Advances in Science, Technology and Engineering Systems Journal
Vol. 2, No. 3, 100-110 (2017)
www.astesj.com

ASTES Journal
ISSN: 2415-6698

Dependence-Based Segmentation Approach for Detecting Morpheme Boundaries

Ahmed Khorsi, Abeer Alsheddi[*]

Computer Science Department, Al-Imam Mohammad Ibn Saud Islamic University, Kingdom of Saudi Arabia

A R T I C L E I N F O

Article history:
Received: 04 March, 2017
Accepted: 01 April, 2017
Online: 10 April, 2017

Keywords :
Classical Arabic language
Computer science
Corpora
Morpheme boundaries
Unsupervised learning
Word segmentation

A B S T R A C T

The unsupervised morphology processing in the emerging mutant languages has the advantage over the human/ supervised processing of being more agiler. The main drawback is, however, their accuracy. This article describes an unsupervised morphemes identification approach based on an intuitive and formal definition of event dependence. The input is no more than a plain text of the targeted language. Although the original objective of this work was classical Arabic, the test was conducted on an English set as well. Tests on these two languages show a very acceptable precision and recall. A deeper refinement of the output allowed 89% precision and 78% recall on Arabic.

1 Introduction

This article is an extension of work originally presented in Proceedings of 4th Saudi International Conference on Information Technology (Big Data Analysis) [1]. The article discusses one of the main challenges in computational linguistics is the identification of morpheme boundaries [2, 3, 4]. Boundary detection process plays a central role in extracting stems from words [5]. Moreover, the noisy infixes affect the performance of some tasks such as similarity computation [6, 7, 8]. The fascinating results of this process performed easily on specific examples made computer learning of language morphology a desirable approached for a long time; as are simple problems hiding dissuasive details. As a result of that, Zelig Harris [9] way of segmenting words into morphemes by only counting letters that may occur after a prefix and marking the picks seems to be the first approach published. Since that, investigations did not cease on building a model easily maintainable to grasp how words are coined in a human language. Benefits of unsupervised approaches step beyond avoiding costly human efforts on known languages. These approaches also have the advantage of being fast enough and to some extent language independent.

In Semitic languages [10, 11] such as Arabic, the challenge is even harder because of the irregularities, like mutations and diphthongs situations. Mutation means that one letter can be replaced by another to fit the pronunciation. Take the Arabic word "اضطجع" ([AD∘TajaE]: lie-down)[1] comes from applying the pattern "ا؟ت؟؟" and then mutates the third letter "ت" ([t]: 3rd Arabic letter) to "ط" ([T]: 16th Arabic letter). This mutation also can be shown with a long vowel[2] deleted or replaced by another. For example, the past tense of the verb "قول" ([qawl]: say) is "قال" ([qaAl]: said) by mutating "و" ([w]: 27th Arabic letter) into "ا" ([A]: 1st Arabic letter), whereas the imperative form of the same verb is "قل" ([qul]: say) with the letter "و" ([w]: 27th Arabic letter) deleted. The second situation, diphthong; means that two consecutive identical letters are merged into one letter. For example, the word "شدّ" ([$ad∼]: pull) is originally "شدْدَ" ([$ad∘da]).

The proposed approach is developed to the design of a fully computerized (i.e., unsupervised) extraction of the morphemes. In light of this goal, the approach uses simple and intuitive statistical feature extracted by reading a corpus of plain text with no tag to become able to identify the morphemes boundaries. The approach also was compared with an existed approach named Morfessor [13]. Although the original goal of this work was to address the much

[*]Corresponding Author Email: abeer.alsheddi@gmail.com (A. Alsheddi)

[1]In this article, Arabic is represented in some or all of three variants according to context: "Arabic word" ([Buckwalter Arabic transliteration] [12]: English translation).

[2]Three Arabic letters have long vowels: "ى" or "ا" ([A]: 1st Arabic letter), "و" ([w]: 27th Arabic letter) and "ي" ([y]: 28th Arabic letter)

richer vocabulary of the classical Arabic, the test was conducted on an English set as well. The results are a proof of concept that unsupervised techniques can accurately handle a morphology as complex as the one of a legacy Semitic language.

The study brings the following values:

1. A model: An intuitive but formal model was developed for morphology learning based on a simple interpretation of the probabilistic dependence [14] to segment a word into morphemes.

2. Languages

   - Classical Arabic: The performance of the model was reported on the morphology of the classical Arabic, which is the language of a huge legacy literature. Arabic morphology is more complicated than simple languages [15]. Because of these complex features, some of the methods widely used in many languages cannot be applied to Arabic [6, 11]. On the other hand, two known varieties of Arabic are classical Arabic and Modern Standard Arabic (MSA). MSA is based on the classical Arabic. However, MSA is used more in informal speech and can combine words borrowed from other languages that may not use Arabic rules. In contrast, the classical Arabic has been used to write literary legacy and traditional vocabulary. It contains purer Arabic words than MSA. Moreover, Arabic morphological and semantic rules are derived from the classical Arabic. At the same time, the researchers on Arabic are paying most of their attention to MSA. To the best of our knowledge, the only studies taking into account the classical Arabic are limited to religious texts.

   - Other Languages: The performance of the model was tested on a concatenative language namely English. It has a totally different degree of morphological complexity from Arabic [16].

3. Comparison: The performance of another word segmentation approach named Morfessor was reported on the same dataset.

4. Corpus: It is a language resource defined as a collection of texts stored in a machine-readable format. To assess the proposed approach, we had to build a corpus of classical Arabic texts authored in the period from 431 to 1104 (in Hijri between 130 b.h and 498 h) counting around 122M words in total and 1M distinct words.

The remaining parts of this article are organized as follows: Section 2 reviews existing methodologies on detecting morpheme boundaries. Section 3 introduces the proposed approach and explains its algorithm. Section 4 discusses the tests and the results it carried out. Finally, section 5 concludes with a summary of contributions and makes suggestions for future research work.

