Lexicon Digitization - A Framework for Structuring, Normalizing and Cleaning Lexical Entries

Hamzeh Amayreh Mohammad Dwaikat Mustafa Jarrar[1]

Birzeit University, Palestine

ABSTRACT

We present a parsing framework that we developed for digitizing about 150 Arabic-multilingual lexicons and storing them into one lexicographic database – available at Birzeit University. The framework consists of 30 parsers for (re)structuring, normalizing, and cleaning lexical entries. This parsing framework can be beneficial in resolving issues faced when hard-copy lexicons are to be digitized and represented in a machine-processable format. Each parser in the framework takes a lexical entry as input and (i) detects a candidate problem(s) to be resolved, (ii) assigns a category label to this problem, and then (iii) generates a suggested correction. The output might then be given to a linguist (if needed) to review and confirm manually.

The parsers were built to handle a broad set of delicate issues in both Arabic and English lexical entries of various types of lexicons. Such issues emerged because lexicons are originally designed to be printed and used as hard copies rather than stored in a machine-processable and understandable form. Symbols and characters in lexical entries might be used to indicate various cases, which is even more delicate when the same symbol is used differently within the same or across lexicons.

KEYWORDS

Lexicon, Dictionary, Arabic, Lexicon Digitization, Lexical entries, Lexicography, Linguistic Resources, Arabic NLP.

1 Introduction and Motivation

Lexicons and language resources are no more limited to hard copies and manual use. They are becoming important components in natural language processing and for building smart applications. Although there are many available dictionaries for most languages, limited number of Arabic lexicons are available in digital formats (Jarrar and Amayreh, 2019). This is not only because of the poor Arabic OCR technologies to recognize hard-copies into machine-readable text but also because converting this text into a machine-understandable (i.e., structured) format is indeed challenging. Furthermore, lexicons may come of different content and different structures; thus, automatic digitization tools are also difficult to develop. Lexicons might be grouped into the following five types, based on their structures and content:

(i) ***Dictionary***, which typically consists of a list of lexical entries and their translation in another language(s).

(ii) ***Glossary***, which typically consists of a set of lexical entries, each with a gloss to define the meaning of this lexical entry. Some glossaries include other information such as synonyms, translation(s), abbreviations, and references (i.e., relations) to other lexical entries.

(iii) ***Linguistic lexicon***: lexical entries with their senses (i.e., meanings), and maybe linguistic features, some inflections, and derivations.

(iv) ***Thesaurus***: sets of synonymous lexical entries.

(v) ***Semantic-variations lexicon***: pairs of lexical entries and the semantic differences between each of these lexical entries.

(vi) ***Morphological database/lexicon***: set of lexical entries (i.e., lemmas), each with some inflections and morphological features.

This report[2] presents the parsing framework that we developed at Birzeit University while digitizing about 150 Arabic-multilingual lexicons and storing them in one database (Jarrar and Amayreh, 2019; Jarrar, 2018). This lexicographic database is accessible through: (*i*) an online linguistic search engine[3] (Jarrar, 2020; Alhafi

[1] Corresponding author (mjarrar@birzeit.edu)

[2] **Published as:** Amayreh, H., Dwaikat, M., & Jarrar, M. (2019): A Framework Digitization Arabic Lexicons – Structuring, Normalizing and Cleaning Lexical Entries. Technical Report. Birzeit University, Palestine.

[3] The Lexicographic Search Engine (https://ontology.birzeit.edu)

et al., 2019), (*ii*) a set of RESTful web services[4], and (*iii*) an RDF representation using the W3C Lemon Model (Jarrar et al., 2019). All lexical concepts in all lexicons are being linked with concepts in the Arabic Ontology (Jarrar, 2021; Jarrar, 2011). Additionally, all lexical entries (i.e., lemmas) across lexicons are also being mapped with each other (Jarrar et al., 2018) and with dialectal lemmas (Jarrar et al., 2017; Jarrar et al., 2014).

