SITA'14 - 9th International Conference on Intelligent Systems: Theories and Applications. 07-08 May 2014, Rabat, Morocco

Benchmark of Arabic Morphological Analyzers Challenges and Solutions

Younes Jaafar*
Mohammadia School of Engineers
Mohammed Vth University - Adgal
Rabat, Morocco.
younes.jaaf@gmail.com

Karim bouzoubaa
Mohammadia School of Engineers
Mohammed Vth University - Adgal
Rabat, Morocco.
karim.bouzoubaa@emi.ac.ma

***Abstract*—Arabic Natural Language Processing (ANLP) has known an important development during the last decade. Nowadays, several ANLP tools are already developed such as morphological analyzers. These analyzers are often used in more advanced applications such as syntactic parsers, search engines, machine translation systems, etc. However, the choice of a morphological analyzer to use, among others, can be difficult for researchers if they ignore its metrics. In this article, we present the challenges of the benchmark of Arabic morphological analyzers. We present also our solution developed in Java, which allows the benchmark by returning the most common metrics, namely the accuracy, precision, f-measure and execution time. This solution has the advantage of being cross-platform, flexible and allows to be extended to cover new morphological analyzers to compare.**

Keywords*—Arabic morphological analyzers; Benchmark; Standard corpus; SAFAR platform

I. INTRODUCTION

The digital world is experiencing a quick and continuous growth in terms of Arabic content (texts, videos, images etc.) From 2000 until 2011, this content has known a large growth estimated by 2501% [1]. Therefore, the processing of this content requires the development of appropriate tools dedicated to Arabic Natural Language Processing (ANLP). Nowadays, many tools for ANLP are already developed, such as syntactic parsers [2], search engines [3], machine translation systems [4], etc. Most of these tools rely on morphological analyzers to define the structure of words [5,6]. Such analyzers are primarily concerned by the decomposition of words into morphemes and by associating to each morpheme morphological information such as the stem, root, POS (Part Of Speech), affixes, etc. Among these analyzers we find BAMA [7,8], MADA+Tokan [9] and Alkhalil [10,11]. For example, the word « **أحسب** » ('OHsb') may be analysed to output the following morphological information: stem = "**حسب**", pattern = "**فَعَلُ**", prefix="**أ**: همزة الاستفهام", root = "**حسب**".

Given its importance, it is essential for a researcher in ANLP to make an optimal choice when selecting a morphological analyzer to use in his research project. The solution generally used by researchers in ANLP is to develop specific programs to compare two given morphological analyzers using a given corpus also. For example, Sawalha and Atwell have proposed a program [12] to evaluate the results of Khoja stemmer [13], BAMA [7] and a root extractor [14]. However, developing such program which is not dynamic and extensible causes several problems of reusability, because it will be often modified and adapted to cover a new analyzer to compare or a new evaluation corpus to use. To overcome this, generic solutions have emerged such as U-Compare [15] but they don't concern Arabic language. This means the development of a generic tool for the benchmark of Arabic morphological analyzers must be done. Thus, our effort is an initiative to fill this gap. It is in our view an efficient way to optimize efforts, collaboration and accelerating developments in the field.

Furthermore, note that when evaluating a morphological analyzer, researchers use metrics related to accuracy of results. However, current Arabic language volume available on the Internet became that large, that it is difficult to deal with using morphological analyzers that do not optimize their execution time. It becomes then essential, when comparing analyzers, to take into account not only the metrics related to the results but also the execution time.

Thus, our objective is to provide an opportunity for researchers to benchmark Arabic morphological analyzers. We have developed a generic solution that compares the results returned by each analyzer according to a corpus that is annotated and manually checked by linguists, and calculates also the execution time of each one of them. This benchmark is used to check the accuracy and recall of results and also to compare analyzers against each other based on their execution time.

To give concrete examples of using our solution of the benchmark, we selected the two most used morphological analyzers in ANLP community, namely BAMA and Alkhalil. They are free and open source and they will serve as example throughout this article. It should be noted that our solution is not limited to the comparison of these two analyzers. Any other

analyzer may be compared using the same tool without any changes.

The rest of the paper is organized as follows. The next section presents BAMA and Alkhalil and the used evaluation corpus. In section 3, we show the challenges of the benchmark of Arabic morphological analyzers. In Section 4, we present our approach to develop our solution. The details of the development of this solution are then presented in Section 5. In Section 6, we present the experiments and results of the benchmark of BAMA and Alkhalil. We present the flexibility of our solution in Section 7. And finally a conclusion in section 8.

