moral Relations::The Dispraised Morals	39	61.5
Faith::The Believers::Their Life in the World and the hereafter	38	696.2

Word Eml has been mentiond in Quran 71 times in Good deed concepts

Word Sketch

Word Sketch is to show the word's grammatical and collocational behaviour



Eml (noun) Unified Quranic Annotations (Latin unvowelled) freq = 359 (2789.6 per million)

verb_left	166	0.8
jzY	5	7.51
Sd	3	7.44
y	12	6.7
lys	3	6.54
{tx*	3	6.13
rA	6	6.1
qAl	17	5.1
kAn	12	4.84
kfr	3	4.79
Elm	4	4.54
Amn	3	4.14

verb_right	245	1.2
HbT	10	9.81
>HbT	4	8.85
Amn	62	8.49
zyn	5	8.34
wfY	4	8.27
>DAE	3	8.15
sA	4	7.91
tAb	5	7.55
kAn	75	7.48
nb>	4	7.04
nEm	3	6.99
>Dl	3	6.66
Eml	8	6.1
rA	3	5.06
y	3	4.67
qAl	8	4.0

noun_left	287	1.1
SlHt	59	11.06
SlH	33	10.18
xbyr	14	9.7
bSyr	15	9.59
mkAnt	4	8.6
mHyT	4	8.44
sw	7	8.03
sy_#At	5	7.98
jnp	6	7.07
Elym	4	6.88
dnyA	3	6.86
>yhA	6	6.76
gyr	4	6.24
Eml	8	6.08
yd	3	6.05
xyr	3	5.75
nfs	5	5.67
*kr	3	5.45
lA	20	5.45
lm	3	4.67
mA	4	3.49
rb	3	3.3

noun_right	162	0.6
gfl	10	9.57
mkAnt	4	9.27
>Hsn	10	8.83
brY	3	8.49
xbyr	4	8.26
bSyr	3	7.58
sw	4	7.45
>jr	4	6.79
>hl	3	6.06
kl	7	5.93
m&mn	3	5.53
nfs	3	4.99
ywm	3	4.79
qwm	3	4.63
rb	7	4.54
lA	10	4.47

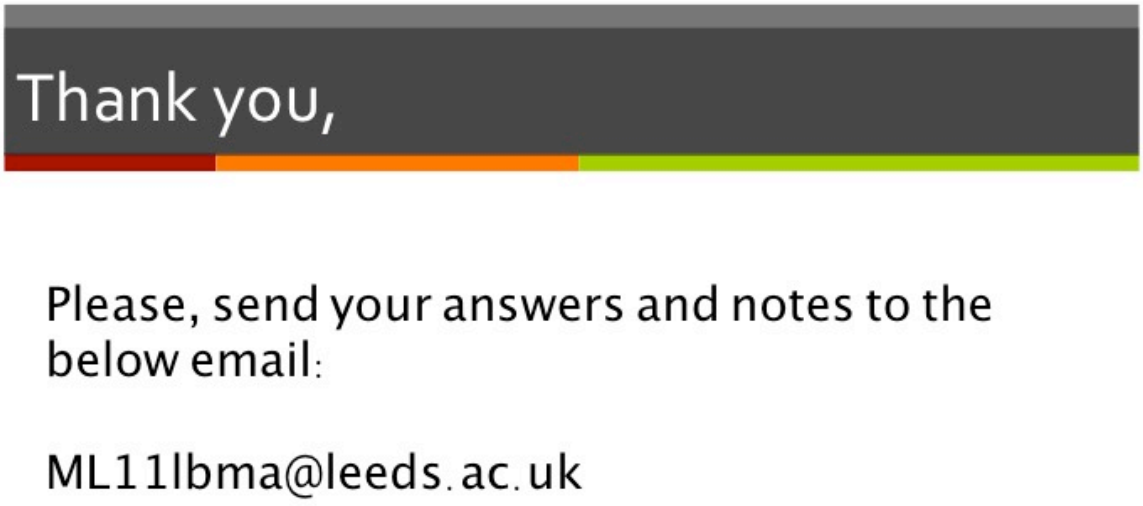
nextleft	359	1.1
SlH	27	9.78
wn	140	9.74
wA	82	8.49
hm	29	6.85
km	13	6.43
mn	9	5.07
h	6	4.77
nA	3	4.67
Eml	3	4.64
{l	8	3.99
f	3	3.95
w	8	3.68

nextright	359	1.1
HbT	9	9.27
tm	26	9.15

Your evaluation

Kindly, I would like to know How you used the system..

1_ Which words have you searched for?
2_ Which type of analysis have you used in your search(Postag, syntactic or semantic)?
3_During your search, have you found any mistakes in the results?
4_ Any suggestions to develop the system?



Thank you,

Please, send your answers and notes to the below email:

ML11lbma@leeds.ac.uk