2 Applied Methodologies

One of the oldest published work on unsupervised word segmentation is due to Harris who suggested a process to break a phonetic text into morphemes [9] using only successors numbering. Bordag [17] partially relies on Letter Successor Variety (LSV) due to Zelig Harris [9, 2, 3] and combines it with a trie based classifier [18]. The approach is claimed to reach 67.48% F-measure. Bernhard [19] too combined the same principle of successors variation with a set of heuristics to filter out the less plausible segments. The F-measure is claimed to reach 60.81%.

One of the well-known systems in this area is Morfessor, developed by Creutz and Lagus [13]. Morfessor tries to capture the morpheme boundaries in a probabilistic model with simplistic features such as the frequency and the length of the morpheme. For comparison, we investigate this approach deeper in Section 4.1.4. Eroglu, Kardes, and Torun [20] tried to make Morfessor 1.0 accommodate the phonetic features of the Turkish language. By adding phone-based features the tried to measure the phonemic confusability between the morphemes. Their results are claimed to be better than Morfessor's for Turkish, Finnish and English. Pitler and Keshava [5] use the forward and backward conditional probability to build a list of substantial affixes. They report a 52% F-measure on their test set and 75% on the gold-standard.

A number of works depend on Markov Models (MM).Melucci and Orio [21] investigated the use of the Hidden Markov Models (HMMs). The evaluation though is done on a whole information retrieval system (extrinsic) [22]. Naradowsky and Toutanova [23] designed a model, which tries to exploit the alignment of morphemes in a source language to the morphemes in a target (machine translated to) language. With different settings of the model, the approach is claimed to reach 84.6 F-measure for Arabic.

Peng and Schuurmans [24] proposed a two-level hierarchical Expectation Maximization (EM) model. Their test set holds 10K words and their results are claimed to reach 69% F-measure. Poon, Cherry and Toutanova [25] use a log-linear model [26, 27] to capture the features inherent to the morpheme itself and its surrounding letters. The authors tested the approach on supervised and semi-supervised settings as well. The unsupervised settings reached a 78.1% F-measure for Arabic and 70% for Hebrew.

It is worth mentioning that, the actual list of similar works is longer. Thus, we limited the overview to the previous works only. Some of the extra works are in the surveys [28] and [29] that provided for the interested reader. In light of the proposed goal in this article, this section intentionally avoids supervised approaches [30].

On the other hand, other works in real environments use the segmentation technique in their approaches, such as document image retrieval systems [31, 32].

3 The Proposed Approach

The word *"unbreakable"* is coined by concatenating the prefix *"un"*, the stem *"break"* and the suffix *"able"*. The vocabulary, which suggests such segmentation, should contain other words with different combinations of prefixes, stems and suffixes (e.g. *"rebreakable"*, *"unbreaking"*...etc), which makes the occurrence of *"un"* have a weak dependence to the occurrence of *"break"* whose occurrence is also relatively independent of the occurrence of the suffix *"able"*. On the other hand, each morpheme is supposed to be inseparable from neither its first nor its last letter. The proposed approach (Dependence-Based Segmentation) is all about exploiting these two facts. A good formal modeling of these facts is the definition of the probabilistic dependence [14].

3.1 Segmentation Algorithm

Algorithm 1 iterates over the word W letter by letter and computes for every letter w_i two dependences: 1. the dependence of the prefix on w_i as its last letter, 2. the dependence of the suffix on w_i as its first letter where the prefix ends (inclusive) at w_i and the suffix starts (inclusive) at it. The difference then points to which of the two (i.e. the prefix or the suffix) is more attached to the current letter w_i. The algorithm keeps going until it encounters a change of the direction of the attachment. The change is done when the prefix depends on the preceding letter w_{i-1} more than the suffix does, and at the same time the suffix depends on the current letter w_i more than the prefix does. Then between the previous and the current letters a cutting point (CP) is marked.

3.2 Computation of the Dependence

The concept of dependence we use is symmetric [14]. In this context: *"the string depends on the letter"* means *"the letter depends on the string"* and vice versa.

We will call the dependence of a letter a on the prefix α: the forward dependence, and we denote FD(u) where u=αa is the prefix tailed by the letter under processing a. We call the dependence of the letter a on the suffix β: the backward dependence, and we denote BD(v) where v=aβ is the suffix headed by the letter under processing a. We also mark the beginning and the end of a word by respectively # and \$. The forward dependence is then:

$$FD(u) = \frac{P(\alpha \to a)}{P(a)} \tag{1}$$

$$BD(u) = \frac{P(a \leftarrow \beta)}{P(a)} \tag{2}$$

Where $P(\alpha \to a)$ is the conditional probability:

$$P(\alpha \to a) = \frac{Count(\alpha a)}{Count(\alpha)} \tag{3}$$

and $P(a \leftarrow \beta)$ is the conditional probability:

$$P(a \leftarrow \beta) = \frac{Count(a\beta)}{Count(\beta)} \tag{4}$$

$$FD(u) = \frac{Count(\alpha a)}{Count(\alpha)P(a)} \tag{5}$$

$$BD(u) = \frac{Count(a\beta)}{Count(\beta)P(a)} \tag{6}$$

Where the probability of a letter a: P(a) is estimated by its normalized frequency in the corpus.

$$P(a) = \frac{Count(a)}{\sum_{b \in \mathcal{A}} Count(b)} \tag{7}$$

Where $\mathcal{A}$ is the alphabet. Count(α) expresses how often an n-gram α occurs in the corpus.

By the definition of the dependence measure [14], when both forward and backward dependencies at the same letter are lower than one, this means that the letter does not depend on the prefix or the suffix. Thus, we assigned zero to the difference's value and discarded this position from the segmentation process. This dependence is called a *negative dependence* [14].

After segmenting a word using the cutting points, we supposed that a morpheme with the least frequency among the other morphemes is a stem and the other morphemes are affixes.

