Unsupervised Morphemes Segmentation

Khalid ur Rehman
School of Computing
University of Leeds
Leeds, LS2 9JT
scs5kur@leeds.ac.uk

Iftikhar Hussain
School of Computing
University of Leeds
Leeds, LS2 9JT
scs5lh@leeds.ac.uk

Abstract

In this work, we describe the algorithm adopted to split the words into smallest possible meaningful units or morphemes. The algorithm is unsupervised and not dependent on any language. The model is developed using English language. However, the linguistic rules specific to English language are not implemented. The algorithm focuses on the identification of smallest units of words based on their frequency of occurrence in a given text corpus. The model works in two stages. In first stage the model learns from a given text corpus and makes a list of possible morphemes. In the second stage the model divides the words into possible meaningful segments. There is no predefined list of morphemes attached or hard-coded in the model.

1 Introduction

Generally, words are considered as the most basic unit of any language. However, this assumption is not true. In fact, words are the means to communicate in a language and their use vary with time and location. For example, there are words in English language that are spoken in UK but not in USA and vice versa. Similarly, the words in old English poetry are no more in use. Another interesting fact is that the numbers of words in any language are not fixed. According to Dr. Goodword's Separation hypothesis, "*there are no such things as words*" [1]. He claims that "*what we take as words are in fact two distinct phenomena lexemes and morphemes. Lexemes are noun, verb, and adjective …….. Morphemes are everything else, including suffixes like -y, -ness, -er, -ing, -ly and prefixes like re-, un-, anti"*[1]. Lexemes refer to things in real world whereas the morphemes only refer to the grammatical categories.

A morpheme is the smallest meaningful unit in the grammar of a language [1]. There are two basic types of morphemes: roots and affixes [2]. Roots make the main part of the word and repeat only once in a word. On the other hand, affixes are the subordinate parts that may or may not exist in a word. Affixes either precede or follow their root.

2 Assumptions

Following assumptions are made in order to reduce the complexity and improve the efficiency of algorithm.
The special characters (like -, /, ‘ etc.) are treated as the word separators.
The maximum length of a suffix is limited to three characters and maximum length of a root morpheme is limited to 13 characters.
After separating affixes (prefix and suffix) remaining word will be considered as root morpheme and its length must be no less than five characters for further division. As shown in example below.

EXAMPLE:
Actual Word: uneducated
Divided word: un……educate……d
Prefix – un
Root morpheme – educate
Suffix – d

3 Model

The learning model learns from the corpus and prepares a list of possible segments based on their frequency. After the learning is done the

words from the corpus are picked up one by one and segmented into possible morphemes.

3.1 Learning Model

This is implemented in Microsoft Access. It takes the most frequent words from the given corpus for identification of possible morphemes. At first, the model extracts the list of words from the corpus ignoring the words having frequency less than seven. The model was also tested by adding lower frequency words. However, it was not affecting the segmentation results but was making the segmentation process slower. Therefore to reduce the complexity the lower limit of frequency was set to seven.

