A description and demonstration of SAFAR framework

Karim Bouzoubaa [1], Younes Jaafar [1], Driss Namly [1], Ridouane Tachicart [1], Rachida Tajmout [1], Hakima Khamar [2], Hamid Jaafar [3], Si Lhoussain Aouragh [4], Abdellah Yousfi [5]

[1] Mohammadia School of Engineers, Mohammed V University in Rabat, Morocco
[2] Faculty of Letters and Human Sciences, Mohammed Vth University, Rabat, Morocco
[3] Polidisciplinary faculty of Safi, Caddi Ayyad University, Morocco
[4] Faculty of Legal, Economic and Social Sciences - Sale, Mohammed V University in Rabat, Morocco.
[5] Faculty of Legal, Economic and Social Sciences - Souissi, Mohammed V University in Rabat, Morocco.
karim.bouzoubaa@emi.ac.ma; jayounes@yahoo.fr; namly_driss@yahoo.fr; tachicart@gmail.com; tajmoutrachida@yahoo.fr; khamarhaki@gmail.com; jaafarhamid1973@gmail.com; jaafarhamid1973@gmail.com; l.aouragh@um5r.ac.ma; yousfi240ma@yahoo.fr

Abstract

Several tools and resources have been developed to deal with Arabic NLP. However, a homogenous and flexible Arabic environment that gathers these components is rarely available. In this perspective, we introduce SAFAR which is a monolingual framework developed in accordance with software engineering requirements and dedicated to Arabic language, especially, the modern standard Arabic and Moroccan dialect. After one decade of integration and development, SAFAR possesses today more than 50 tools and resources that can be exploited either using its API or using its web interface.

1 Introduction

NLP infrastructures, referred also as NLP architectures, represent an efficient way for standardization, optimization of efforts, collaboration and acceleration of developments in the field of NLP. For the last decade, the NLP research community witnessed an extensive release of these infrastructures. Some become very famous such as GATE[1] or Stanford CoreNLP[2], while others existed only for a very short time. Some are multilingual while others are not, some are targeting multiple domains while others are not, etc.

However, it is known that only a few of them are dedicated to only one language such as AraNLP (Althobaiti et al. 2014) or "ITU Turkish Natural Language Processing Pipeline" (G. Eryiğit, 2014). On another hand, the literature shows that existing infrastructures are using randomly three different namings: "toolkit", "platform" and "Framework". From the Software Engineering (SE) perspective, these namings have different meanings. It is then necessary to first define them before presenting, categorizing, and benchmarking NLP infrastructures. Briefly speaking[3], a toolkit is a set of tools within a single box used for a particular purpose. A platform consists of several interoperable tools with a homogeneous structure but without providing any API to extend their components. A framework is a layered structure developed to be used as a support and guide to build NLP programs and tools.

In this work, we focus on the Arabic language infrastructures. We demonstrate that the "Software Architecture for ARabic" (SAFAR) framework[4] is one of the most interesting frameworks to consider when developing any Arabic NLP component.

The rest of this article is as follows. Section 2 presents SAFAR in terms of principles, architecture and standards. Section 3 describes SAFAR content. Section 4 is dedicated to SAFAR use and exploitation. Finally, in the last section, we conclude the paper.

2 SAFAR framework

2.1 Principles

In most cases, the development of Arabic NLP applications requires the use of several tools at once, each dealing with a certain level of language. Generally, these tools are heterogeneous and raise many SE problems such as interoperability, reusability, portability, etc. Moreover, researchers are usually in need not only of tools but also of Language Resources (LRs).

To overcome the above-mentioned SE issues and to suit the needs of the ANLP community in terms of processing Arabic effectively and providing reusable LRs, we developed SAFAR as a software

[1] https://gate.ac.uk

[2] https://stanfordnlp.github.io/CoreNLP/

[3] https://whatis.techtarget.com/

[4] http://arabic.emi.ac.ma/safar

Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations, pages 127–134
April 19 - 23, 2021. ©2021 Association for Computational Linguistics

architecture for Arabic with the following principles:

- Integrate not only tools and programs but also LRs;
- Structure the architecture to integrate two types of Arabic, namely MSA, and dialects;
- Respect the Arabic language features in the structure of the architecture;
- Develop tools or LRs when available ones are not satisfactory;
- Provide the architecture to be exploited not only by computer scientists but also by linguists;
- Involve in our team computer scientists, statisticians and linguists.