During the process of digitizing 150 Arabic multilingual lexicons, we faced a considerable set of challenges. Not only that each lexicon follows a different structure, but we also found that the same lexicon is not always consistent with the way it is supposed or claimed to follow.

We did not use any OCR technology because of their very low accuracy. Instead, and as will be discussed in section 3, lexicons were manually typed (into MS Word) and then parsed and converted into the two temples depicted in Figure 1, then mapped to database tables. This report focuses only on the parsing of the lexical entries before converting them into a database, which is the most challenging task in lexicon digitization.

Our goal is to automate the digitization process as much as possible by detecting and correcting errors, then give the results to humans to validate and confirm when needed. We designed a parsing framework consisting of 30 parsers[5] designed to collectively handle Arabic and English lexical entries and ensure their correctness. Each parser was designed to detect and correct a specific issue.

Because lexicons are language references, they are assumed to be free of errors, thus should be correctly parsed and should not contain any mistake. To meet this strict requirement, each parser in our framework assigns a category label to each detected or corrected issue. This is very helpful for humans to review later and confirm each category of corrections.

The digitizing phases of the 150 lexicons are overviewed in (Jarrar and Amayreh, 2019), in which we also presented the linguistic search engine that we developed to allow people to search the lexicons online. This article presents the parsing framework and focuses only on the normalization and cleaning issues.

In the rest of this report, section 2 overviews the related work, and in section 3, we present the parsing framework. Section 4 presents the 30 parsers.

2 Related Work

Research on digitizing Arabic lexicons is limited. In what follows we review the most important works.

An electronic lexicon, called Al-Madar (Khemakhem et al., 2016), was constructed based on a printed copy of the Al-Ghani Lexicon and then represented using the ISO LMF standard. Most of the tasks in constructing Al-Madar were done manually through a web interface that was developed specifically for this lexicon.

A Medieval Arabic lexicon, called (*'al-qa mu s al-muhi t*), was digitized by Nahli et al. (2016) and then represented using the ISO LMF standard and using the Lemon model (Khalfi et al, 2016). This lexicon was originally in a plan-text format and was structured and normalized through several processing steps based on patterns for markers that are found in the text.

Five Hadith lexicons that were digitized by Soudani et al. (2015) and represented using the ISO LMF standard. The digitization of these lexicons was more sophisticated as they were digitized through several structuring and normalization phases. The structuring phase includes the identification of markers and blacks of linguistic information. The normalization phase mains to map the extracted blocks into linguistic categories, mainly LMF classes and attributes.

As lexicons typically are of different types and different structures that serve various purposes, it is difficult to generalize or reuse a digitization methodology for other lexicons.

Instead, we propose a parsing framework enriched with 30 parsers. The framework, as shall be explained in the next section, assumes a lexicon to be structured into Lexical Entry and Lexical Concept templates. Then, the normalization of lexicographic information elements can be semiautomated using appropriate parsers, given the specific needs for each lexicon. We do not claim that our proposed framework is suitable for digitizing any type of lexicons, but it was used for digitizing our 150 Arabic-multilingual lexicons, which were of different types.

3 The Parsing Framework

During our work on the digitization of 150 Arabic multilingual-lexicons, we have come across many issues for which we developed a parsing framework to resolve semi-automatically. Most of these lexicons were first manually typed (in MS Word) as they were only available in hard copies. Other lexicons that we

[4] The LexAPI page (https://ontology.birzeit.edu/lexapi) for retrieving synonyms, translations, definitions, ontology concepts, morphological features, and others.

[5] All parsers and other Arabic NLP tools can be downloaded from (https://ontology.birzeit.edu/tools/)

found in digital textual formats, were parsed semi-automatically. In case a lexicon uses regular markers (e.g., comma, semicolon, tab, new line) to separate different linguistic features, then it was converted automatically into a table (in MS Word), which we designed for each lexicon; otherwise, such markers were manually added to the text before it is automatically converted. Each resultant table was then parsed and restructured into a normalized model using two general templates (see Figure 1). Each lexicon was parsed and mapped to these templates by: First, extracting lexical entries and their linguistic features and storing them in the *Lexical Entry* template. Second, by extracting definitions and their relations and storing them in the *Lexical Concept* template.

