Pattern-based Stemmer for Finding Arabic Roots

Riyad Alshalabi
Yarmouk University, Irbid, Jordan

Abstract: This study provides a technique for extracting the triliteral Arabic root for an unvocalized Arabic corpus. It provides an efficient way to remove suffixes and prefixes from the inflected words. Then it matches the resulting word with the available patterns to find the suitable one and then extracts the three letters of the root by removing all infixes in that pattern. This technique does not use any dictionary to check the resulting stem. We define some rules that help to decide if the letters belong to the root or not. This algorithm has been tested on a corpus of 72 abstracts (10,582 words) from the Saudi Arabian National Computer Conference, the accuracy of this algorithm is about 92%.

Key words: Root, stem, morphology, pattern, prefix, suffix

INTRODUCTION

The Arabic language is a highly inflected language, this increases the difficulty of the stemming process\(^\text{[1]}\). Our methodology depends on reducing the inflected word by removing all its suffixes and prefixes according to a certain methodology. When this process is done correctly; it becomes easy to find the pattern that matches this word and extract the stem characters (Fig. 1). Aljflyl and Frieder\(^\text{[2]}\) used this idea to develop a light stemmer for information retrieval applications. Here we used this general idea but with a different technique.

Removing all suffixes and prefixes from the word helps in reducing the number of patterns. It facilitates the pattern matching process and enables more variations of the stem to be conflated to the same pattern\(^\text{[3]}\).

Arabic stemming: Arabic words demonstrate an intricate morphology\(^\text{[4]}\). The Arabic language can be said to use root-and-pattern morphotactics where a pattern can be

<table>
<thead>
<tr>
<th>Arabic word</th>
<th>English meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>كتاب</td>
<td>the book</td>
</tr>
<tr>
<td>كتاب</td>
<td>like the book</td>
</tr>
<tr>
<td>كتاب</td>
<td>for the book</td>
</tr>
<tr>
<td>كتاب</td>
<td>by the book</td>
</tr>
<tr>
<td>كتاب</td>
<td>and book</td>
</tr>
</tbody>
</table>

Fig. 2: Some prefixes attached to the word book thought of as a template adhering to established grammatical rules. Such patterns are applied by adding affixes (prefixes, infixes or suffixes) to roots (which are simple bare verbs that are three letters in length) to form their parent root. Prefixes and suffixes can be further added to Arabic stems to express common grammatical usages such as the possessives, plurals, definite forms, gender, etc\(^\text{[5]}\). For example, some of the additional forms of the word ( كتاب ) “book” are shown in Fig. 2. Many characters are attached to the word ( كتاب ) while in English these additions appear as separate forms.

Stemming can be defined as the process of normalizing word variations by removing prefixes and suffixes to get the affix free word\(^\text{[6]}\). There are four kinds of stemmers\(^\text{[7]}\):

1. Manually constructed dictionaries: These are easy to use tables listing the roots and stems of words. The only problem with these is that they may not be exhaustive\(^\text{[8]}\).

Fig. 1: Extracting the stem of the word كتاب from the pattern فاعل Arabic stemming
2. Light stemmers: These remove suffixes and prefixes to discover the original stem and group words with the same parent stem. They are usually referred to as stem-based stemmers[7].

3. Morphological analyzers: These algorithms attempt to restore the original root of a word and group words accordingly[9]. These are known as root-based stemmers and are far more complex than light stemmers. Because such stemmers group terms based on their roots, terms that are not semantically related may also be grouped into an equivalence class.

4. Statistical stemmers: These stemmers attempt to group word variants using clustering techniques[10]. The different techniques vary from using letter N-gram based retrieval to using co-occurrence analysis. This process essentially involves a process of repartitioning and regrouping terms into new classes to correct the errors from earlier stemming stages. A unique advantage that statistical stemmers enjoy is that they are somewhat more language independent.