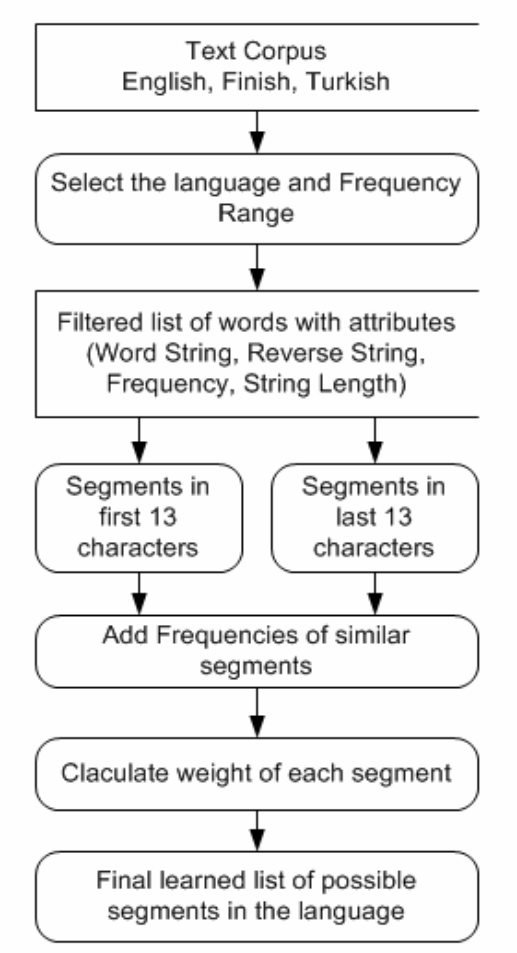

Figure 1. Learning Model

Segments (affixes, root morphemes) are searched from within first and last thirteen characters of a word. The maximum length of affixes in English is mostly shorter than four characters however in our model maximum limit is set to thirteen characters. This helps in separating affixes and root morphemes having a maximum length of thirteen characters. The process of finding possible segments in a word works on two sets of thirteen characters (leading 13 characters and trailing 13 characters). See Fig.1.

Before executing the algorithm, the list is sorted using dictionary sorting. Now to understand the algorithm execution let's take an example of six words to be segmented.

- ab
- abacus
- about
- abreast
- again
- bargain

From the above list, model will pick the first character from first word (i.e. 'a') and will check its frequency in the remaining words. The frequency of 'a' is five, now if 'a' also exists as a single word then it will be qualified as a valid segment. However, in the list there is no complete word as 'a' therefore it will be ignored. In the next step model will pick up 'ab' and then check its frequency in the list. The frequency of 'ab' is four. Now as 'ab' exists in the list as a word therefore it will be qualified as a valid segment and will be added to the learned list. This process will continue till maximum thirteen characters of a word (if the length of word is thirteen characters or more).

In order to find segment from trailing side of word, similar procedure will be followed starting from last character.

If a similar segment is found in both processes mentioned above then their frequency will be added and the segment will be included in the learned list only once. See Fig.1.

The last step calculates the weights of different segments. In any corpus single characters have maximum frequencies. Like in our learned list "t, c, g, r and n" occur more than 5000 times however these characters may not be valid segments. Therefore, in order to ignore these characters during segmentation process their weights are calculated by subtracting the standard deviation from their respective frequencies. For example, "t" occurs 5408 times and the standard deviation of frequencies of all single character segments is 3614. This makes the weight of "t" 1794. Similarly frequency of "h" is 2908 but its calculated weight is -705 (2908-3614=-705). As the weight is negative, therefore the segmentation model will not consider "h" as a valid segment.

3.2 Segmentation Model

The segmentation portion of the model is developed in Visual Basic 6.0. The segmentation process pursues the following steps:

- Separation of prefix from the word
- Separation of suffix from the word
- Segmentation of root morpheme (if the word length is more than five character)

This model takes a word from corpus and compares it with the learned list of segments prepared during the execution of learning model.

The segmentation model creates a model list of all words that have been segmented. During the process of segmentation, this list is continuously updated.

As the segmentation model receives a word for segmentation it is broken into parts depending upon the existence of assumed, word separation characters (like -, / , ‘ etc). Both the character strings before and after separation character are treated as independent words. However in this case the word before the separation character is not evaluated for the suffix and the word after the separation character is not evaluated for prefix.

The segmentation is done in two phases. First phase checks each segment of characters starting from the first character till the last character. If any segment of character/s is found in the list and the remaining segment is also found in the list then the separated segment is treated as a possible prefix. At this stage if the segment is of two character length or less its weight is assessed. If the weight is negative, the segment will be disregarded. This process continues until a valid prefix found.

The remaining string (after removing prefix) is passed to the suffix separation module. This module starts from the first trailing character and goes till maximum segmentation of three trailing characters. It could result in more than one suffix. The one with high weight will be considered as valid suffix.

At this stage the remaining root segment is passed to the prefix separation module to separate any possible root morphemes. For root morpheme segmentation a remaining word must have at least five characters. As per the assumption the words of five characters and less are treated as single root morpheme. This assumption is made because till this stage valid prefix and suffix are already separated.