We faced many challenges that were very difficult to resolve in a fully automatic manner. Therefore, we have built a parsing framework designed to handle this process semi-automatically with minimum human involvement. Our parsers detect and filter out each issue individually, then assign a category label to each of these issues to indicate its nature. Some issues might be assigned several labels as there might be multiple parsers applicable on it. The output of the parsers also includes suggested corrections depending on the nature of the issue. Each labeled issue (i.e., category) can then be given to a linguist to review and confirm the suggested corrections. Complex cases that parsers detect but cannot correct were processed manually by the linguist.

Our framework cannot be used to handle all issues for all types of lexicons, as each lexicon is of different purposes, and each has its own structure and challenges. Nevertheless, the framework was used for digitizing 150 Arabic lexicons.

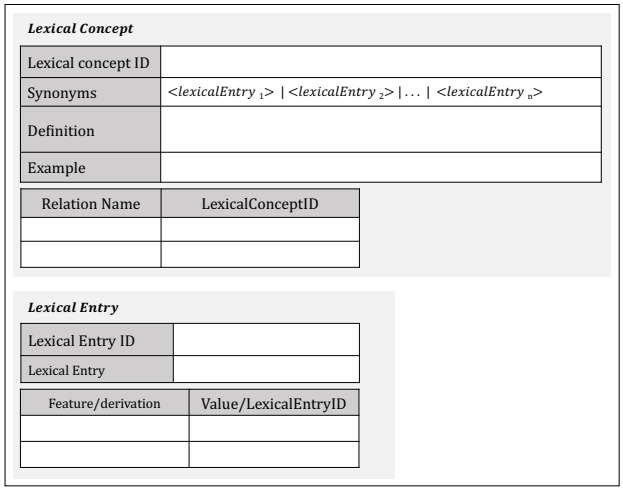

Figure 1: Lexical Concept and Lexical Entry templates

4 Set of Parsers

This section presents our set of parsers. Each parser is designed to (1) detect a certain issue (a candidate problem in a lexical entry), (2) give a label to this issue, and (3) suggest a correction. The input for each parser is a set of lexical entries, and the output is the label and the suggested correction. All parsers, their category labels, and suggested corrections are summarized in Table 1.

4.1 Comma and Semi-colon

This parser is designed to handle lexical entries that include Arabic and English comma(s) and/or semi-colon(s) {“,”, “;”, “؛”, “،”}, which are common in most dictionaries. It is found to most likely mean synonymy. The parser detects commas and replaces them with a chosen delimiter of synonymy (“|” in our case). Then it gives them the label (`Com`) for later review and/or approval.

4.2 Parenthesis

Parenthesis are used in dictionaries for purposes that differ between different dictionaries and within the same dictionary. A special type of their use is to re-arrange words in a lexical entry in a way that maintains an alphabetical ordering, such as: “accelerator (linear...)”, “affinity (chemical)”, “drawing (final)”, “earth (the)”, and “crush (to)”. Most of these cases are identified when there is a single word between the parenthesis that is either an adjective, an adverb, or the word “the”. The parser then deletes the parenthesis and moves the word between it to the beginning of the lexical entry. However, in cases where the text between the parenthesis is “to”, both the parenthesis and the word “to” are removed and the lexical entry is given a POS feature of the value “verb”. These cases are given the label (`P0`), and are flagged for manual inspection of the suggested correction.

4.3 Parenthesis pairs

The use of different types of parenthesis (i.e., (), {}, <>, or []) are found in many dictionaries. Their use however, have variant purposes that cannot be generalized into a rule to apply when such cases are met during parsing. Therefore, these cases were only given the label (`P6`) by the parser, and they were not given a suggested correction. Yet, they were flagged for the reviewer to manually check them.

