Simple Unsupervised Morphology Analysis Algorithm (SUMAA)

Minh Thang Dang
School of Computing
University of Leeds
Leeds LS2 9JT, UK
thangdang@hotmail.co.uk

Saad Choudri
School of Computing
University of Leeds
Leeds LS2 9JT, UK
saad.choudri@gmail.com

Abstract

SUMAA is a hybrid algorithm based on letter successor varieties for an entirely unsupervised morphological analysis. Using language pattern and structural recognition it works well on both isolated and agglutinative languages. This paper gives a detailed analysis of how we developed SUMAA. F-Measures (MorphoChallenge, 2005) achieved by SUMAA for the English, Finnish and Turkish datasets were 51.83%, 60.18% and 55.94% respectively.

1 Introduction

Unsupervised automated word segmentation is required for morphological analysis to replace human intervention, with the primary goal to determine the location of breaks between morphemes within words. In this paper we describe an algorithm that segments words into morphemes on an unsupervised basis, i.e. with no prior knowledge (e.g. a dictionary) of the corpus under consideration. The algorithm is applied to both agglutinative languages, where words are composed of fused morphemes denoting their syntactic meanings, and isolating languages where a majority of morphemes are considered to be full fledged words (Wikipedia, 2005). In the provided corpora, Turkish and Finnish represent the former and English represents the latter (Wikipedia, 2005). A generic characteristic between the three is that of varying forms of metaphonics, vowel harmony, unlike e.g. Cantonese, which we think avoids "overfitting" (Mitchell, 1997). The corpora and evaluation script are available on the MorphoChallenge 2005 official web page (http://www.cis.hut.fi/morphochallenge2005/) which compare our results to a gold standard, or "desired" result. Evaluation measures used are Precision, Recall and F-Measure. Precision is a calculation of the number of correct cuts made against the total cuts made, Recall is the total number of correct cuts made against the total possible boundaries and the F-Measure is a harmonic mean of the two (MorphoChallenge, 2005). The F-Measure's we obtained for testing our algorithm on English, Finnish and Turkish were 51.83%, 60.18%and 55.94% respectively. Section 2 explains the "letter successor varieties" approach (Hafer and Weiss, 1974) on which our algorithm is based. Section 3 describes our most relevant experiments leading up to our algorithm. The morpheme boundary statistics are visualised for all our experiments on English in figure 3 and our preferred experiments on all three languages in figure 5. Section 3.4 and 3.5 describe our final algorithm and data structure respectively. The pseudo code is in Table 1 located at the end of section 3.

2 Letter Successor Varieties

Hafer and Weiss (1974) suggested a method called "letter successor varieties" for segmenting lexical text into stems and affixes based on Z.S Harris's solution to the problem of morpheme discovery for phonetic

Test word: READABLE

Corpus: READABLE, ABLE, APE, BEATABLE, FIXABLE, READ, READING, READS, RED, ROPE, RIPE

Successor Frequency			Predecessor Variety		
Prefix	Successor variety		Suffix	Predecessor variety	
R	3	E, O, I	E	2	L, P
RE	2	A, D	LE	1	B
REA	1	D	BLE	1	A
READ	3*	A, I, S	ABLE	3*	D, T, X
READA	1	B	DABLE	1	A
READAB	1	L	ADABLE	1	E
READABL	1	E	EADABLE	1	R
READABLE	1*	blank	READABLE	1*	blank

Figure 1. Letter successor varieties recreated from Hafer and Weiss (1974).

text. The method uses statistical properties, successor and predecessor variety counts, of a corpus to indicate where words should be divided (Hafer and Weiss, 1974).

2.1 Successor varieties and predecessor varieties

The successor frequency (we use the term frequency and variety interchangeably), defined as W [1...n], of the n^{th} letter of a word is the total number of distinct letters occurring at the $n+1^{st}$ position in the words of a corpus that match this set of letters from the selected word. Figure 1 illustrates this, and has been adapted from Hafer and Weiss's paper. In the example, "READABLE" is the "test" word in the corpus consisting of 11 words. If n=1 then the prefix is "R" and comparing "R" to the rest of the words gives a total of 3 distinct letters "E", "O" and "I" occurring at position n+1. Hence, the successor variety is 3. The same is repeated for W [1...2] until n reaches the end of the word. The results are shown. The predecessor variety is a similar concept but with the reverse of the test word, e.g. "ELBADEAR", and the reverse of the corpus (Hafer and Weiss, 1974). This is also shown in figure 1 under the heading of "Predecessor Variety".