Arabic example: The Arabic word "الكتابان" ([AlokitaAbaAk]: the two books) contains two cutting points as shown in Table 1. The first one occurs when the second letter "ل" is more dependence on the prefix "#ال" than the suffix "$لكتابان, while the third letter "ك" is more dependent on the suffix "$كتابان" than the prefix "#الك". The second position is the sixth letter "ب" when the dependence tendency is changed. Thus, these two cutting points "ال|كتاب|ان" produce the following morphemes: "ال", "كتاب" and "ان". Consequently, the morpheme "كتاب" is the stem depending on the frequency list in Table 3.

Negative dependence example: The Arabic word "كتيب" ([kutayb]: small book) does not contain any cutting point as the table 2 shows it. Consequently, the morpheme "كتيب" is the stem of the word. At the same time, we can note the second letter "ت" ([t]: 3ed Arabic letter) has zero as its difference value. That means both forward and backward dependencies are less than one.

English example: Table 4 is a simulation of the algorithm on the word *"unbreakable"*. The difference at each letter ($FD(u)$-$BD(v)$) points a dependence direction of that letter. In other words, the positive difference means that the current letter w_i is more attached to the prefix $w_{1...i}$ than suffix $w_{i...n}$ and vice versa. The second letter *"n"* depends on the prefix *"#u"* more than it does on the suffix *"breakable\$"*, where the third letter *"b"* depends on the suffix *"reakable\$"* more than it does on the prefix *"#un"*. This change of the dependence direction makes the point *"un|breakable"* a cutting point. The same logic applies to the seventh letter

Algorithm 1: Word Morphological Segmentation

```
input  : A word $W = w_0 w_1 \dots w_n$
output: Segments of word W
1  Cutting points: CP
2  foreach $w_i$ where $0 < i \leqslant n$ do
3      if $FD(\# \dots w_i) < 1$ and $BD(w_i \dots w_n\$) < 1$ then
4          discard this point
5      else
6          if $FD(\# \dots w_{i-1})$-$BD(w_{i-1} \dots w_n\$) > 0$
7          and $FD(\# w_0 \dots w_i)$-$BD(w_i \dots w_n\$) < 0$ then
8              $CP \leftarrow CP \cup i$
9  $seg \leftarrow$ segment $W$ at each $i$ in $CP$
10 return $seg$
```

Algorithm 2: N-grams model construction algorithm

```
Input   : A string of words $W$
Output: $n$-grams mapped to their frequency
1  foreach $W$ in string do
2      W ← '#'+W+'$'
3      for $i \leftarrow 1$ to $|w| - 1$ do
4          for $j \leftarrow i + 1$ to $|w| - 1$ do
5              if $w_{i \dots j} \in n\text{-}grams$ then
6                  $freq(w_{i \dots j}) \leftarrow freq(w_{i \dots j}) + 1$
7              else
8                  $n\text{-}grams \leftarrow n\text{-}grams \cup \{w_{i \dots j}\}$
9                  $freq(w_{i \dots j}) \leftarrow 1$
10 return $n$-grams and $freq$
```

Table 1: Arabic example of the segmentation

i	u	FD(u)	v	BD(v)	Difference	Direction
1	#ا	0.991	الكتابان$	7.174	-6.182	↓
2	#ال	9.349	لكتابان$	9.153	0.195	↑
3	#الك	1.017	كتابان$	17.174	-16.157	↓
4	#الكت	0.716	تابان$	1.210	-0.494	↓
5	#الكتا	2.532	ابان$	0.544	1.987	↑
6	#الكتاب	12.067	بان$	1.552	10.514	↑
7	#الكتابا	1.366	ان$	1.865	-0.498	↓
8	#الكتابان	4.560	ن$	1.792	2.767	↑

Table 2: Example of the negative dependence

i	u	FD(u)	v	BD(v)	Difference	Direction
1	#ك	1.125	كتيب$	5.631	-4.505	↓
2	#كت	0.646	تيب$	0.349	0	0
3	#كتي	0.685	يب$	2.030	-1.344	↓
4	#كتيب	6.675	ب$	0.256	6.419	↑

Table 3: A frequency list of chosen words

Morpheme	Frequency	Morpheme	Frequency
un	1,986	ال	72,845
break	46	كتاب	47
able	1,426	ان	10,047

Table 4: English example of the segmentation

i	u	FD(u)	v	BD(v)	Difference	Direction
1	#u	0. 46	unbreakable$	30.97	-30. 51	↓
2	#un	6. 36	nbreakable$	5. 84	0. 52	↑
3	#unb	1. 35	breakable$	46. 70	-45. 35	↓
4	#unbr	1. 94	reakable$	10. 45	-8. 51	↓
5	#unbre	3. 11	eakable$	5. 01	-1. 90	↓
6	#unbrea	6. 79	akable$	1. 68	5. 11	↑
7	#unbreak	34. 71	kable$	1. 50	33. 21	↑
8	#unbreaka	5. 09	able$	7. 38	-2. 29	↓
9	#unbreakab	46. 70	ble$	9. 89	36. 81	↑
10	#unbreakabl	9. 72	le$	2. 76	6. 96	↑
11	#unbreakable	10. 02	e$	1. 15	8. 87	↑

"*k*" and the eighth letter "*a*". Therefore, the resulting morphemes will be: "*un*", "*break*" and "*able*". The least frequent morpheme is "*break*" by using Table 3. Consequently, the morpheme "*break*" is the stem.

4 Tests and Results

Before describing the test settings, we will list main steps, that we followed in the study:

1. Build the corpus. (see Section 4.1.1).
2. Make the normalization (see Section 4.1.2).
3. Generate the language model (see Section 4.1.3).
4. Test the proposed approach (see Section 4.1.4) using materials (see Section4.1.5).
5. Evaluate the performance by using the metrics (see Section 4.1.6).