4.4 Plural sign

Some lexical entries include the plural form between parenthesis. For regular plurals, the text “(s)” is appended to the end of the lexical entry as in “border(s)”. In this case the parser removes the plural sign (i.e., (s)) and copies the same lexical entry with an “s”

letter appended to its end as the plural feature of the word. However, in case of irregular plurals such as: “appendix (pl. appendices)”, the plural sign (i.e., pl.) and the parenthesis are removed and the plural form of the lexical entry is added as its plural feature. Cases of the plural sign are labeled as (`P1`) by the parser and are flagged for manual inspection.

4.5 Singular sign

The singular form of a lexical entry is often included in dictionaries, as in “arteriolae (sing. arteriola)”. Cases as such are detected by our parser, and are given the label (`P2`), they are also given a suggested correction which extracts the singular form (“arteriola” in this case) as the singular feature of the lexical entry. The parser then removes the singular sign (i.e., sing.) and the parenthesis. These cases are also flagged to be manually reviewed later.

4.6 POS sign

Some dictionaries denote that a lexical entry is a verb by appending (to) -sometimes (to…)- to the lexical entry. Adjectives are denoted by the sign (adj.) and nouns by the sign (n.). For such cases, the parser removes the POS sign from the lexical entry and map the sign to its corresponding value as the POS feature of the lexical entry. These cases are given the label (`P3`) by this parser.

4.7 Origin

Dictionaries sometimes include the origin of a lexical entry. The origin is denoted by the abbreviation of its original language. Examples of such include: (“(It.)”, “(Fr.)”, “(Hun.)”, “(Sp.)”, “(Lat.)”, “(Ger.)”, “(n.)”, “(Por.)”, “(Ar.)”, “(Ind.)”, “(Ice.)”, “(Swe.)”, “(Nor.)”, “(Isl.)”, “(Rus.)”, “(Pl.)”, “(L.)”, “(Esk.)”, “(Pol.)”, “(Fin.)”, “(Jap.)”, “(Tur.)”). In such cases, the lexical entry is copied to the appropriate language column, and the origin sign is removed from both columns. The parser also gives these cases the label (`Orig`) and adds the origin feature to the lexical entry.

4.8 Hyphen

Dictionaries use different forms of hyphens (i.e., “-”, “–”) for different purposes. All forms are converted to the standard hyphen form “-” before the parsing step which handles different cases as such: (1) When the hyphen appears immediately after the second word in a two-word lexical entry (e.g., “rotation curl-”), the parser removes the hyphen and moves the second word before the first word of the lexical entry (e.g., “curl rotation”). (2) When the hyphen is preceded with space and appears immediately before the second word (e.g., “sound -stone”) the parser removes the hyphen (e.g., “sound stone”). (3) When the hyphen is surrounded by spaces (e.g., “radiance - radiant intensity per unit area (at a point of a surface; in a given direction)”), it is most likely used to separate between the lexical entry and its explanation. For this case, the parser removes the hyphen and moves the explanation to the gloss of the lexical entry. (4) Finally, when the hyphen is used -with no spaces around it- as a separator between multiword phrases (e.g., “semi-tone”), it is kept as it is. All cases where a hyphen exists are given the flag (`H`) by the parser and are flagged for manual review.

4.9 Special Symbols

Sometimes dictionaries include one of these symbols (!, @, #, $, %, ⁒, &, ~, _) in their lexical entries. Since this is not a prominent issue and since symbols might be valid in certain lexical entries, and due to the difficulty of parsing such a case; symbols as such were kept intact except for the “!” which the parser removes. However, all cases of symbols were given the flag (`Sym`) for later manual inspection.