In general, our philosophy is not to develop ourselves all the NLP layers and modules, but to integrate existing ones consistently. Consequently, our approach consists in providing the specifications in terms of APIs for each module of our architecture and also providing (if any) implementations of these APIs with tools that have proved to be efficient and published under a free license such as GNU GPL, Apache or Non-Commercial Software. Indeed, the main challenge faced during this integration process is to develop bridges between different programming languages for tools and data structures for resources to use them in a single environment. However, when modules and LRs are not available, we develop them from scratch inside SAFAR. It is worth mentioning that after a certain threshold of maturity (for instance, it is the case of stemming as per the third release), it is useless to continue integrating every new implementation of a given level, with the flexibility that the framework is open enough to allow researchers to do it if needed.

2.2 Architecture

SAFAR is a Java-based framework dedicated to Arabic Natural Language Processing. As shown in Figure 1, SAFAR has several layers that provide services directly usable by other layers in accordance with the relationships modeled with arrows in the figure.

- Basic: designed to implement tools dealing with morphology, syntax and semantics;
- Tools: includes a set of technical services and pre-processing tools as well as machine and deep learning utilities;
- Resources: provides services for maintaining, consulting and managing Arabic language resources such as corpora, dictionaries and ontologies;
- Application: contains high-level applications such as sentiment analysis or Question/Answering systems;
- Client applications: interacts with all other layers to serve clients via web applications, web services, etc.

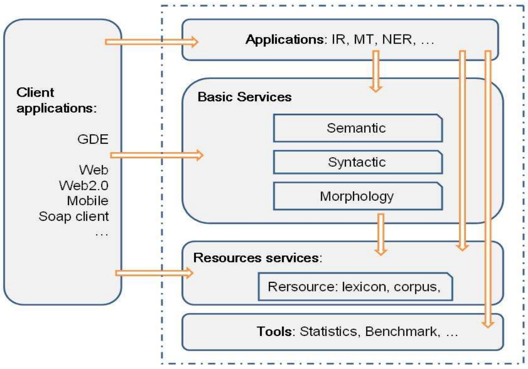


Figure 1: SAFAR framework general architecture.

2.3 Standards

Concerning the respect of international standards, and in order to facilitate their use in different contexts, we adopt the interoperability guides for all SAFAR components. Indeed, SAFAR tools input/output and LRs are formatted using the XML representation standard. In addition to the respect of representation standard, we use structuring standards such as Arab League Educational, Cultural and Scientific Organization (ALECSO)[5] recommendations for the design of Arabic morphological analyzers, Lexical Markup Framework (ISO 24613:2008) (LMF) for lexicons and Text Encoding Initiative (Lou Burnard et al. 2008) (TEI) for corpora.

3 SAFAR content

As previously explained, the structure of SAFAR is split into three main packages: MSA, Dialects and Machine learning models. Since Dialects are

[5] http://www.alecso.org/site/

numerous, we have been interested so far to integrate only the Moroccan dialect even if the architecture is flexible enough to embed any other dialects.

3.1 MSA

This package is the most populated one. Indeed, for almost two decades the research community spent all their efforts in developing components (tools and resources) for this type of Arabic.

Table 1 shows all the integrated tools for MSA[6]. These tools have been widely used by the ANLP community and it will be very advantageous to use them within a homogenous and flexible framework. Other tools have been developed from scratch such as “SAFAR stemmer”, “SAFAR POS tagger”, etc. Tools starting with “SAFAR” in the table have been developed from scratch by our research team for one of the following reasons 1) available tools return incorrect results, 2) there are no similar tools within the community, or 3) existing tools cannot be reused in several technical environments. In addition, the integration of multiple implementations for the same layer allows their benchmarking. Thus, we were able to make a detailed evaluation and/or comparison of stemmers (Jaafar and Bouzoubaa, 2016), morphological analyzers (Jaafar and Bouzoubaa, 2014) and parsers (Jaafar and Bouzoubaa, 2017).