4.1 Test Settings

4.1.1 Test Dataset

Arabic The purpose of the research, which drew our efforts to design this approach, was to address challenges in processing the morphology of the classical Arabic. Works we are aware of try to handle MSA. Actually, MSA is known to include a mixture of words borrowed from other languages and words just coined for the convenience, which usually breaks the classical Arabic rules. Correctly handling the classical Arabic brings three interrelated values: 1. The classical Arabic vocabulary is much richer in terms of pure Arabic words. 2. The Arabic rules in the morphological, grammatical and semantic levels amply documented since the eighth century have been designed on the basis of the classical Arabic. 3. A huge legacy literature is written in the classical Arabic, and most of the time standing beyond the scope of the supervised approaches designed for the MSA. The present corpus[3] gathers Arabic texts dating back from 431 to 1104 (130 b.h and 498 h). It contains around 122M words in total and 1M distinct words with an average size of 6.22 letters per word. Table A.1 in the appendix shows a sample of the index in Arabic that was published with our corpus. Such dataset is by itself a valuable resource for researches in NLP field.

English We obtained the English text by merging the two corpora "wiki" and "news"[4]. This resulted in more than 1M unique words and more than 8M words in total, having an average size of 8.00 letters per word.

4.1.2 Normalization

The normalization process is frequently used to transform text into an approved form, which aids in reducing the noise and sparsity in the text. In present work, the following normalization was applied to the Arabic words:

1. The removal of diacritics.
2. The substitution of all variants of Hamza "ء" ([´]) with the form "ء" ([´]).
3. The substitution of the letter "آ" ([|]) with "اء" ([´A]).

4.1.3 Language Model

The proposed language model extracts all possible *q*-grams from a word of *n* length, where *q* rises from one to *n*. Then it assigns each gram to its occurrence in the corpus. Algorithm 2 explains the extraction. Table A.2 in the appendix shows the *n*-grams of the two example words "الكتابان" and "*unbreakable*".

4.1.4 Test Process

Morfessor 2.0[5] and the proposed approach (Dependence-Based Segmentation) are run over the same corpus, then three samples of 100 words each are randomly picked out of the whole segmented corpus. The results of these approaches are evaluated manually. Precision, recall and F-measure of the cutting points are then recorded. This is repeated on each of the four settings combinations where each setting combines one of two occurrence settings with one of two affix settings.

Occurrence Settings

- Distinct: To build the language model, the approach used distinct words, i.e. each word in the corpus occurs one time.
- Plain: The number of occurrences is taken into account during building the model, i.e. a word may occur more than once.

Affix Settings

- Raw: Results sample is picked randomly with no restriction.
- Non-empty affix: Samples are picked randomly only among words for which the proposed approach has carried out at least one cutting point. For a number of words, the segmentation simply did not identify any cutting point. Most of them were because of the scarcity of the stem or stem.affix remaining combination. We then tried to assess the impact of such cases on the performance and how accurate the identified cutting was.

It is worth noting that our objective was to address the challenge of segmenting the classical Arabic words. To the best of our knowledge, there is no suitable gold standard for TA. We had to build ourselves the set of words. Then we proceeded to the manual segmentation of three different randomly picked samples for each setting.

[3]`https://sourceforge.net/projects/classical-arabic-corpus/`

[4]`http://corpora.informatik.uni-leipzig.de/download.html`

[5]One of our objectives is to keep the process entirely unsupervised. Thus, we used the latest version without using any optional parameters to discard any semi-supervised extension.

4.1.5 Test Platform

We used the following materials to test the proposed algorithm:

- Operating system: Ubuntu 12.04, 64-bit.
- Programming language: C++.
- Compiler: GCC 4.9.0.
- Database management system: MySQL Server 5.5.43.
- C++ external Library: libmysqlcppconn-dev 1.1.0-3build1 to connect C++ with MySQL server.
- CPU: Intel(R) Xeon(R) CPU E5530 @ 2.40GHz * 16.
- RAM: 6 GB for RAM and 12 GB for Swap.

4.1.6 Performance Metrics

For evaluating, we used three metrics: recall, precision and F-measure [33], where we took into account a number of cutting points during the evaluation.

1. Recall: This metric measures how many correct positions are found among all the existing correct positions. The higher the recall, the more correct positions are found and returned.

$$Recall = \frac{\textit{correct cutting points in the result}}{\textit{all correct cutting points in the sample}}$$

2. Precision: Precision measures how many positions are actually correct among all the positions that the algorithm found. The high precision indicates that the algorithm found significantly the correct positions more than incorrect positions.

$$Precision = \frac{\textit{correct cutting points in the result}}{\textit{all found cutting points in the result}}$$

3. F-measure: A weighted average of the recall and the precision is measured.

$$F - measure = 2 * \frac{Recall * Precision}{Recall + Precision}$$

4.2 Result

4.2.1 Arabic Results

Compared to the English, the results of the proposed approach (Dependence-Based Segmentation)are clearly lower on Table 5. The two obvious causes might be the irregularities in Arabic morphology and the typos in the test set. The latter is confirmed by results obtained when the sample is restricted to the words with a relatively high frequency (thresholded) as shown in Figure 1. Thresholded denotes an additional filter applied to *non-empty affix* sample, where a segmentation is picked only if the affix reappears in more than 1K other segmentations. It is worth noting that we omitted this filter in the English tests because we observed a rise in values of the standard deviation between the three samples. It has fluctuated around 0 to 29 words. Reasons for this deviation need to be investigated more closely in a separate work, and we paid our attention in this work to Arabic results.

Another reason affects the performance which is the lack of a dataset. The proposed approach works totally with the unsupervised method and depends only on the count of words in the corpus. However, it is difficult to include all possible derivatives in the corpus. For example, the derivative “الإجحافات” ([Al◦<ij◦HaAfaAt]: the prejudices) appears one time in the dataset; that is, there is no other derivative that appears in the dataset without the prefix ”ال”, for instance. Indeed, the word ”إجحافات” would be more dependent on the prefix ”ال” and the proposed approach does not learn that the prefix ”ال” can be cut off from the word ”الإجحافات” as shown at the top of Table 6. This problem will be removed if we add the word ”إجحافات” into the dataset. The approach then finds the segmentation position ”ال|إجحافات” as it appears at the bottom of Table 6.