4.10 Arithmetic symbols

A lexical entry in a dictionary might contain one of these arithmetic symbols (+, *, ×, ÷, /, ^, =). Such arithmetic symbols might be part of the lexical entry indeed, but often they are used as markers to indicate certain issues. Our parser detects them all and label them with (`Arth`) for manual treatment, but it provides suggested corrections only in two cases. One of which is related to the “=” symbol that is often used to separate between synonyms as in “ear crystals (= statolith)” which was corrected to “ear crystals | statolith” by removing the parenthesis and replacing the “=” sign with “|” (the delimiter we use to separate between synonyms). The other case is related to the “/” symbol which is sometimes used also to separate between synonyms as in “bank examiner/ inspector” which was dealt with by replacing the “/” with the delimiter “|” and copying the text before “/” symbol excluding the last word just after the added delimiter “|” and before the first word that came immediately after the “/” symbol.

4.11 Quotation marks

Quotation marks (", ') were all removed by the parser except for these two cases: (1) when the (") is used as the inch measure unit as in “oil well cartridge 4" standard” which is detected by checking for a missing (") pair and by checking the immediate character before the (") to be a number. In this case the (") was kept. (2) when the (') is used before the possessives s as in “Grimm's law” which was kept as it is. All cases of quotation marks including the ones that were not changed were all given the label (`Q`) by our parser.

4.12 Punctuation

Cases in which punctuation marks (`, ?, :, ., …, ‘, ’, ؟) were used in lexical entries were flagged for manual inspection, except for the comma “,” which is dealt with by another parser/rule, and the colon “:” which is replaced by our synonyms delimiter “|”. These cases were all given the label (Pun) by the parser. In addition, the dot “.” punctuation mark is removed if it comes at the end of a lexical entry.

4.13 Abbreviation

This parser detects cases in which the dot “.” is used to separate between letters of an abbreviation as in “adenosine triphosphate (A.T.P)”. The parser extracts the abbreviations into a new abbreviation feature and label such cases with (Abbr) for manual review.

4.14 Digits

Digits (0-9, ٠-٩) are often found in lexical entries where it is there either to denote listing of different sneses for a lexical entry as in “Bill 1 - Billet de banque”, and “Bill 2 - facture” or as a valid part of a lexical entry as in “solubility in CCL4” or as a result of an error as in “1odging” (notice the first character is 1 not L). There was a difficulty in determining which one is the case, therefore the parser removes all numbers except for lexical entries that are entirely numbers, and give all cases of numbers the flag (Num) so that a reviewer can later approve the change.

4.15 Non-English characters in English entries

When there are non-English characters in an English lexical entry, our parser detects this case and checks whether the character is Latin or not. If Latin, it keeps it. Otherwise, the parser deletes the character. The parser also gives such cases the label (NE) for later manual review.

4.16 Non-French characters in French entries

Similarly, for non-French characters in a French lexical entry, our parser detects these cases and deletes the characters only if they are non-Latin. The parser also gives them the label (NF) for later manual review.

4.17 Non-Arabic characters in Arabic entries

Non-Arabic characters in an Arabic lexical entry are also detected and given the label (NA) by our parser for them to be manually reviewed later. No suggested correction is provided in this case.

4.18 Arabic word starts with final-form Alif

As a result of an error, sometimes Arabic words in dictionaries start with the Alif letter in its final form (ى) which is syntactically wrong in Arabic. However, it is very hard to determine whether this letter should be removed or replaced with one of the beginning-forms of the Alif. Therefore, our parser detects such words, deletes the first letter of the words (final-form Alif), and give them the flag (SA) for later inspection by a linguist.

4.19 Arabic word starts with a diacritic

Often times Arabic words in Arabic lexical entries start with a diacritic character rather than a letter (e.g., “◌َحَدَّث”). This is a result of an error as it is not syntactically correct in Arabic. Our parser removes diacritics at the beginning of an Arabic word, and labels these cases with (D0) for later review. Hence, “◌َحَدَّث” is corrected to “حَدَّث” by the parser.

4.20 non-terminal Arabic letter has its diacritic as Tanween

Another syntactically wrong case in Arabic is when a Tanween diacritic (◌ً, ◌ٌ, ◌ٍ) appears as the diacritic for a non-terminal letter in Arabic lexical entries. Such cases are dealt with by removing the Tanween diacritic and flagging the case as (T1) for later review.