The column “Per” indicates how many researchers have been involved in the development/integration of the corresponding tool. The "Vr" column indicates SAFAR version from which the tool is present.

Layer	Package	Implementation name	Reference	Per	Vr
App	key_words_extractor	SAFAR key_words_extractor		3	3
	stopwords_analyzer	SAFAR stopwords_analyzer		3	3
	moajam_moaassir	SAFAR moajam_moaassir		2	1
	moajam_tafaoli	SAFAR moajam_tafaoli		2	1
	Light summarization	SAFAR light_summarization		2	2
	morphosyntactic	SAFAR morphosyntactic_processor		2	1
	stem_counter	SAFAR stem_counter		2	1
	Syntax	Farasa parser	Zhang et al. 2015	2	2
		Stanford parser	Green and Manning 2010	2	3
		Farasa POS tagger	Zhang et al. 2015	2	1
		SAFAR POS tagger		3	3
	Morphology	Alkhalil analyzer	Boudlal, et al. 2010	2	2
		Alkhalil 2 analyzer	Boudchiche et al. 2017	2	2
		BAMA (Aramorph) analyzer	Buckwalter 2002	2	1
		MADAMIRA analyzer	Pasha, et al. 2014	2	1
		Farasa lemmatizer	Abdelali, et al. 2016	2	3
		SAFAR lemmatizer	Namly et al. 2020	3	3
		ISRI stemmer	Algasaier 2005	2	2
		Khoja stemmer	Khoja 2002	2	1
		Light10 stemmer	Larkey et al. 2007	2	1
		Motaz stemmer	Motaz and Ashour 2010	2	2
		Tashaphyne stemmer	Zerrouki 2012	2	2
		SAFAR stemmer	Jaafar et al. 2016	2	2
Util	StopWords	SAFAR StopWords remover		3	3
	Benchmark	SAFAR Analyzers benchmark	Jaafar and Bouzoubaa, 2014	2	2
		SAFAR Stemmers benchmark	Jaafar et al. 2016	2	2
		SAFAR Parsers benchmark	Jaafar and Bouzoubaa, 2017	2	2
	Normalization	SAFAR Normalizer		3	1
	Splitting	SAFARS sentence Splitter		2	1
	Tokenization	SAFAR Tokenizer		2	1
	Pattern detector	SAFAR Pattern detector		2	3
	Transliteration	SAFAR Transliterator		2	1

Table 1: MSA tools implemented in SAFAR

[6] Almost all integrated MSA tools have their own license. Users are invited to be aware of these third party licenses and respect them.

On another hand, Table 2 shows all integrated resources for MSA. The LRs building process is based on the Arabic language structure. The concatenative inflection denotes that the lemma concatenates to affixes to produce the stem, which in turn concatenates to clitics to yield the word. And according to their features, a lemma is either a verb, a noun or a particle. From this, we identify the basic components taking part in the composition of the Arabic words which are the lemmas (particle, verb and noun), stems and clitics. Thus, SAFAR follows the above Arabic language structure for lexical resources and contains the three basic alphabets (Loukili and Bouzoubaa 2011, Namly et al. 2016), clitics (Namly et al. 2015) and particles lexicon. We also make use of existing and known dictionaries (Contemporary and Interactive). It is worth mentioning that SAFAR contains currently one of the most comprehensive lexicons with more than 7 million stems and corresponding lemmas (Namly et al. 2019).

On another hand, because of the importance of ontologies in many NLP processes, we enriched and integrated the existing Arabic WordNet (Abouenour et al. 2013) (AWN). We note that enriched AWN is approved as the official version of the Global WordNet association[7].