The segmentation seems to be the best when the word counts are omitted (distinct). This seems to be due to the diversity of affixed in the classical Arabic. Indeed some stems/affixes may not appear more than once in the whole corpus (e.g. “كجخف” ([kajux◦f])as a deep sleep).

Surprisingly, Morfessor was unable to detect any boundary. The expected reason is that Morfessor may not be able to deal effectively with non-Latin letters, such as Arabic letters. We then evaluated Morfessor using Buckwalter Arabic transliteration [12]. Despite the simplicity of the proposed approach, the results of Morfessor is close to the results of the proposed approach especially after applying the filter.

Running the proposed approach over the whole corpus resulted in around 1,805,231 morphemes and 596,358 distinct morphemes with an average size of 5.94 letters per morpheme.

4.2.2 English Results

With no restriction on the results sample, Morfessor outperforms the proposed approach (Dependence-Based Segmentation) in terms of precision where the latter has a better recall as shown in Table 7 . This matches well the results given in [34]. The low recall of Morfessor explains the worsening of its performance with affixed words while the proposed approach improves. The F-measure shows that the overall performance of the proposed approach is better as shown in Figure 2.

When the counts of the words are involved in the computation of the dependence, Morfessor improves. The dependence-base’s precision increases too, where its recall worsens.

Table 5: Arabic results

		Distinct		Plain	
		Dependence	Morfessor	Dependence	Morfessor
Raw	Precision	81.21±2.76	86.0215±0.61	78.66±6.73	83.6547±4.32
	Recall	48.11±1.32	76.0147±1.94	44.88±3.98	75.9473±1.74
	F-measure	60.62±1.47	80.7091±1.08	57.14±4.96	79.6149±1.67
Non-empty affix	Precision	78.73±4.97	83.2916±3.37	78.02±1.96	83.9520±5.34
	Recall	74.96±2.44	75.1702±4.66	69.26±1.90	72.0343±5.03
	F-measure	76.80±2.54	79.0228±4.08	73.38±1.90	77.5379±5.19
Thresholded	Precision	89.36±5.18	90.1493±4.60	85.80±8.13	89.9908±4.07
	Recall	78.69±2.30	73.5789±1.29	72.78±7.89	73.0709±2.74
	F-measure	83.69±3.49	81.0256±2.57	78.76±7.90	80.6530±2.43

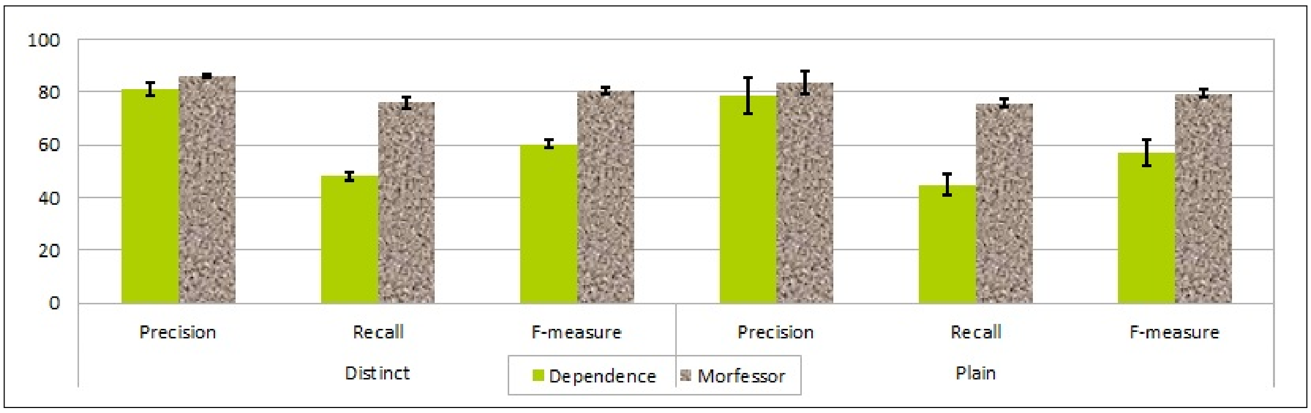
Table 6: Example of the lack of a dataset

i	u	FD(u)	v	BD(v)	Difference	Direction
1	#ا	0.991	الإجحافات$	7.174	-6.182	↓
2	#ال	9.349	لإجحافات$	12.204	-2.855	↓
3	#الإ	2.637	إجحافات$	26.555	-23.917	↓
4	#الإج	2.096	جحافات$	5.499	-3.402	↓
5	#الإجح	1.019	حافات$	3.443	-2.423	↓
6	#الإجحا	4.484	افات$	1.404	3.079	↑
7	#الإجحاف	11.911	فات$	1.296	10.614	↑
8	#الإجحافا	3.587	ات$	2.422	1.164	↑
9	#الإجحافات	17.184	ت$	0.885	16.298	↑
i	u	FD(u)	v	BD(v)	Difference	Direction
1	#ا	0.991	الإجحافات$	7.174	-6.182	↓
2	#ال	9.349	لإجحافات$	6.102	3.246	↑
3	#الإ	2.637	إجحافات$	26.555	-23.917	↓
4	#الإج	2.096	جحافات$	10.081	-7.984	↓
5	#الإجح	1.019	حافات$	3.374	-2.714	↓
6	#الإجحا	4.484	افات$	1.411	3.072	↑
7	#الإجحاف	11.911	فات$	1.298	10.612	↑
8	#الإجحافا	3.587	ات$	2.422	1.164	↑
9	#الإجحافات	17.184	ت$	0.885	16.298	↑