II. TWO MORPHOLOGICAL ANALYZERS AND ONE EVALUATION CORPUS

A. *Presentation of two morphological analyzers*

BAMA: Is a morphological analyzer for Arabic coded with Perl by Tim Buckwalter [7]. BAMA uses three components in order to analyze a text: the lexicon, the compatibility tables and the analysis engine. Since we target portability, we used AraMorph [16] which is a Java version of BAMA.

Alkhalil: Is a morphological analyzer for Arabic written in Java [17]. This analyzer identifies all possible solutions of a word and establishes a list of morphological features of these solutions (type, pattern, root, POS …). The output can be either in HTML or CSV formats.

B. *Presentation of the evaluation corpus*

In order to perform the benchmark process, the results returned by a morphological analyzer must be compared to results of an annotated corpus. This corpus should be verified manually by linguists to maximize its precision. We selected the « Gold Standard of Arabic » [18] as our evaluation corpus. It is considered by its authors as a standard for the evaluation of Arabic morphological analyzers, because it contains an important number of information relevant to the morphology such as stem, root, affixes, POS, etc. It consists of the chapter number 29 of the holy Qur'an, « sourhat Al-ankaboot ». This corpus contains 976 words (575 unique words without diacritics) which are analyzed according to their context, annotated and checked by Arabic linguists.

III. THE MORPHOLOGICAL ANALYZERS BENCHMARK CHALLENGES

Comparing the results of the two morphological analyzers (BAMA and Alkhalil) using the selected evaluation corpus highlights several challenges that we must overcome:

Challenge 1 - Heterogeneity of the form and format: Each of the two morphological analyzers has its own internal structure, its own form/format of output and its own execution environment. For example, the output of BAMA is in the form of text file, while the output of Alkhalil is in the form of HTML/CSV file. Thus, a solution must be considered so that both analyzers share the same characteristics, in order to make the comparison easier.

Challenge 2 - Different numbers of analyzes: It should be noted that the evaluation corpus that we have selected is annotated according to word contexts. Thus, each word has only one analysis with several known tags set such as root, prefix, suffix, stem etc. However, the two morphological analyzers we selected do not take into account the context of the word while analyzing. As a result, the number of analysis of each word will be different from one morphological analyzer to another. For example, for the word « **أحسب** » ('OHsb'), Alkhalil returns 49 possible analysis while BAMA returns only 3. In such case, the challenge is how to compare the set of analysis of each word in the morphological analyzer result according to the single appropriate analysis in the evaluation corpus and how to be fair concerning this comparison.

Challenge 3 - Different tags used: There is a big difference in the tags set used either in the evaluation corpus or in the morphological analyzers results. For example, the pattern and the root for each analysis are mentioned in both Alkhalil results and the evaluation corpus, but not in BAMA results. According to the evaluation corpus tags set, it will not be fair to evaluate BAMA's results according to the root, otherwise, all of BAMA's results will be considered as wrong. In such case, we should find a compromise between the different tags set, in order to take into account only those that are returned by the morphological analyzer and which are present in the evaluation annotated corpus.

Challenge 4 - Several presentations of data**:** Because there is also no standard form of morphological analyzers outputs, the comparison of these results becomes a challenging task. For example, Alkhalil returns a concatenated presentation of prefix for some words, while this same prefix is separated into letters in the evaluation annotated corpus. For example for the word «وَلَيَعْلَمَنَّ», Alkhalil returns a prefix containing two units, namely : « **وَلَ** » + « **ي**» while this same prefix is decomposed into three units inside the evaluation corpus, namely : « **وَ**» + « **لَ**» + « **يَ**». Thus, the challenge here is to find a solution to avoid penalizing an analyzer having a concatenated prefix, because it is correct in the two cases presented. To overcome this, one solution is to concatenate the prefix into a single unit either in analyzer results or in the evaluation corpus, in this way the prefix shown above will have the same following structure «**وَلَيَ**» .

Challenge 5 - Presence of diacritics: Alkhalil returns elements without diacritics while they are present in the evaluation annotated corpus. For example, for the word « **أحسب** », Alkhalil returns the stem "**حسب**" without diacritics, whereas in the evaluation corpus we find the stem "**حَسِبَ**" with diacritics. Thus, these diacritics are removed for some elements such as the stem and the prefix/suffix, in order to make comparison possible.