Finally, we also developed and integrated corpora used as reference and evaluation corpora. Indeed, as mentioned above, these corpora as exploited to benchmark integrated tools at the stemming and morphological levels.

SAFAR resources are freely available for the community. They can be downloaded from our team website[8]. Indeed, in order to contribute in their wide dissemination within the community, we advertise on SAFAR resources in some well-known catalogs and repositories such as European Language Resources Association (ELRA)[9] and Common Language Resources and Technology Infrastructure (CLARIN)[10].

Finally, let us mention that a more detailed survey and a software engineering comparative study with similar Arabic frameworks can be found in (Jaafar and Bouzoubaa, 2018).

Layer	Package	Processing level	Implementation name	Size[11]	Per	Vr
Resource	Lexicon	Alphabet	SAFAR Alphabet	42	3	1
		Clitics	SAFAR Clitics	167	3	1
		Particles	SAFAR Particles	413	5	1
		Contemporary	Contemporary dictionary	32.300	2	2
		Interactive	Interactive dictionary	61.101	2	2
		CALEM	SAFAR Stems Lemmas	7.133.106	3	3
		Arabic WordNet	SAFAR Arabic WordNet	56.164	3	2
	Corpus	NAFIS	SAFAR Stemming gold standard	172	4	3
		Morpho evaluation	morphological analysis evaluation	100	3	2
		Stems evaluation	Quranic stemming evaluation	1000	3	2

Table 2: MSA resources implemented in SAFAR

3.2 Moroccan Dialect

Besides being interested in processing the Arabic language, we take into consideration the informal variety of Moroccan Arabic dialect (MD). Regarding resources, a Moroccan dialect electronic Dictionary (Tachicart et al. 2014) (MDED) has been developed containing almost 12,000 entries with useful annotations. Another lexicon is the Moroccan reference vocabulary (Tachicart et al. 2019) (MRV), which compiles 4.5M possible Moroccan words with respect to a normalization guideline.

Also, a corpus for language identification tasks is available with SAFAR. It is composed of 57k comments collected from social media and then manually classified into three categories: MSA, MD, and code-switched. Besides and based on neural models, a lexicon of orthographic variants that covers almost 54% of the MRV has been generated. It can be useful for several dialectal NLP tasks such as spelling normalization.

Table 3 shows all integrated resources for the Moroccan dialect. Concerning tools, a language identification system (Tachicart et al. 2018) has been developed and integrated within SAFAR in

[7] http://globalwordnet.org/resources/arabic-wordnet/

[8] http://arabic.emi.ac.ma/alelm/?q=Resources

[9] http://www.elra.info/en/

[10] https://www.clarin.eu

[11] Entries for lexicons and words for corpora

order to distinguish between MD and MSA. Besides, we developed and integrated a spelling normalization systems that helps to convert a given Moroccan dialectal word into its standard form without taking into consideration the word context.

Layer	Package	Processing level	Implementation name	Size[4]	Per	Vr
Resources	Lexicon	Mded	SAFAR Mded	12.000	2	3
		Moroccan_vocabulary	SAFAR MRV	4.500.000	2	3
		Orthographic_variants	SAFAR OV	2.385.000	2	3
	Corpus	LID	SAFAR Lang. Identification	519.000	2	3
Util	LID sys	SAFAR Lang. Identification	SAFAR Lang._Identification	--	2	3
	Spell	Spelling_normalization	SAFAR SPELL	--	2	4

Table 3: Moroccan dialect resources and tools implemented in SAFAR

3.3 Machine learning models

Our tools have been developed combining both the rule-based approach, embedded in lexicons and hardcoded, and the ML approach. Thus, SAFAR includes a set of popular ML libraries (Table 4) geared at different purposes, without the need to perform external tasks. For instance, the SAFAR POS tool exploited weka to output a Decision tree model (Tnaji et al., 2020), the SAFAR lemmatizer exploited HMM (Namly et al., 2020), while the Spelling normalization for the Moroccan dialect used fastText (Tachicart and Bouzoubaa, 2019). Consequently, a researcher making use of SAFAR has the possibility to code calling all integrated Arabic NLP tools and resources in addition to exploiting the integrated ML libraries.