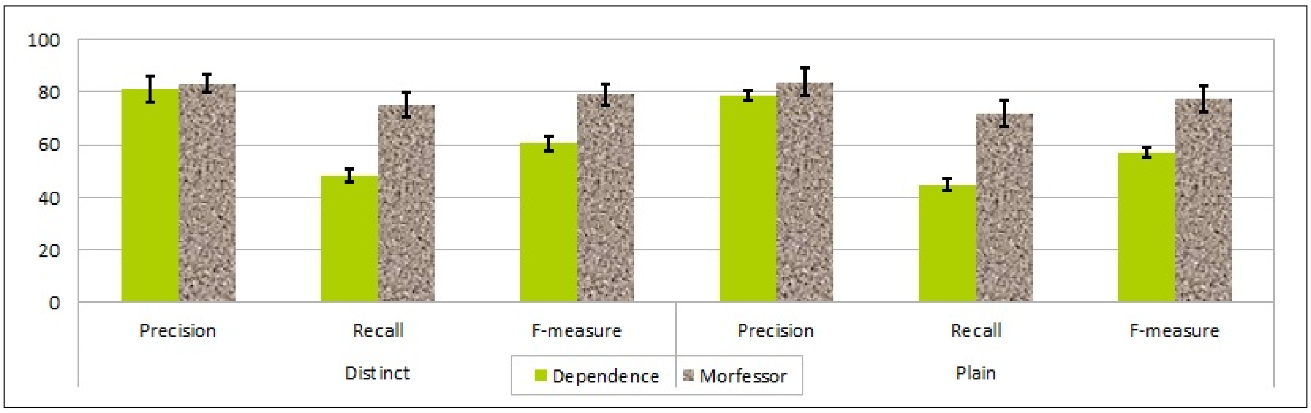
Table 7: English results

		Distinct		Plain	
		Dependence	Morfessor	Dependence	Morfessor
Raw	Precision	71.60±1.68	87.36±2.64	74.99±1.14	90.72±4.83
	Recall	78.11±3.98	70.90±4.84	79.66±1.88	71.98±4.73
	F-measure	74.71±2.17	78.27±3.57	77.25±0.73	80.27±2.52
Non-empty affix	Precision	75.82±1.98	89.73±1.74	76.26±2.34	91.37±1.24
	Recall	88.10±5.73	67.11±6.12	84.19±4.61	68.77±3.40
	F-measure	81.50±3.43	76.79±4.64	80.03±2.81	78.47±1.88

(A) Raw sample



(B) Non-empty affix sample



(C) Thresholded sample

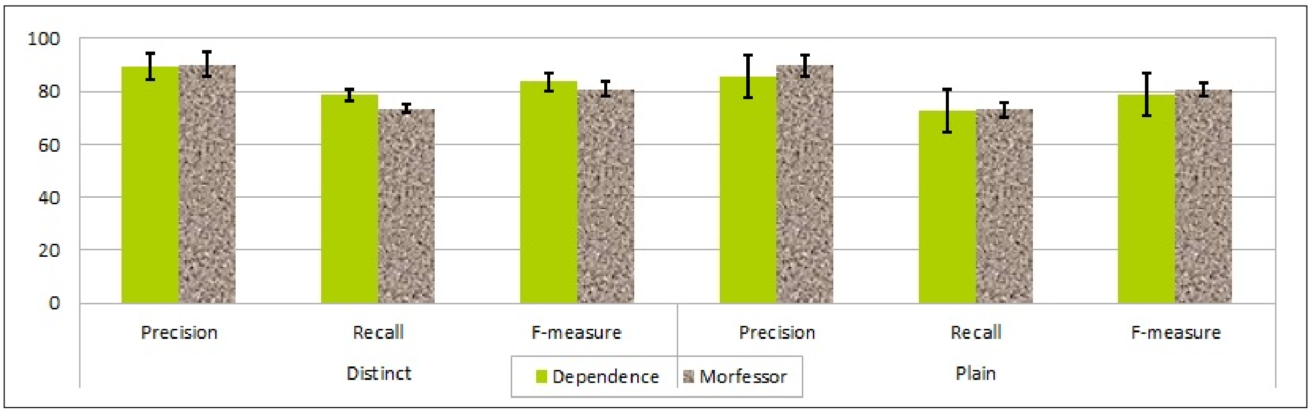


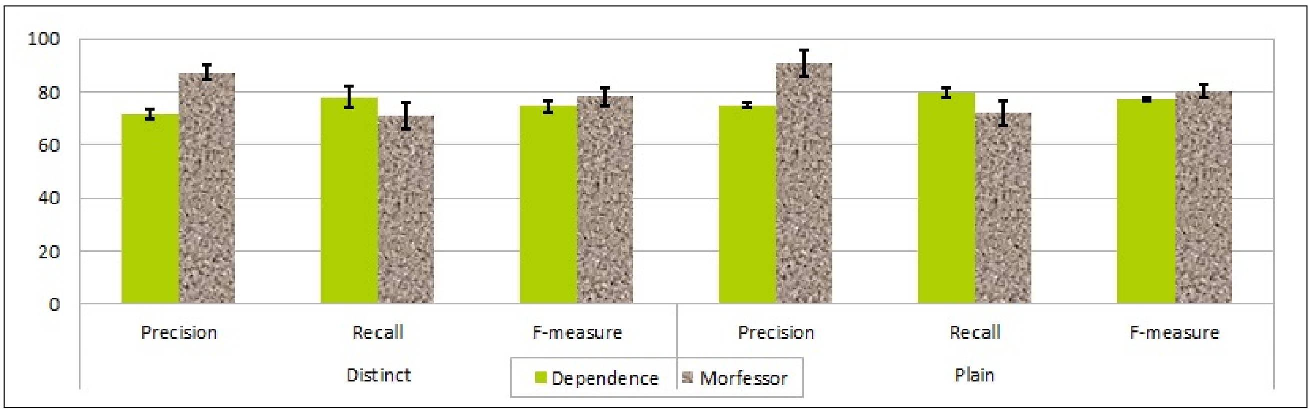
Figure 1: Arabic results of the two approaches

5 Conclusion and Future Work

We investigated the use of an intuitive yet formal definition of event dependence in the detection of morpheme boundaries. The initial target of our work was the classical Arabic, which is a Semitic language recognized to have a complex morphology and a vocabulary richer in pure Arabic words than the Modern Standard Arabic (MSA). The test also was conducted on an English set as well. Results on the classical Arabic and on English have been compared to the results of another unsupervised complicated segmentation approach so-called Morfessor that initially designed for European languages. As simple and elegant as is the proposed approach (Dependence-Based Segmentation) , it seems to be more committed to the recall where the latter's precision is better on English words. This has been confirmed when the evaluation focused on affixed words. On Arabic, Morfessor could not identify boundary. The results are a proof of concept that unsupervised techniques can decently handle a morphology as complex as the one of a legacy Semitic language like Arabic.