IV. OUR APPROACH : USING SAFAR PLATFORM

We decided to integrate our benchmark solution into the SAFAR platform[19,20,21,22]. Indeed, SAFAR (Software Architecture For Arabic language pRocessing) is an integrated platform dedicated to ANLP written in Java. It is cross platform, modular, extensible, flexible and provides an integrated development environment (IDE).

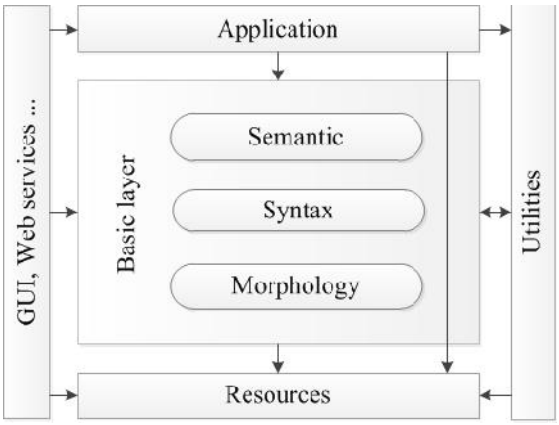

FIG I. ARCHITECTURE DE LA PLATEFORME SAFAR

As specified in Fig. 1, SAFAR has several layers. The utilities layer includes a set of technical services, the resources layer provides services for consulting language resources such as lexicon, the basic layer contains the three regular layers (morphology, syntax and semantics), the application layer contains high-level applications that use the layers listed above, and finally the client applications layer which interacts with all other layers providing to users web applications, web services, etc.

Our solution of the benchmark is integrated in the utilities layer of SAFAR. This solution could be reused in any other layer of the platform without any modification. In addition, the dynamic aspect of SAFAR makes this tool capable to be integrated into any other Java project. It should also be noted that the characteristics of SAFAR allow easy integration of new morphological analyzers to compare and new evaluation corpus to use. Thus, an ANLP researcher will be able to either use directly our benchmark tool that takes into account several analyzers by default, or easily integrate the new analyzer within SAFAR in order to be compared to others via the same tool. Thus, the technical characteristics of SAFAR meet the objective of genericity and reusability of our benchmark.

V. DEVELOPING THE BENCHMARK SOLUTION

In order to develop our solution of benchmarking, we have divided our work into several phases:

Phase1 - Integration: This step, consisting of integrating the two morphological analyzers, has been done previously when constructing the morphological layer [21] of SAFAR. This integration allowed us to master the output format of any integrated morphological analyzer. In addition, this output format is standardized according to the standard format proposed by Alecso [23] that we consider to be more compatible with the rules of the Arabic language. Thus, we guarantee that all results have the same structure (Fig. 2) and then it will be easy to compare them. Via this integration, we answer the challenge number 1 concerning the heterogeneity of the form and format of the morphological analyzers outputs.

```
<?xml version="1.0" encoding="UTF-8"?>
<morphology_analysis total_words="1">
  <word w_id="1" value="أحسب" total_analysis="49">
    <analysis a_id="1" vowled="أَحْسَبُ" stem="حسب"
              pattern="فعل" root="حسب"
              pos="مفرد مذكر مرفوع في حالة الاضافة"
              number="مفرد" gender="مذكر" mood="مرفوع"
              case="في حالة الاضافة " type="اسم جامد"
              prefix="[أ: همزة الاستفهام]" suffix="#"
              additional_info=""
    />
    <!-- 48 autres analyses ici pour le mot أحسب... -->
  </word>
</morphology_analysis>
```

FIG II. EXAMPLE OF SAFAR XML OUTPUT USING ALKHALIL IMPLEMENTATION

The Fig. 2 shows the XML output of SAFAR concerning the results of the integrated morphological analyzers. This XML file contains all the analyzes of one or more words in the input text. Each of the analyzes contains a set of tags such as root, stem, prefix, suffix, etc. For example, for the word « **أحسب** » without diacritics, Alkhalil returns 49 possible analyzes, the Fig. 2 shows precisely the analysis with the id number 1 and which has tags such as: vowled = "**أَحْسَبُ**" (which represents the word with diacritics), stem = "**حسب**", pattern = "**فَعَلُ**", root = "**حسب**", etc.