If the prefix separation module fails to segment a word then the word will not be passed to the suffix separation module. It will be passed to the second phase of the model.

The second phase separates first trailing character of the word and then passes the remaining segment to the prefix separation module. Now the prefix separation module repeats the process with one trailing character trimmed. The phase two continues to trim the trailing characters and keep on passing the remaining segment to the prefix separation module till the time the prefix separation module can find any valid segment. The purpose of this module is to pick those words, which cannot be separated by prefix separation module during the first phase. The integration of second phase helps in segmenting the words where two valid segments do not exist in the learned list. For example if we take the word “controlled” the prefix separation module will start from the first character ‘c’ and the first valid morpheme boundary will be after ‘l’, which will make a valid segment “control”. However, there is a possibility that the learned list does not have the remaining segment “led”. Therefore, the programme will not split the word “control led”. Similarly as the word ‘controll’ does not exist in the learned list therefore the other possible segmentation “controll ed” will also not take place. Combination of two phases of model ensures that maximum possible morpheme boundaries are detected.

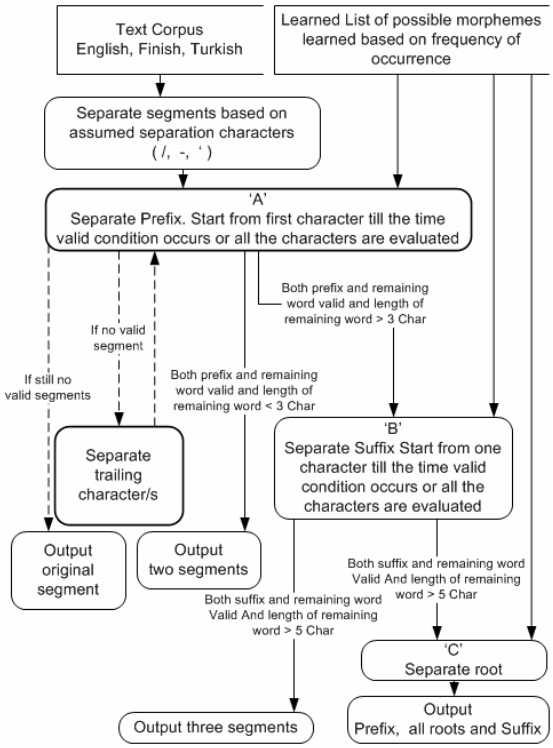

Figure 2. Segmentation Model

4 Evaluation

The evaluation is done by using the Perl script given on the Morpho Challenge website [3]. The script compares the segmented lists with the gold standard lists given for each language. The evaluation is based on three possibilities mentioned below.

- **Hit.** A valid cut that means word is cut at the right place.
- **Insertion** A wrong cut that means word is cut at the wrong place.
- **Deletion** A missed cut that means a valid cut is ignored.
- Following three parameters are calculated based on these possible cuts.
- **Precision** It is the number of hits divided by the sum of the number of hits and insertions.
- **Recall** It is the number of hits divided by the sum of the number of hits and deletions.
- **F-measure** It is the harmonic mean of precision and recall. As per the gold standard the results having higher value of F-Measure are considered as better segmentation results.

The model was run using English, Turkish and Finnish word lists. The results achieved are as follows:-

Morpheme Boundary Detections Statistics

	English	Turkish	Finnish
F-measure	56.68%	44.38%	43.46%
Precision	53.36%	59.46%	67.18%
Recall	60.46%	35.39%	32.12%