In addition, required for the evaluation, we carefully built a pure classical Arabic dataset. Where such linguistic resource is abundant in languages like En-

(A) Raw sample



(B) Non-empty affix sample

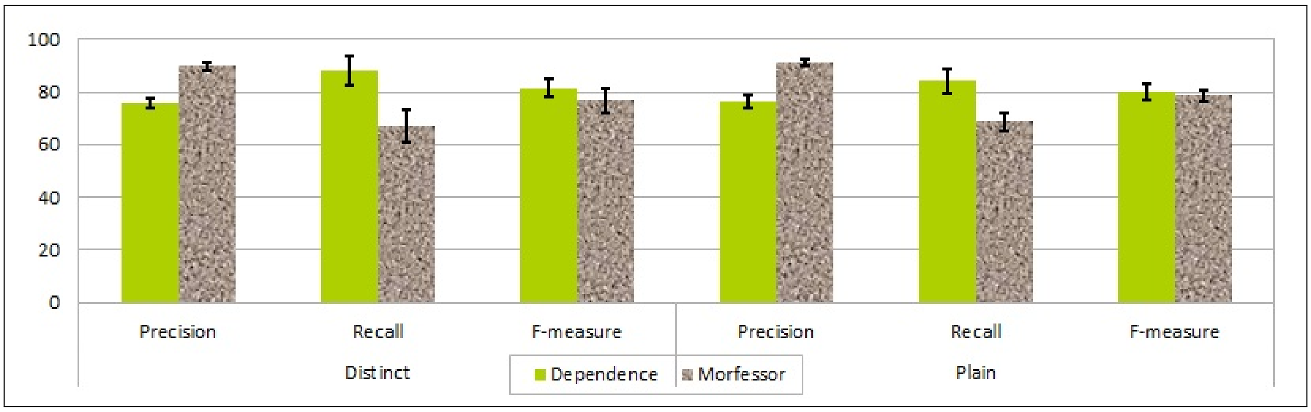


Figure 2: English results of the two approaches

glish, we are not aware of any other classical Arabic dataset.

One of the things made possible by morpheme boundaries identification is the automated word generation. The segmentation of a small set of well-chosen words will result in a set of diverse morphemes. The generation of new words is then a proper concatenation of the resulting morphemes. The challenge is then two folds: to build a suitable word-set of a decent size, which allows an accurate segmentation and a wide coverage of the different morphemes, then to find the appropriate concatenation strategy to minimize the number of irrelevant words.

References

[1] A. Khorsi and A. Alsheddi, "Unsupervised detection of morpheme boundaries," in *2016 4th Saudi International Conference on Information Technology (Big Data Analysis) (KACSTIT)*, Nov 2016, pp. 1–7.

[2] Z. Harris, *Morpheme Boundaries Within Words: Report on a Computer Test*, ser. Pennsylvania. University. Dept. of Linguistics. Transformations and discourse analysis papers. University of Pennsylvania, 1967.

[3] M. A. Hafer and S. F. Weiss, "Word segmentation by letter successor varieties," *Information Storage and Retrieval*, vol. 10, no. 11–12, pp. 371 – 385, 1974.

[4] J. R. Saffran, E. L. Newport, and R. N. Aslin, "Word segmentation: The role of distributional cues," *Journal of Memory and Language*, vol. 35, no. 4, pp. 606 – 621, 1996.

[5] S. Keshava and E. Pitler, "A segmentation approach to morpheme analysis," in *Working Notes for the CLEF Workshop 2007*, Hungary, 2007, pp. 1–4.

[6] S. H. Mustafa, "Character contiguity in n-gram-based word matching: The case for Arabic text searching," *Information Processing & Management*, vol. 41, no. 4, pp. 819–827, Jul. 2005.

[7] S. H. Mustafa, "Word-oriented approximate string matching using occurrence heuristic tables: A heuristic for searching Arabic text," *Journal of the American Society for Information Science and Technology*, vol. 56, no. 14, pp. 1504–1511, Dec. 2005.

[8] A. Khorsi, "On morphological relatedness," *Natural Language Engineering*, vol. 19, no. 4, pp. 537–555, Oct. 2013.

[9] Z. S. Harris, "From phoneme to morpheme," *Language*, vol. 31, no. 2, pp. 190–222, Jun. 1955.

[10] G. A. Kiraz, *Computational Nonlinear Morphology with Emphasis on Semitic Languages*. Cambridge, England: Cambridge University Press, 2001.

[11] S. Wintner, "Morphological processing of semitic languages," in *Natural Language Processing of Semitic Languages*, ser. Theory and Applications of Natural Language Processing, I. Zitouni, Ed. Springer Berlin Heidelberg, 2014, pp. 43–66.

[12] N. Habash, A. Soudi, and T. Buckwalter, "On Arabic transliteration," in *Arabic Computational Morphology*, ser. Text, Speech and Language Technology, A. Soudi, A. v. d. Bosch, and G. Neumann, Eds. Springer Netherlands, 2007, no. 38, pp. 15–22.

[13] M. Creutz and K. Lagus, "Unsupervised models for morpheme segmentation and morphology learning," *ACM Transactions on Speech and Language Processing*, vol. 4, no. 1, pp. 1–34, Jan. 2007.