Phase2 - Transformation: Because it's difficult to compare two different structures of information, we have transformed the xml file of the evaluation annotated corpus into another xml file in order to match the stantard format of Alecso. This is essential to ensure data consistency. The xml file resulting from this transformation will be used in Phase 3. It should also be noted that this file is now part of the resources of SAFAR platform.

Phase3 – Development of the Benchmark utility: Once all morphological analyzers and evaluation corpus are integrated in SAFAR and have the same structure of results, we have developed a Java utility that compares the results returned by the morphological analyzers against those of the evaluation corpus. This utility does the Benchmark in four steps (Fig. 3).

In step 0, the utility converts the file resulting of the previous phase2, which actually represents the evaluation corpus, into memory objects according to SAFAR API. We opted for the use of memory objects instead of output files in order to be faster when benchmarking. In step 1, each morphological analyzer processes the input text of the evaluation corpus. The results of each morphological analyzer are then retrieved as memory objects in step 2. In step 3, the utility compares the results of each morphological analyzer with those of the evaluation annotated corpus. Thus, for a given word, it is enough for the morphological analyzer to return a single correct analysis to be considered as a good result for such a word, which answers to the challenge number 2 concerning the different number of analyzes.

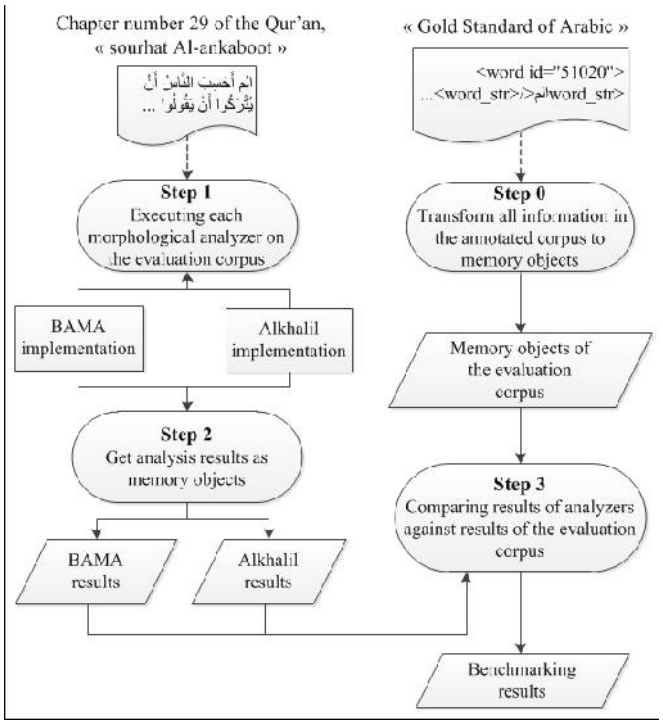

FIG III. STEPS FOR DEVELOPING THE MORPHOLOGICAL ANALYZERS BENCHMARK SOLUTION

Phase4 – Metrics**:** The last phase of the benchmark is to return a list of metrics on which researchers can rely to measure the performance of a given morphological analyzer. To measure this performance, we use the usual metrics [1] for this kind of benchmark as follows: the precision, recall, accuracy, and F-measure. Table 1 lists the set of metrics that we return.

TABLE I. METRICS RETURNED BY OUR SOLUTION OF MORPHOLOGICAL ANALYZERS BENCHMARK

About the corpus	Morphological analyzer results	Statistical metrics	Technical metrics
N words N unique words	N analyzed words N correct analysis N incorrect analysis N words not analyzed	Precision Recall Accuracy F-measure	Exec. time

VI. EXPERIMENTS AND RESULTS

The two morphological analyzers (BAMA and Alkhalil) have been compared using the Gold Standard of Arabic as evaluation corpus. Here are the results of this comparison:

TABLE II. COMMON METRICS OF COMPARING BAMA AND ALKHALIL USING THE GOLD STANDARD OF ARABIC AS EVALUATION CORPUS

Common metrics	BAMA	Alkhalil
Total analyzed words	543	546
Total words not analyzed	32	29
Execution time	8.977 s	68.791 s

[1] http://fr.wikipedia.org/wiki/Pr%C3%A9cision_et_rappel

Table 2 presents the global common metrics for the morphological analyzers after the benchmak, namely: the number of analyzed words, the number of words not analyzed as well as the execution time of each one. As indicated in the table, BAMA and Alkhalil analyze the majority of words in the evaluation corpus, they reach respectively 543 and 546 of the analyzed words. This means that they are nearly equal on this point, it only remains to check the reliability of their results. However, they have a large difference in terms of execution time. BAMA completes the analysis in 8.9 seconds while Alkhalil takes 68.7 seconds before completing. This means that in the context of Big Data, it will be preferred to use BAMA instead of Alkhalil since the execution time is a critical metric. It should also be noted that these experiments were performed on a computer having the following characteristics: CPU = Core 2 Duo @2.13 GHZ, RAM = 4GO, Operating System = Win7, 32bits.