4.21 Arabic Maddah character

The Arabic Tatweel character (ـ) is used in Arabic for lengthening words to justify them (e.g., “سمـــــاء”). This is non-relevant to the syntax and the semantics of the word. Therefore, our parser deletes this character from Arabic lexical entries and give them the flag (Mad) for later review. Hence, “سمـــــاء” is corrected to “سماء”.

4.22 Arabic word starts with Ta Marbuta

An Arabic word that starts with the letter Ta Marbuta (ة) is considered syntactically wrong. Our parser detects such cases, removes the letter Ta Marbuta, and give them the flag (T2) for the reviewer to determine whether it should be replaced or removed completely.

4.23 Inconsistent diacritics on the same Arabic letter

Another syntax error that might occur in Arabic lexical entries is when there are multiple inconsistent diacritics on the same letter, for example: in the word (فَعِل) the first letter (ف) has two diacritics, the Fatha (◌َ) and the Kasra (◌ِ) which is not correct. Our parser removes all inconsistent diacritics and give each case the label (D1) for later correction by a linguist whom will decide what diacritic(s) to keep and what to delete.

4.24 Arabic AL (ال)

To maintain an alphabetical ordering, some dictionaries remove the Arabic AL (ال) ("The" in English) from the beginning of Arabic words in lexical entries and put it between parenthesis after the word as in "كون (ال...)". Our parser detects such cases and deletes the (ال...). The parser also gives such cases the label (AL) for the reviewer to approve the suggested correction.

However, other cases might need further correction, such as: "حركيات (ال...) الكهربائية", which the parser initially corrects to: "حركيات الكهربائية". This might be syntactically incorrect in some cases, since the second word (right-to-left) starts with an AL, while the AL of the first word was removed by the parser. Therefore, for each case of AL (ال...), the parser removes AL (ال) from the beginning of all successive words in the lexical entry. Hence, "حركيات (ال...) الكهربائية" is corrected to "حركيات كهربائية".

4.25 Character-set issues

Often characters with the same orthography have different encodings in the Unicode character set. Examples of such are the No-Break space character ( ) (U+00a0) compared to the regular space ( ) (U+0020). This problem is even more apparent in Arabic, for example the two Arabic letters Lam (ل) and Alif (ا) when the Lam precedes the Alif, it is written as (لا). Sometimes this (لا) is found represented as two character encoded as (U+0644U+0627), while other times, it is found as a single character encoded as (U+FEFB) depending on the Unicode version used when the dictionary was typed. Our parser detects such cases and corrects them. For example: The No-Break space character is replaced with the regular space character, and the (لا) encoded as (U+FEFB) is replaced with the (لا) encoded as (U+0644U+0627). Our parser also gives such cases the label (CS).

4.26 Lengthy multiword lexical entries

There are cases in which a lexical entry has many words such as "buildings or other structures recurrent taxes on land" which forms "poor" lexical entries. Our parser detects all cases with more than five words and give them the label (Long) so that a reviewer can later decide whether to make it shorter, consider it a definition, skip it, etc.

4.27 Multiple white spaces

This parser detects cases in which there are more than one consecutive whitespace character. The parser replaces these cases with a single space character and give them the label (space).

4.28 Sub-term synonymy

This parser detects the case in which a lexical entry -either from source or after parsing- has two synonyms; one of which is part of the other as in "internal condition | internal condition of a body". It is very likely that the second entry is an explanation rather than a synonym of the first entry. The parser in this case, removes the smaller synonym (e.g., "internal condition" in this case) and the synonymy delimiter "|". Finally, the parser gives such cases the label (STS) for later manual review.

4.29 Arabic related symbol

Arabic glosses in dictionaries use delimiter (ظ.) that is a shortcut for the word (انظر) (i.e., see in English) to refer the reader to another related lexical entry. This parser replaces all occurrences of (ظ.) with (انظر:) and adds the referenced lexical entry as a related feature to the lexical entry. The parser also gives these cases the label (ARS) for later review.