Implementation name	Type	Per	Vr
Hidden markov model	Model	3	3
Language model	Model	2	3
Levenshtein	Model	2	3
Weka	Tool	1	3
FastText	Tool	1	3

Table 4: Machine learning models and tools in SAFAR

4 SAFAR use and exploitation

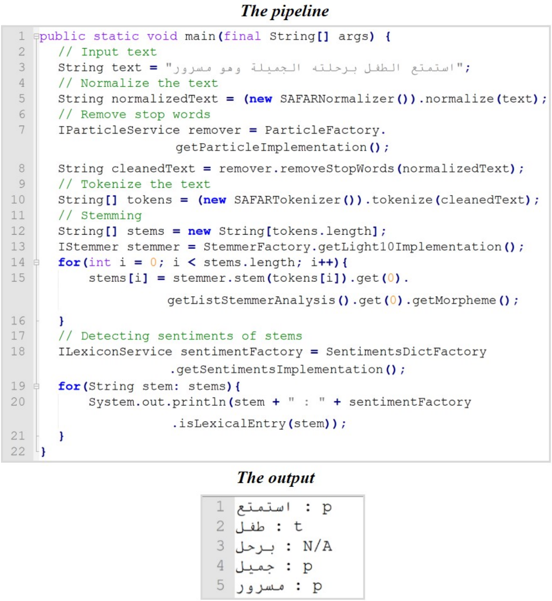
As previously mentioned, SAFAR tools and integrated resources can be exploited either as an API or from client applications.

4.1 API

For each level of processing, we standardize all aspects shared by the same type of tools according to APIs and models so that they become homogenous and flexible in their exploitation. This ensures the standardization inside SAFAR. Users have several possibilities when calling methods by specifying appropriate parameters according to their needs.

The execution of a normalizer within SAFAR can be simple as calling "normalizer.normalize(text)". If the normalization should be customized, overloaded methods can be called. It is worth mentioning that when developing the SAFAR API [12], we fully respect "Checkstyle" [13] and "FindBugs" [14] which are two development tools that help adhering to coding standards.

Users could also easily create customizable pipelines where the output of one component is the input of another (Jaafar and Bouzoubaa, 2015). All these aspects of SAFAR help solving SE issues especially the interoperability, the reuse and the flexibility of exploitation.



The pipeline

```
 1 public static void main(final String[] args) {
 2     // Input text
 3     String text = "استمتع الطفل برحلته الجميلة وهو مسرور";
 4     // Normalize the text
 5     String normalizedText = (new SAFARNormalizer()).normalize(text);
 6     // Remove stop words
 7     IParticleService remover = ParticleFactory.
                     getParticleImplementation();
 8     String cleanedText = remover.removeStopWords(normalizedText);
 9     // Tokenize the text
10     String[] tokens = (new SAFARTokenizer()).tokenize(cleanedText);
11     // Stemming
12     String[] stems = new String[tokens.length];
13     IStemmer stemmer = StemmerFactory.getLight10Implementation();
14     for(int i = 0; i < stems.length; i++){
15         stems[i] = stemmer.stem(tokens[i]).get(0).
                     getListStemmerAnalysis().get(0).getMorpheme();
16     }
17     // Detecting sentiments of stems
18     ILexiconService sentimentFactory = SentimentsDictFactory
                     .getSentimentsImplementation();
19     for(String stem: stems){
20         System.out.println(stem + " : " + sentimentFactory
                     .isLexicalEntry(stem));
21     }
22 }
```

The output

```
1 استمتع : p
2 طفل : t
3 برحل : N/A
4 جميل : p
5 مسرور : p
```

Figure 2: A pipeline using SAFAR API.