TABLE III. RESULTS OF COMPARING BAMA AND ALKHALIL USING THE GOLD STANDARD OF ARABIC AS EVALUATION CORPUS

Evaluation according to	Metrics	BAMA	Alkhalil
Root	N correct analysis	Not returned	**431**
	N incorrect analysis	Not returned	115
	Precision	Not returned	78.94%
	Recall	Not returned	74.96%
	Accuracy	Not returned	74.96%
	F-measure	Not returned	76.9%
Stem	N correct analysis	**475**	**313**
	N incorrect analysis	68	233
	Precision	87.48%	57.33%
	Recall	82.61%	54.43%
	Accuracy	82.61%	54.43%
	F-measure	84.97%	55.84%
Pattern	N correct analysis	Not returned	**209**
	N incorrect analysis	Not returned	337
	Precision	Not returned	38.28%
	Recall	Not returned	36.35%
	Accuracy	Not returned	36.35%
	F-measure	Not returned	37.29%
POS	N correct analysis	**517**	**528**
	N incorrect analysis	26	18
	Precision	95.21%	96.7%
	Recall	89.91%	91.83%
	Accuracy	89.91%	91.83%
	F-measure	92.49%	94.2%
Lemma	N correct analysis	**376**	Not returned
	N incorrect analysis	167	Not returned
	Precision	69.24%	Not returned
	Recall	65.39%	Not returned
	Accuracy	65.39%	Not returned
	F-measure	67.26%	Not returned

The experiments of Table 3 concern the specific metrics of the two morphological analyzers BAMA and Alkhalil. We ran

each analyzer using the evaluation corpus as input and we compared the results each time according to a collection of tags, namely the root, the stem, the pattern, the POS, and the lemma. This answers the challenge number 3 concerning the different tags used. This list is not exhaustive of course, but it represents the common tags that often appear in the results of morphological analyzers.

It should be noted that before analyzing a text, Alkhalil deletes duplicated words, while BAMA does not. In order to be fair while experimenting, we deleted duplicated words from the evaluation corpus used. Thus, the two morphological analyzers will have to deal with the same number of unique words.

The results show that Alkhalil extracts most of the roots from the evaluation corpus, it achieves 78.94% and 74.96% of precision and accuracy respectively, while BAMA does not return roots. On the other side, BAMA is a good stemmer compared to Alkhalil, it achieves 87.48% and 82.61% of precision and accuracy respectively. For the pattern, Alkhalil reach only 38,28% et 36,35% of precision and accuracy respectively while BAMA does not return this tag. Concerning the POS, both of BAMA and Alkhalil achieve good rates for all the metrics. Finally, BAMA extract most lemmas, it reaches 69,24% et 65,39% of precision and accuracy respectively, while Alkhalil does not return it.

VII. FLEXIBILITY OF OUR SOLUTION

The benchmark of morphological analyzers inside SAFAR is not limited to BAMA and Alkhalil. Every morphological analyzer can be integrated in the platform and be compared via the same utility. Thereby, there are no changes to make on the benchmark utility side, only integration of new analyzers is necessary. In addition, any other annotated corpus can be integrated also in SAFAR platform and be used as an evaluation corpus. Thus, end users will have a large choice of morphological analyzers to compare and evaluation corpora to use. In addition to that, we offer two possible solutions to use our benchmark solution:

Developers solution: By including the jar[2] of SAFAR in any Java project. Once this is done, the benchmark utility can be executed using some few lines of code (Fig. 5).