Algorithm description: In our algorithm, we have the following steps:

Step 1: Normalize corpus
Step 2: Remove the determiner “ال” (the) and its combinations from the beginning of the word.
Step 3: Check for prefixes with duplicate letters and remove the first one.
Step 4: IF the first letter is “ب” then:
   • Remove all suffixes.
   • IF CheckPrefix(“ب”)=TRUE, then remove “ب”, ELSE go to step 7
ELSE IF the first letter is “ف” then
   • Remove all suffixes.
   • IF CheckPrefix(“ف”)=TRUE, then remove “ف”, ELSE go to step 7
Step 5: IF the first letter is “د” or “ج” or “چ” then
   • Remove all suffixes.
   • IF CheckPrefix(“د” or “ج” or “چ”)=TRUE, then remove it, ELSE go to step 7
Step 6: Reduce the word
   • Remove non single letter prefixes.
   • Remove all suffixes.
   • Remove single letter prefixes (“ب” and “ب”)
Step 7: Match with pattern.
   • IF no match then return the original word and EXIT.
Step 8: Normalize root.

Normalizing the corpus: The corpus is normalized as follows:

1. Convert the first letter of the word “ات”, “اً” or “ات” into “ات”.
2. Remove vowels (except the Shaddah symbol “’”).
3. Duplicate any letter that has the Shaddah symbol.
4. Remove punctuation.
5. Remove stop words.

Removing stop words: Before finding the stem of any word, we check if the word is a stop word or not. To do this we use a list that contains most of the Arabic stop words.

Removing the determiner “ال” : The second step in the algorithm is to remove the determiner “ال” and its combinations (Fig. 3). All these characters must be removed from the word, since these letters are the leftmost prefixes that can appear in an Arabic word.

Before removing any prefix or suffix, the algorithm checks the size of the word; the number of characters remaining word length must be greater than or equal to 3. For example, the prefix “ بل” will not be removed from the word “ يلغ”. Some words have these same characters as root characters (e.g., “ تاب” , “ هلته” and “ هلته”). To stem such words correctly we check these patterns before removing their prefixes. Using this rule the word “ يلغ”, for example, will be reduced to the word “ يلغ”, as we will explain later and then return the stem “ يلغ”.

Removing prefixes: The next step is to remove all multiletter prefixes that have no duplicated letters. Figure 4 shows the duplicated letters. If these letters are found then the first one is considered a prefix and will be removed. For example, the words “كتب”, “تى” and “شغ” will be reduced to “كتب”, “تى” and “شغ”, respectively.

We check the multiletter prefixes shown in Fig. 5. In this step we do not check the single letter prefixes (“ي” and “ئ”) because these characters could be root letters and not prefixes. For example, the letters “ئ” and “ئ” in the words “ئى” and “ئى” respectively both belong to the stem. So after removing the suffixes later, the remaining word will be retained as a stem since its length is 3.

Removing suffixes: The word must be reduced to match an appropriate pattern. When the inflected word enters this step, the algorithm checks for the suffixes shown in Fig. 6 working from the longest to the shortest one. As mentioned above, the algorithm checks the length of the word before removing any suffix; the length of the remaining word must be greater than 2.
For example, let’s start with the word “في المكتبات”. In step 2 the determiner “ال” is removed, returning “في المكتبات”. No prefixes are found in step 3. In step 4 the suffix “ة” is removed, returning “في المكتبات”.

As another example the word “فسماعا” enters step 4 and its suffixes are removed starting with the longest “هما” then the shorter “نا” finally return “فسماع” as a suffix-free word.

Some conditions for suffix removal are illustrated later in this study.

Removing “ف” and “و”: These two letters have the meaning of (then) and (and) in English respectively, so they written before any single letter prefix as “في” which indicates the present form of the verb, but in Arabic they cannot be used together and still have the same meaning. So, if both of them appear, the second letter will not be a prefix. In this step we check one of them only. These letters can sometimes be root letters not prefixes, for example: “فاس,” “فاف,” “فاف,” “فاف,” “فاف,” “فاف,” etc. it is difficult to distinguish these words without using a database containing all Arabic stems. To resolve this ambiguity we use some rules that depend on patterns. If the word matches a certain pattern, then the letter will not be removed, as will be explained later.

Although this technique resolves this problem partially, it sometimes fails with some words, especially when two words reduce to the same string. For example, consider the pair of words “فول” and “_FW_W”, the letter “ف” is a prefix in the first word but not in the second one.

Now in step 4 if the first letter of the inflected word is “ف” or “و”, we remove all suffixes of the word. This helps us to match this word with certain patterns. If the match succeeds, then the character is considered to be a root letter and will not be removed from the beginning of the word.

Removing “د”, “ل” and “ب”: These letters are not used as prefixes before the prefixes “ب” or “ف”, so we check them in the next step (step 5). Here we use the approach illustrated above. In the next section we discuss the rules used to check these prefixes.