As mentioned in Figure 2, at line 3, we specify the input text. At line 5, we call the

[12] http://arabic.emi.ac.ma/safar-api/SAFAR_v3.jar

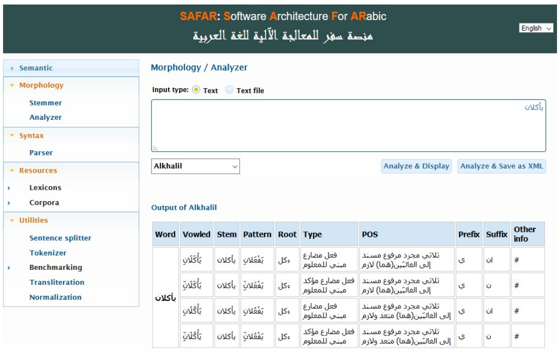
[13] https://checkstyle.sourceforge.io/

[14] http://findbugs.sourceforge.net/

"SAFARNormalizer" tool to normalize the text. At line 7 we call SAFAR "IParticleService" (Namly, et al. 2015) in order to delete stop words. At line 10, we instantiate the "SAFARTokenizer" tool which takes a text as input and outputs all tokens of the text. At line 13, we proceed to stemming tokens by calling the "IStemmer" service and specifying the Light10 stemmer in this case. At line 18, we call "ILexiconService" to detect stems sentiments and then print the sentiments of each word according to the predefined lexicon. Executing the whole process with another stemmer is simply to keep the same code and change only line 13 such as ".getKhojaImpletation".

4.2 Web application

For non-developers such as linguists, SAFAR framework can be executed using an online application [15] in which all SAFAR levels are developed as online processing. Accessing the website allows the user to have access to all tools and resources mentioned above. Results can be either printed on the same page or downloaded as XML files.



SAFAR: Software Architecture For ARabic

منصة سفر للمعالجة الآلية للغة العربية

Morphology / Analyzer

Input type: ● Text ○ Text file

يأكلان

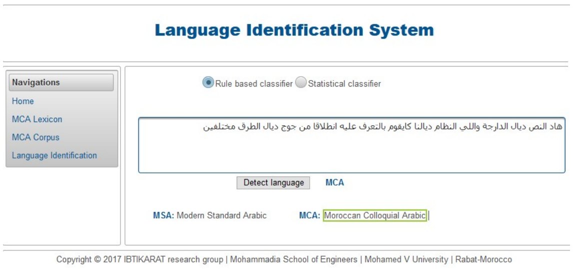
Alkhalil

Output of Alkhalil

Word	Vowled	Stem	Pattern	Root	Type	POS	Prefix	Suffix	Other info
يأكلان	يَأكَلَانِ	يأكلان	يَفْعَلَانِ	ءكل	فعل مضارع مبني للمعلوم	ثلاثي مجرد مرفوع مسند إلى الغائبَين(هما) لازم	ي	ان	#
	يَأكَلَانّ	يأكلان	يَفْعَلَانّ	ءكل	فعل مضارع مؤكد مبني للمعلوم	ثلاثي مجرد مرفوع مسند إلى الغائبَين(هما) لازم	ي	ن	#
	يَأْكُلَانِ	يأكلان	يَفْعُلَانِ	ءكل	فعل مضارع مبني للمعلوم	ثلاثي مجرد مرفوع مسند إلى الغائبَين(هما) متعد ولازم	ي	ان	#
	يَأْكُلَانّ	يأكلان	يَفْعُلَانّ	ءكل	فعل مضارع مؤكد مبني للمعلوم	ثلاثي مجرد مرفوع مسند إلى الغائبَين(هما) متعد ولازم	ي	ن	#

Figure 3: Alkhalil morphological analysis within SAFAR web.

As an example, Figure 3 shows the online morphological analysis for the word "يأكلان" (they eat). After selecting the morphological analyzer to use via the drop-down menu (Alkhalil in this case) and clicking on the "Analyze & display" button, the output is displayed in a table format.