Table 1: Morpheme Boundary Detections Statistics

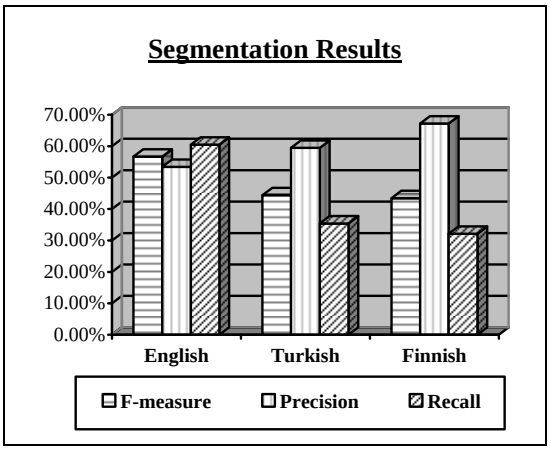

Figure 3. Segmentation Results

- F-measure is maximum in English language. However, it is almost same in Turkish and Finnish.
- Precision is least in English and it increases in Turkish and Finnish.
- Recall is maximum in English and reduces considerably in Turkish and Finnish.

If we plot the line graph of precision and recall then it shows reciprocal behaviors.

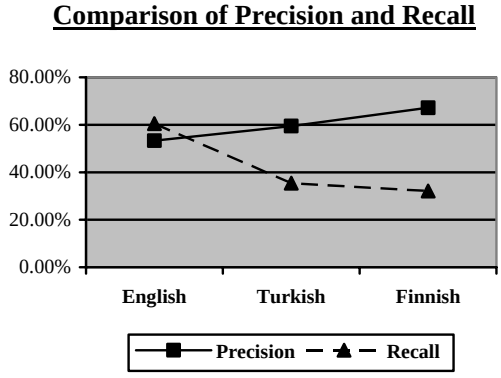

Figure 4. Comparison of Precision and Recall

The proposed model detects the morpheme boundaries based on the frequency of various segments in a given corpus. The results show that possibility of ignoring a valid cut is more than putting a wrong cut in Turkish and Finnish language; however it is opposite in English. As the model always compares both segments for validity, therefore at some occasions a valid morpheme boundary may be ignored. For example if we consider the word 'stopped' for segmentation it may not identify any valid segments. Like if 'stop' is identified as a valid segment then the remaining 'ped' may not be a valid segment. Similarly while the segmentation is done in reverse order 'ed' may be recognized as a valid segment, however 'stopp' may not be found in the learned list. Under such circumstances the second phase of the model helps to cut the trailing character/s till valid segments are found in the initial set of characters. In this case "stopped" will be segmented as "stop p ed" (here weight for p is 654). This approach helped in avoiding wrong cuts because of which the precision is high in Turkish and Finnish corpus.

The high recall and low precision in English shows that there are less ignored cuts as compared to wrong cuts. The wrong cuts are because our model finds more segments in longer words. Like if we take the example of word "unconstrainedly". Our learner model has calculated the weight for 's' and 't' as '13999' and '1794' re-

spectively. Therefore the segmentation of the word would be ‘un con s t rained ly’. This has resulted in more wrong cuts in longer words, especially in English language.

The assumptions made at the beginning make the model a bit specific to English. The separation characters helped the model in identifying the morpheme boundaries. The limit on the length of word to be assessed for segmentation of root morpheme, which is set to five, is also based on English language knowledge. As the model is developed by learning from English language corpus, therefore it has resulted in better identification of morpheme boundaries in English language.

The lower **recall** in Turkish and Finnish language has adversely affected the value of F-Measure in these languages. However, the value of **F-measure** for these languages is almost same. This shows that the effect of assumptions on both languages (Turkish and Finnish) is same.

Reference

http://www.alphadictionary.com/articles/ drgw004.html, “How many words are in English? – alphaDictionary”, 14-01-06

http://www.ruf.rice.edu/~kemmer/Words/ rootaffix.html, “Words in English: Roots and Affixes, 14-01-06

http://www.cis.hut.fi/morphochallenge2005/, “Unsupervised segmentation of Words into morphemes challenge 2005”, 14-01-06

Microsoft Corporation-Online MSDN, http://msdn.microsoft.com/vbasic/, visual Basic Developer Centre, 14-01-06

Mathias Creutz, “Unsupervised Segmentation of Words Using Prior Distribution of Morph Length and Frequency”, Neural Network Research Centre, Helsinki University of Technology, Finland.