Reducing the inflected word: In this step the inflected word is completely stripped of all suffixes and prefixes, this is done after dealing with all previous affixes, but here more single letter prefixes are removed (“ف” and “و”) if they are considered as prefixes. As we described, multiletter prefixes are removed first, then all suffixes and finally single letter prefixes. For example the word “علمهم” will enter directly at step 6 and will be reduced as follows:

"علمهم"
"علمهم"
"علمهم"

Pattern matching: After removing all prefixes and suffixes of the inflected word we match it with all available patterns. If a pattern is found then we can extract the
letters that form the root, if no match found we return the
inflected word as it is.

**Patterns:** We tried to reduce the number of patterns by
increasing both the prefix and suffix lists. For example we
do not store any of these patterns:

"استعلم", "فعل", "فعل", "فعل", etc. Instead, we
remove all these prefixes and suffixes before matching the
word with its pattern. This reduces the number of patterns and
makes it easy to find the correct pattern.

In addition, we grouped patterns according to their
length, so we have patterns of length 4 as "فعل", length 5 as "فعال "، length 6 as "فعل" . The total
number of patterns with length 4 is 11, the total number of
patterns with length 5 is 25 and the total number of
patterns with length 6 is 15.

We match any word with patterns according to its
length, by using a set of conditions to check the infix
letters in the word. For example, the word "حاسيب"
has length6, so we search the patterns using the following conditions:
Find a pattern with length 6 that has
• "أ" as third letter and
• "ب" as fourth letter and
• "د" as fifth letter.

These conditions match only the pattern "فعل".
Now we remove these letters and extract the stem
"حاسيب".

The order of these rules and the conditions used for
matching are very important factors in guaranteeing a
correct matching.

**Normalized roots:** The final step in the algorithm is to
normalize the resulting stem. For example, the stem for the
word "إن" is "إن" , to normalize it we replace the
first letter by "أ" and the root becomes "أ". Another
example is to convert "أ" in the beginning of the stem to
"أ". For example the inflected word "استعمل" has the
root "عمل". After normalization it I becomes "عمل".

**Rules for removing single letter prefixes:** After removing
all multiletter prefixes and all suffixes of the inflected word
we match the resulting word with certain patterns. If a
match is found, then we consider the checked letter as a
root letter. These patterns are not the same for each prefix
(as we will show).

We encountered problems with this technique,
because for some words ambiguity resolution is not
possible without using a database containing all Arabic
stems or alternatively using a vocalized corpus. This is

because both words contain the same consonant string
To solve this problem we selected the most [appropriate?
frequent?] usable word of the pair.

For example the pair of words "فأ" and "فأ" have the
same string, the letter "أ" is a prefix in the
first word but not in the second one. Here one of them must be
selected. The formal statement of the rule is: If a word that
starts with "أ" has the pattern "فعل " then remove it,
because it is used with a verb, So, the prefix "أ" will
be removed from any word that has the pattern "فعل ".
We used this technique with all other single letter
prefixes. There are some patterns that do not have such
problems, for example the pattern "فعل "

**Rules for the prefix "ف"**
CheckPrefix(ف) = FALSE IF the word matches one of the
following patterns:

2. "أ," "أ", etc. Ex: "أ," "أ", etc.

ELSE CheckPrefix(ف) = TRUE

**Rules for the prefix "أ"**
CheckPrefix(أ) = FALSE IF the word matches one of the
following patterns:

1. "أ," "أ", etc. Ex: "أ," "أ", etc.
2. "أ," "أ", etc. Ex: "أ," "أ", etc.

ELSE CheckPrefix(أ) = TRUE

**Rules for the prefix "د"**
CheckPrefix(د) = FALSE IF the word matches one of the
following patterns:

1. "أ," "أ", etc. Ex: "أ," "أ", etc.
2. "أ," "أ", etc. Ex: "أ," "أ", etc.
5. "أ," "أ", etc. Ex: "أ," "أ", etc.

ELSE CheckPrefix(د) = TRUE

Notice that this prefix is not used with verbs, so more
patterns can be used and the number of problems
becomes smaller.

**Rules for the prefix "ب"**
We used the same patterns that are used with the prefix
"د" above. but there are some problems. For example, the
word "بلاع " is corrected reduced to "لب "， but the
word "بلاع " is also reduced, which is not correct.
Rules for the prefix “س”
This prefix is used to talk about the future, so we add "ست", "س", "سمن", and "سي" to the prefix list, of strings to be removed at the beginning of the algorithm (step 3). For example: "لاب", "أبد", "سمن", etc.