Language Identification System

● Rule based classifier ○ Statistical classifier

هاد النص ديال الدارجة واللي النظام ديالنا كايقوم بالتعرف عليه انطلاقا من جوج ديال الطرق مختلفين

MCA

MSA: Modern Standard Arabic MCA: Moroccan Colloquial Arabic

Copyright © 2017 IBTIKARAT research group | Mohammadia School of Engineers | Mohamed V University | Rabat-Morocco

Figure 4: Language identification system.

Furthermore, the language identification system (Tachicart et al. 2018) demonstrated in Figure 4, aims to distinguish between Moroccan Dialect and MSA using two different methods. Indeed, the first is rule-based and relies on stop word frequency, while the second is statically-based and is based on an SVM machine learning classifier.

5 Conclusion

SAFAR is a monolingual framework dedicated to Arabic language. It is considered as a repository and collaborative work where multiple developers of Arabic tools and resources can meet and share their products. It is in its second decade of existence and integrates more than 50 tools and resources. The next steps of our journey are to:

- Concentrate on less considered layers such as semantics and applications;
- Integrate and develop other tools and resources for dialects and standard Arabic;
- Build bridges with multilingual or other language frameworks for developers interested to consider more than one language in their projects such as machine translation.

References

Abdelali, A., Darwish, K., Durrani, N., and Mubarak, H. 2016. Farasa: A fast and furious segmenter for Arabic. *In Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Demonstrations*, pp. 11-16.

Abouenour, L., Bouzoubaa, K., and Rosso, P. 2013. On the evaluation and improvement of Arabic WordNet coverage and usability. *Language Resources and Evaluation*, vol. 47, n° 13, pp. 891-917.

[15] http://arabic.emi.ac.ma:8080/SW_V3/

Algasaier, H. The ISRI Arabic Stemmer. 2005. http://www.nltk.org/_modules/nltk/stem/isri.html (accessed February 1, 2015).

Althobaiti, M., Kruschwitz, U., and Poesio, M. 2014. AraNLP: a Java-Based Library for the Processing of Arabic Text. In Proceedings of the 9th Language Resources and Evaluation Conference (LREC'14), Reykjavik, Iceland.

Boudchiche, M., Mazroui, A., Bebah, M. O. A. O., Lakhouaja, A., and Boudlal, A. 2017. AlKhalil Morpho Sys 2: A robust Arabic morphosyntactic analyzer. *Journal of King Saud University-Computer and Information Sciences 29, no. 2*: 141-146.

Boudlal, A., Lakhouaja, A., Mazroui, A., Meziane, A., Bebah, M. O. A. O., and Shoul, M. 2010. Alkhalil Morpho Sys1: A Morphosyntactic analysis System for Arabic texts. *Proceedings of the 11th International Arab Conference on Information Technology (ACIT'10). Benghazi*. 1-6.

Buckwalter, T. 2002. Buckwalter Arabic Morphological Analyzer Version 1.0." *Linguistic Data Consortium*.

Green, S., and Manning, C. D. 2010. Better Arabic parsing: baselines, evaluations, and analysis. *The 23rd International Conference on Computational Linguistics (COLING '10). Beijing: Association for Computational Linguistics*. 394-402.

Jaafar, Y. and Bouzoubaa, K. 2014. Benchmark of Arabic morphological analyzers: Challenges and Solutions. 9th International Conference on Intelligent Systems: Theories and Applications (SITA'14), Rabat,

Jaafar, Y. and Bouzoubaa, K. 2015. Arabic Natural Language Processing from Software Engineering to Complex Pipelines Cicling Cairo, Egypt.

Jaafar, Y., Namly, D., Bouzoubaa, K., Yousfi, A. 2016. Enhancing Arabic Stemming Process Using Resources and Benchmarking Tool. King Saud University - Computer and Information Sciences (JKSU-CIS).

Jaafar, Y., and Bouzoubaa, K. 2017. A New Tool for Benchmarking and Assessing Arabic Syntactic Parsers. 6th International Conference on Arabic Language Processing CITALA 2017, Fes, Morocco Fes, Morocco.