At line 13, the list of morphological analyzers to compare is established. This list is then filled in lines 14 and 15 by the two morphological analyzers BAMA and Alkhalil. The benchmark tool is executed in lines 18 and 19, it takes as parameters the list of analyzers to compare and the evaluation corpus to use. At lines 21, 22 and 23, the results are displayed as a final step by iterating over all results. Thus, the benchmark tool can be executed in a few lines of code regardless the number of analyzers to compare and the evaluation corpus to use. For example, if we want to add a new analyzer named A3 to the list of analyzers to compare, only one line should be added to the code after line 15, namely:

```
listAnalyzersImplNames.add(Analyzer.A3);
```

Similarly, if we want to change the evaluation corpus in order to use another that is named CORPUS2, only the line 19 should be changed by setting «`Corpus.CORPUS2`» instead of « `Corpus.GOLD_STANDARD_OF_ARABIC` ».

Web solution: The benchmark solution can be also executed using the web application[3] (Fig 4). This solution can be used either by developers or linguists without writing any line of code. More information and documentation about the use of the tool of the benchmark can be found in the examples on the website[4].

Output

About Corpus		
Total Unique Corpus Words	575	
Benchmark Results	BAMA	Alkhalil
Total Analyzed Words	543	546
Total Correct Analyses	517	528
Total Incorrect Analyses	26	18
Total Not Analyzed Words	32	29
Metrics	BAMA	Alkhalil
Precision	95.21%	96.7%
Recall	89.91%	91.83%
Accuracy	89.91%	91.83%
F-measure	92.49%	94.2%
Execution Time	8.8 s	67.41 s

FIG IV. RESULTS OF THE BENCHMARK OF BAMA AND ALKHALIL BASED ON THE POS AND USING THE WEB SOLUTION

VIII. CONCLUSION

In this paper, we presented a solution for benchmarking Arabic morphological analyzers. We outlined some challenges to take into account when developing a system for comparing Arabic analyzers. Then we presented our general solution written in Java to solve this problem. This solution is integrated in the Software Architecture For Arabic language pRocessing (SAFAR) platform. This integration allows more flexibility and interoperability. We selected two Arabic morphological analyzers (BAMA and Alkhalil) in order to compare their results and give an example of the use of this solution. The evaluation corpus we used to perform this task is the Gold Standard of Arabic, it consists of the chapter number 29 of the Qur'an, « sourhat Al-ankaboot ». This chapter contains 976 words, 575 of them are unique words.

As future work, we plan to deal with new morphological analyzers by integrating them into SAFAR platform, and also new evaluation corpora. Thus, end users will have a large choice of morphological analyzers to compare and evaluation corpora to use.

Finally, to further refine the benchmark process, other metrics will also be taken into account such as the complexity of the analysis algorithms and the richness of information returned by the morphological analyzers (tags returned and not returned).

[2] http://sibawayh.emi.ac.ma/safar/download.php

[3] http://sibawayh.emi.ac.ma:8080/SafarWeb/BenchmarkController

[4] http://sibawayh.emi.ac.ma/safar/examples.php#Benchmark

```
1  package util;
2
3  import common.constants.Analyzer;
4  import common.constants.Corpus;
5  import java.util.ArrayList;
6  import java.util.List;
7  import safar.util.benchmark.morphology.analyzer.MorphologyAnalyzerBenchmark;
8  import safar.util.benchmark.morphology.analyzer.MorphologyAnalyzerMetrics;
9
10 public abstract class BenchmarkTest {
11     public static void main(final String[] agrs) {
12         // prepare the list of morphological analyzers to be compared
13         List<String> listAnalyzersImplNames = new ArrayList();
14         listAnalyzersImplNames.add(Analyzer.BAMA);
15         listAnalyzersImplNames.add(Analyzer.ALKHALIL);
16
17         //compare the list of morphological analysers using an avaluation coprus
18         List<MorphologyAnalyzerMetrics> results = MorphologyAnalyzerBenchmark.
19                 compare(listAnalyzersImplNames, Corpus.GOLD STANDARD OF ARABIC);
20         //print the results of benchmark
21         for (int i = 0; i < results.size(); i++) {
22             printMetrics(results.get(i));
23         }
24     }
25 }
```

FIG V. EXECUTING THE MORPHOLOGICAL ANALAYZERS BENCHMARK SOLUTION USING JAVA CODE

REFERENCES

[1] Miniwatts Marketing Group. (2001) Internet World Stats. [Online]. http://www.internetworldstats.com

[2] Spence Green and Christopher D. Manning, "Better Arabic parsing: baselines, evaluations, and analysis, COLING '10," in *The 23rd International Conference on Computational Linguistics*, 2010.