4.30 Arabic special symbols

Other cases were found apparent in Arabic lexical entries, such as: having angular brackets "< >" or "(ن)" which are removed and given the label (AOS) by our parser for it to be later checked manually.

4.31 Duplicate Arabic lexical entries

This parser detects duplicate Arabic lexical entries, that are exactly the same or didactically-compatible words (i.e., their Implication Direction metric is greater than or equal to zero) [5]. For example, the entries "فعَل" and "فعل" are considered didactically-compatible since they have an Implication Direction metric of zero, which means that each word implies the other, and hence can be treated as duplicates. The parser then. and for each entry, removes the diacritic of a letter if its corresponding letter on the other word has no diacritic (e.g., "فعَل" and "فعل" become "فعل" and "فعل"). The parser gives these cases the label (DAE) for later manual review.

Table 1. Summary of the parsers, category labels, and suggested corrections.

	Parser	**Example**	**Label**	**Suggested Correction**	
1	Comma	“austral, southern”	Com	“austral | southern”	
2	Parenthesis	“affinity (chemical)”	P0	chemical affinity	
3	Parenthesis pairs	“عدد >أفوجادرو”	P6	“عدد أفوجادرو”	
4	Plural sign	“border(s)” “appendix (pl. appendices)”	P1	“border” “appendix”	Plural
5	Singular sign	“arteriolae (sing. arteriola)”	P2	“arteriolae”	Singular
6	POS sign	“acidic (adj.)”	P3	“acidic”	POS
7	Origin	“andante(It.)”	Orig	“andante”	Origin
8	Hyphen	“rotation, curl-” “sound -stone”	H	“curl rotation” “sound stone”	
9	Special Symbols	“full ahead!” “newscar (TV & radio)”	Sym	“full ahead” “newscar (TV & radio)”	
10	Arithmetic symbols	“ear crystals (= statolith)” “bank examiner/ inspector”	Arth	“ear crystals | statolith” “bank examiner | bank inspector”	
11	Quotation marks	“huff and puff “process”” “oil well cartridge 4” standard” “Grimm's law”	Q	“huff and puff process” “oil well cartridge 4” standard” “Grimm's law”	
12	Punctuation	“R?entgen rays”	Pun	“Rentgen rays”	
13	Abbreviation	“adenosine triphosphate (A.T.P)”		“adenosine triphosphate”	Abbr.
14	Digits	“solubility in CCL4”	Num	“solubility in CCL”	
15	Non-English	“Cliché”	NE	None	
16	Non-French	“N° de compte”	NF	None	
17	Non-Arabic	“U واد بشكل”	NA	None	
18	Start with final-form Alif	“نموذج”	SA	“موذج”	
19	Start with a diacritic	“ حَدَّثَ◌”	D0	“حَدَّثَ”	

20	Non-terminal letter with Tanween	"تبنٍ"	T1	"تبنّ"	
21	Maddah	"سمـــــاء"	Mad	"سماء"	
22	Ta Marbuta	"تسمم ةكحولي"	T2	"تسمم كحولي"	
23	Inconsistent diacritics	"صِوَائَة"	D1	"صوَائَة"	
24	Arabic AL	"كون (ال...)" "حركيات (ال...) الكهربائية"	AL	"كون" "حركيات كهربائية"	
25	Charset	(see the text)	CS	(see the text)	
26	Lengthy lexical entries	"buildings or other structures recurrent taxes on land"	Long	None	
27	Multiple white space	"reach into"	space	"reach into"	
28	Sub-term synonymy	"internal condition of a body | internal condition"	STS	"internal condition of a body"	
29	Arabic related symbol	" سمة الصامت الذي يتميز بالتعطيش حين نطقه. ظ. تعطيش."	ARS	" سمة الصامت الذي يتميز بالتعطيش حين نطقه. انظر: تعطيش."	
30	Arabic special symbols	"تنشيط (ن)"	AOS	"تنشيط"	
31	Duplicate Arabic lexical entries	"فعَل" and "فَعل"	DAE	"فعل" and "فعل"	

REFERENCES

Aïda Khemakhem, Bilel Gargouri, Abdelmajid B. Hamadou, and Gil Francopoulo. 2016. ISO standard modeling of a large Arabic dictionary. Natural Language Engineering, 22(6), Pages 849-879.