Jaafar, Y., Nasri, M., and Bouzoubaa, K. 2018. Semantic Analysis of Arabic Texts within SAFAR Framework. In proceedings of the 5th International IEEE Congress on Information Science and Technology (CIST'18), Marrakech, Morocco.

Jaafar, Y., and Bouzoubaa, K. (2018). A Survey and Comparative Study of Arabic NLP Architectures. In: Shaalan K., Hassanien A., and Tolba F. 2018. (eds) *Intelligent Natural Language Processing: Trends and Applications. Studies in Computational Intelligence*, volume 740. Springer, Cham.

Khoja, S. Khoja stemmer. 2002. http://zeus.cs.pacificu.edu/shereen/research.htm#stemming (accessed February 1, 2015).

Larkey, L. S., Ballesteros, L., and Connell, M. E. 2007. Light Stemming for Arabic Information Retrieval. In Arabic computational morphology: knowledge-based and empirical methods, 221-243. Springer Netherlands.

Lou B., and Syd, B. 2008. TEI P5: Guidelines for electronic text encoding and interchange". TEI Consortium.

Loukili,T., and Bouzoubaa, K. 2011. Structuration et Standardisation des ressources linguistiques de l'Arabe - cas de l'alphabet, préfixes et suffixes, Journées Doctorales en Technologies de l'Information et Communication, Tangier, Morocco, 7/ 2011.

Saad, M. K., and Ashour, W. M. 2010. Arabic morphological tools for text mining. *6th International Conference on Electrical and Computer Systems (EECS'10)*. Lefke, North Cyprus.

Namly, D., Bouzoubaa, K., Tajmout, R., Tahir, Y., and Khamar, H. 2015. A Complex Arabic stop-words list design. The Second National Doctoral Symposium On Arabic Language Engineering (JDILA'2015) ENSA of Fez USMBA.

Namly, D., Regragui, Y., and Bouzoubaa, K. 2016. Interoperable Arabic language resources building and exploitation in SAFAR platform. *In Proceeding of the 13th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA'16), Agadir, Morocco*.

Namly, D., Bouzoubaa, K., El Jihad, A., and Aouragh, S. L. (2020). Improving Arabic Lemmatization Through a Lemmas Database and a Machine-Learning Technique. *In Recent Advances in NLP: The Case of Arabic Language, pp. 81-100. Springer, Cham*.

Pasha, A., Al-Badrashiny, M., Diab, M., El Kholy, A., Eskander, R., Habash, N., Pooleery, M., Rambow, O., and Roth R. M. 2014. MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic. In Proceedings of the 9th Language Resources and Evaluation Conference (LREC'14), Reykjavik, Iceland.

Gülşen, E. 2014. ITU Turkish NLP Web Service. in European Chapter of the Association for Computational Linguistics, Sweden

Tachicart, R., Bouzoubaa, K., and Jaafar, H. 2014. Building a Moroccan dialect electronic Dictionary (MDED). In Proceedings of the 5th International Conference on Arabic Language Processing (CITALA'14), Oujda, Morocco.

Tachicart, R., and Bouzoubaa, K. 2019. Towards Automatic Normalization of the Moroccan Dialectal Arabic User Generated Text. In Arabic Language Processing: From Theory to Practice, *Springer International Publishing*, 2019, pp. 264-275.

Tachicart, R., Bouzoubaa, K., Aouragh, S. L., and Jaafar, H. 2018. Automatic Identification of Moroccan Colloquial Arabic. Arabic Language Processing: From Theory to Practice, *Springer International Publishing*, Cham, vol. 782, pp. 201-214.

Tnaji, K., Bouzoubaa, K., and Aouragh, S.L. 2021, A light Arabic POS Tagger using a hybrid approach. In *the international conference on digital technologies and applications*, January 29-30, 2021.

Zerrouki, T. Tashaphyne 0.2. 2012. https://pypi.python.org/pypi/Tashaphyne. Retrieved April 14, 2016.

Zhang, Y., Li, C., Barzilay, R., and Darwish, K. 2015. Randomized greedy inference for joint segmentation, POS tagging and dependency parsing. *In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 42-52.