[3] Mamoun Hattab, Bassam Haddad, Mustafa Yaseen, Asem Duraidi, and A. Abu Shmais, "Addaall Arabic Search Engine: Improving Search based on Combination of Morphological Analysis and Generation Considering Semantic Patterns," in *The 2nd International Conference on Arabic Language Resources & Tools*, 2009, pp. 159-162.

[4] Caroline Champsaur, "La traduction automatique : un outil pour les traducteurs?," *The Journal of Specialised Translation*, no. 19, January 2013.

[5] Al-Sughayer Imad and Al-Kharashi Ibrahim, "Arabic morphological Analysis Techniques: a comprehensive Survey," *Journal of the American Society for information Science and Technology*, vol. 55, no. 3, pp. 189–213, February 2004.

[6] Kais Dukes and Nizar Habash, "Morphological Annotation of Quranic Arabic," in *Language Resources and Evaluation Conference (LREC)*, Malta, 2010.

[7] Buckwalter. (2002) QAMUS. [Online]. http://www.qamus.org/morphology.htm

[8] T. Buckwalter, "Buckwalter Arabic Morphological Analyzer Version 1.0", 2002.

[9] Nizar Habash, Owen Rambow, and Ryan Roth. MADA+TOKAN. [Online]. http://www1.cs.columbia.edu/~rambow/software-downloads/MADA_Distribution.html

[10] (2013) Alkhalil Morpho Sys. [Online]. http://sourceforge.net/projects/alkhalil/

[11] A. Boudlal, A. Lakhouaja, M. Azzeddine, and M. Abdelouafi, "Alkhalil Morpho Sys1: A Morphosyntactic analysis System for Arabic texts," in *Proceedings of ACIT'2010*, 2011.

[12] Majdi Sawalha and Eric Atwell, "Comparative Evaluation of Arabic Language Morphological Analysers and Stemmers," in *International Conference on Computational Linguistics - COLING*, 2008, pp. 107-110.

[13] Shereen Khoja. (2002) Stemming Arabic text. [Online]. http://zeus.cs.pacificu.edu/shereen/research.htm

[14] Al-Serhan H., Al Shalabi R., and Kannan G., "New approach for extracting Arabic roots," in *Arab conference on Information Technology (ACIT'2003)*, 2003, pp. 42-59.

[15] Y. Kano, R. Dorado, L. McCrochon, S. Ananiadou, and J. Tsujii, "U-Compare: An integrated language resource evaluation platform including a comprehensive UIMA resource library," in *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC 2010)*, 2010, pp. 428-434.

[16] Pierrick Brihaye. (2003) AraMorph. [Online]. http://www.nongnu.org/aramorph/english/index.html

[17] برنامج الخليل الصرفي. (2011) أهم المزايا الفنية لبرنامج الخليل الصرفي. [Online]. http://www.alecso.org.tn/index.php?option=com_content&task=view&id=1302&Itemid=956&lang=ar

[18] Majdi Sawalha. Gold Standard for Evaluating Arabic Morphological Analyzers. [Online]. http://www.comp.leeds.ac.uk/sawalha/goldstandard.html

[19] Younes Souteh and Karim Bouzoubaa, "SAFAR platform and its morphological layer," in *Eleventh Conference on Language Engineering ESOLEC'2011*, Cairo, 2011.

[20] Karim Bouzouba and Younes Souteh, "Vers une plateforme intégrée pour le traitement automatique de la langue Arabe," in *Technologies d'Information et Communication en Langue Arabe TICLA*, Paris, 2011.

[21] Younes Souteh and Karim Bouzoubaa, "Approche de mise en œuvre du niveau morphologique dans la plateforme SAFAR," in *International Workshop on Information Technologies and Communication WOTIC'2011*, Casablanca, 2011.

[22] Seddik Sidrine, Younes Souteh, Karim Bouzoubaa, and Taoufik Loukili, "SAFAR: vers une plateforme ouverte pour le traitement automatique de la langue Arabe," in *The 6th Intelligent Systems: Theory and Applications SITA*, Rabat, 2010.

[23] Alecso. مواصفات نظام التحليل الصرفي في اللغة العربية. [Online]. http://www.alecso.org.tn/images/stories/OULOUM/MOHALLILAT%20SARFIA_DAMAS_2009/022%202%20SPECIFICATIONS.pdf