Diana Alhafi, Anton Deik, Mustafa Jarrar: Usability Evaluation of Lexicographic e-Services. The 16th IEEE/ACS International Conference on Computer Systems and Applications (AICCSA). Pages(1-7). IEEE. Abu Dhabi, UAE. 2019

Mustapha Khalfi, Ouafae Nahli, and Arsalane Zarghili. 2016. Classical dictionary Al-Qamus in lemon. In Proceeding of the 4th IEEE International Colloquium on Information Science and Technology. IEEE, Morocco, 325-330.

Nadia Soudani, Ibrahim Bounhas, Bilel Elayeb, and Yahya Slimani. 2015. An LMF-based Normalization approach of Arabic Islamic dictionaries for Arabic Word Sense Disambiguation: application on hadith. International Journal on Islamic Applications in Computer Science and Technology, 3(2).

Ouafae Nahli, Francesca Frontini, Monica Monachini, Fahad Khan, Arsalane Zarghili, and Mustapha Khalfi. 2016. Al Qamus al Muhit, a Medieval Arabic Lexicon in LMF. In Proceeding of the LREC.

Mustafa Jarrar, Nizar Habash, Faeq Alrimawi, Diyam Akra, Nasser Zalmout: Curras: An Annotated Corpus for the Palestinian Arabic Dialect. Journal Language Resources and Evaluation. Pages(745-775). Volume(51), Issue(3). Springer (doi.org/10.1007/s10579-016-9370-7). 2017

Mustafa Jarrar: Search Engine for Arabic Lexicons. The 5th Conference on Translation and the Problematics of Cross-cultural Understanding. The Forum for Arab and International Relations. Doha, Qatar. December, 2018

Mustafa Jarrar, Fadi Zaraket, Rami Asia, and Hamzeh Amayreh. 2018. Diacritic-Based Matching of Arabic Words. ACM Trans. Asian Low-Resour. Lang. Inf. Process. 18, 2, Article 10 (December 2018), 21 pages.

Mustafa Jarrar, Hamzeh Amayreh: Linguistic Search Engine. In Proceedings of the International World Wide Web conference (WWW 2019). ACM, San Francisco, CA, USA.

Mustafa Jarrar: Building a Formal Arabic Ontology (Invited Paper). Proceedings of the Experts Meeting on Arabic Ontologies and Semantic Networks. ALECSO, Arab League. Tunisia. July, 2011

Mustafa Jarrar, Nizar Habash, Diyam Akra, Nasser Zalmout: Building a Corpus for Palestinian Arabic: a Preliminary Study. Arabic Natural Language Processing Workshop, at the Conference on Empirical Methods in Natural Language Processing (EMNLP 2014). Pages(18-27). Association for Computational Linguistics. ISBN:9781937284961. Qatar. October, 2014

Mustafa Jarrar, Hamzeh Amayreh, John P. McCrae: Representing Arabic Lexicons in Lemon - a Preliminary Study. The 2nd Conference on Language, Data and Knowledge (LDK 2019). Pages(29-33). CEUR, Volume 2402. ISSN:1613-0073. Leipzig, Germany. 2019

Mustafa Jarrar: Digitization of Arabic Lexicons. Arabic Language Status Report. UAE Ministry of Culture and Youth. Pages 214-2017. Dec 2020

Mustafa Jarrar: The Arabic Ontology - An Arabic Wordnet with Ontologically Clean Content. Applied Ontology Journal, 16:1, 1-26. IOS Press. 2021