[14] R. Falk and M. Bar-Hillel, "Probabilistic dependence between events," *The Two-Year College Mathematics Journal*, vol. 14, no. 3, pp. 240–247, Jun. 1983.

[15] D. Jurafsky and J. H. Martin, *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition*. Prentice Hall, 2009.

[16] I. Al-Sughaiyer and I. Al-Kharashi, "Arabic morphological analysis techniques: A comprehensive survey," *Journal of the American Society for Information Science and Technology*, vol. 55, no. 3, pp. 189–213, Feb. 2004.

[17] S. Bordag, "Unsupervised and knowledge-free morpheme segmentation and analysis," in *Advances in Multilingual and Multimodal Information Retrieval*. Springer, 2008, pp. 881–891.

[18] D. Morrison, "Patricia&mdash;practical algorithm to retrieve information coded in alphanumeric," *JACM: Journal of the ACM*, vol. 15, no. 4, pp. 514–534, Oct. 1968.

[19] D. Bernhard, "Simple morpheme labelling in unsupervised morpheme analysis," in *Advances in Multilingual and Multimodal Information Retrieval*. Springer, 2008, pp. 873–880.

[20] O. Eroglu, H. Kardes, and M. Torun, "Unsupervised segmentation of words into morphemes," 2009.

[21] M. Melucci and N. Orio, "A novel method for stemmer generation based on hidden markov models," in *Proceedings of the twelfth international conference on Information and knowledge management*. ACM, 2003, pp. 131–138.

[22] A. Clark, C. Fox, and S. Lappin, *The Handbook of Computational Linguistics and Natural Language Processing*. John Wiley & Sons, 2013.

[23] J. Naradowsky and K. Toutanova, "Unsupervised bilingual morpheme segmentation and alignment with context-rich hidden semi-Markov models," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, 2011, pp. 895–904.

[24] F. Peng and D. Schuurmans, "A hierarchical EM approach to word segmentation," in *In 6th Natural Language Processing Pacific Rim Symposium (NLPRS2001) Shai Fine, Yoram Singer, and Naftali Tishby.1998*, 2001, pp. 475–480.

[25] H. Poon, C. Cherry, and K. Toutanova, "Unsupervised morphological segmentation with log-linear models," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2009, pp. 209–217.

[26] R. Christensen, *Log-Linear Models and Logistic Regression*, ser. Statistics. Springer Verlag, 1997.

[27] S. Haberman, *The Analysis of Frequency Data*. Chicago: The University of Chicago Press, 1974.

[28] B. Can and S. Manandhar, "Methods and algorithms for unsupervised learning of morphology," in *Computational Linguistics and Intelligent Text Processing*. Springer, 2014, pp. 177–205.

[29] H. Hammarström and L. Borin, "Unsupervised learning of morphology," *Comput. Linguist.*, vol. 37, no. 2, pp. 309–350, Jun. 2011.

[30] J. Goldsmith, "Unsupervised learning of the morphology of a natural language," *Comput. Linguist.*, vol. 27, no. 2, pp. 153–198, Jun. 2001.

[31] R. Tavoli, E. Kozegar, M. Shojafar, H. Soleimani, and Z. Pooranian, "Weighted pca for improving document image retrieval system based on keyword spotting accuracy," in *2013 36th International Conference on Telecommunications and Signal Processing (TSP)*, July 2013, pp. 773–777.

[32] M. Keyvanpour, R. Tavoli, and S. Mozaffari, "Document image retrieval based on keyword spotting using relevance feedback," *International Journal of Engineering-Transactions A: Basics*, vol. 27, no. 1, p. 7, 2014.

[33] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge University Press, 2008.

[34] M. Creutz and K. Lagus, "Unsupervised models for morpheme segmentation and morphology learning," *ACM Transactions on Speech and Language Processing (TSLP)*, vol. 4, no. 1, 2007.

Appendix

Table 1: Sample of index of the corpus

اسم الملف	اسم الكتاب	المؤلف	سنة الوفاة	مكان الميلاد	مكان الوفاة	عدد الكلمات	المصدر
0.1.txt	القرآن الكريم	-	-	-	-	78,245	`http://ksucorpus.ksu.edu.sa`
44.1.txt	معاني القرآن وإعرابه	أبو إسحاق الزجاج، إبراهيم بن السري بن سهل	311 AH	بغداد	بغداد	558,238	`http://ksucorpus.ksu.edu.sa`
52.4.txt	المغازي	أبو عبد الله الواقدي، محمد الأسلمي	207 AH	المدينة المنورة	بغداد	279,632	`http://shamela.ws`
198.1.txt	فضائل القرآن وتلاوته	أبو الفضل، عبد الرحمن بن الرازي	454 AH	مكة المكرمة	نيسابور	12,245	`http://shamela.ws`
198.2.txt	أحاديث في ذم الكلام وأهله	أبو الفضل، عبد الرحمن بن الرازي	454 AH	مكة المكرمة	نيسابور	80,945	`http://sourceforge.net/projects/newarabiccorpus`

Table 2: Sample of N-grams table

N-gram	Frequency
#	1,138,616
#ا	157,361
#ال	120,542
#الك	2,856
#الكت	119
#الكتا	42
#الكتاب	21
#الكتابا	4
#الكتابان	1
الكتابان$	3
لكتابان$	3
كتابان$	4
تابان$	10
ابان$	142
بان$	1,871
ان$	29,092
ن$	111,887
$	1,138,616
#	1,180,304
# u	17,387
#un	7,583
#unb	219
#unbr	29
#unbre	9
#unbrea	6
#unbreak	4
#unbreaka	2
#unbreakab	2
#unbreakabl	1
#unbreakable	1
unbreakable$	2
nbreakable$	2
breakable$	5
reakable$	5
eakable$	7
akable$	14
kable$	85
able$	2,949
ble$	4,069
le$	19,211
e$	135,377
$	1,180,304