Rules for the prefix “ل”
CheckPrefix(:)=FALSE IF the word matches one of the following patterns:
1. "لاع": Ex: "لاع" "قلب" "قلب" "قلب"
2. "لع": Ex: "لعب" "لعب" "لعب"
3. "لح": Ex: "لح" "لح" "لح"
ELSE CheckPrefix(:)=TRUE

Rules for the prefixes “ت” and “ي”
These two letters are checked in step 5, so if the remaining strings of length 3 then we are guaranteed that one of these two letters belongs to the root.
CheckPrefix(:)=FALSE IF the word matches one of the following patterns:
1. "تاجر": Ex: "تاجر" "تاجر" "تاجر"
2. "تقال": Ex: "تقال" "تقال" "تقال"
ELSE CheckPrefix(:)=TRUE

Suffix removal constraints: Some suffixes cannot be removed from the inflected word, because they are considered to be part of the root, so we need to add some exceptions to the rules for removing suffixes. For example, the inflected word “تسي” (according to the algorithm) will be reduced to “تسي”, which is wrong, because the letter “ي” is a single letter suffix (as in “كتني”). To deal with such problems we defined some constraints on such suffixes:

6. Do not remove “ت” from the pattern “فعل” "فعل" "فعل". Ex: " فعل " " فعل " " فعل " .
7. Do not remove “و” from the pattern “تجر” "تجر" "تجر". Ex: "تجر" "تجر" "تجر".

General problems: Problems may arise with some inflected words, since an original letter(s) may be removed or match a word with non-suitable pattern. These problems arise as a result of the similarity of strings of some words.

Examples
1. Words such as the inflected word “يفهم”. Here the last two letters will be removed, because the algorithm recognizes “ه فهم” as a suffix.
2. The algorithm cannot distinguish between the two words “تقول” (that matches the pattern “تقول” “تقول”) and “تقول” “تقول” (that matches the pattern “تقول” “تقول”). As we described above we match “تقول” correctly because it is more common.
3. The pair of words “مشتري” and “مشتري”. The letter “ي” belongs to the root in the first word but not in the second one. The problem here lies with the first word “مشتري”.
4. The pair of words “اللهم” and “اللهم”, have the same problem for pattern matching. The first word must match the pattern “قال” ”, while the second word must match the pattern “قال” “قال”. The problem lies with the word “قال” “قال”.

Examples
The inflected word “قالبة” enters the algorithm at step 6, no multi character prefixes are found, so the suffix list will be checked and the suffix “ت” will be removed, only a three letter root will remain (كوب). Also the word “سمعها” will enter at step 6 and the suffix removal process will start as follows:

1. سميت
2. سميت

Then the prefix “ي” will be removed (not a root character). Now if we have the inflected word “ويليفارقا”, it will enter step 4:

1. remove suffix as follows:
2. check(و) and remove it
Enter step 5:
1. Remove suffix: no match.
2. Check (.) and remove it
Enter step 6: (reduce word): remove single letter prefix.
Enter step 7: match “فرق” with patterns and return the root “فرق”.
The inflected word "استعمالات" is stemmed as follows:
Enter step 6:
1. Remove prefix, resulting word is "استعمالات".
2. Remove suffixes, resulting word is "استعمال".
3. No single letter prefix found.
Enter step 7: match it against the pattern "فعال" and extract "فعال".
Enter step 8: convert (ة) to (ت). Another example, the inflected word "والملاحظات" Enter step 2: remove "وال".
Enter step 6 as follows:

Enter step 7 and return نتائج
The word can be stemmed as follows:

CONCLUSIONS

Morphological analysis is the first step in most natural language processing applications. We have developed a new algorithm that runs an order of magnitude faster than other algorithms in the literature. This study provides an efficient technique for extracting the triliteral root for an unvocalized Arabic corpus. This technique does not depend on searching, since we do not store any Arabic stems. It depends on suffix removal, prefix removal and pattern matching. The algorithm has been implemented using Visual Basic 6.0. We tested our algorithm using a corpus of 72 abstracts (10582 words) from the Saudi Arabian National Computer Conference. The algorithm performs very well and the accuracy is approximately 92%.

REFERENCES