2.2 Experimental design

Hafer and Weiss (1974) proposed four basic segmentation strategies that use the statistical method mentioned, viz. cut-off, peak and plateau, complete word and entropy. Our algorithm is based on the peak and plateau design. Take S_n to be the successor count S of the position n in a word W. A cut is made in W after a prefix denoted by the successor count S_n forms a local peak or a plateau of the count vector. The same is applied with predecessor count. In the example discussed in section 2.1, the predecessor count and the successor count both suggest that "READ" and "ABLE" are affixes. Figure 1's 2^{nd} and 3^{rd} columns show the "*" at "3" marking the "peak" of the counts found for both varieties.

3 Our experiments leading up to our proposed algorithm

This section describes only a few of our experiments that have lead to our final solution which is described in section 3.4. Their results are visualised in figure 3.

3.1 Successor frequency at peak/plateau (SF)

By applying only the successor variety counts the results obtained for English were F-Measure 37.95%, Precision 43.00% and Recall 33.97%. This method appeared to split words at the beginning as "peaks" tend to occur there and occasionally ended up with splits such as "cre dit ing" for "crediting" and "cre wmen" for "crewmen".

3.2 Predecessor frequency at peak/plateau (PF)

We then tried the "predecessor frequency at peak/plateau" a reverse of the above method. The results improved to an F-Measure of 41.43%, Precision of 41.67% and Recall of 41.20% for English. This is because the words in the corpus are more heavily suffixed than prefixed. In this case "crediting" was split as "credit ing".

3.3 Successor and predecessor frequency at peak/plateau

We realised that the words not segmented by method 3.1 were segmented correctly by method 3.2. Thus we tried a combination of the two. First, we segmented the words using SF and then for the words that had not been segmented, we segmented them using PF. The improved results for English were now 44.36%, 42.81% and 46.04% for F-Measure, Precision and Recall respectively. We applied this to Turkish and got an F-Measure of 53.24%, Precision of 60.45% and Recall of 47.57%. With Finnish we got an F-Measure of 58.03%, Precision of 63.88% and Recall of 53.16%. Trying a reverse of the combination, i.e. PF first and SF second, did not give good results. For English we got 42.48%, 40.27% and 44.94% for F-Measure, Precision and Recall respectively.

3.4 Our proposed algorithm, SUMAA

So far the first combination of SF and PF (SFPF) has shown the best results. However in the result files, splits like "abandonedly" as "abandon edly" and "acceptances" as "accept ances" were noticed. Analysing the corpora we saw that if one word was a substring of a word below it, it was often a morpheme of that word. We applied this concept to our algorithm. This procedure resembles the bubble sort algorithm except after the comparison the words are not sorted and remain in their original positions. This is illustrated in figure 2. Consider an extraction of the English corpus in the following order: aaa, abandon, abandoned and abandonedly. The steps are as below:

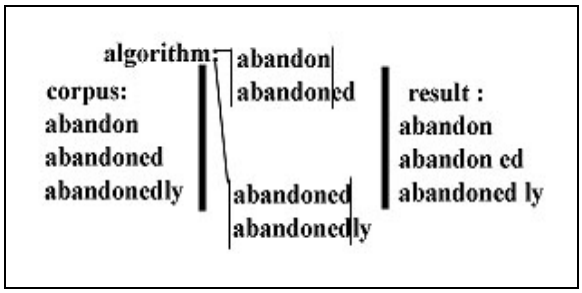

Figure 2. Bubble sort string boundary finder example.

1. Read "aaa". As it has no preceding word, apply SFPF & segment it .Print to file.
2. Read "abandon". Check if it has the preceding string "aaa". It doesn't, so segment it with SFPF and print to file.
3. Read "abandoned". It contains its preceding word, so first segment it into "abandon" and "ed" and then apply SFPF to the left side split ("abandon"). Print to file.
4. Read "abandonedly". It contains its preceding word, so segment it into "abandoned" and "ly" and then apply SFPF to the left side split ("abandoned"). In this case, "abandoned" segmented by SFPF resulted in "abandon" and "ed". Thus the word is finally segmented into "abandon", "ed" and "ly". Print to file.

The results were F-Measure 51.83%, Precision 48.06% and Recall 56.23% for English, F-Measure 55.94%, Precision 59.39% and Recall 52.87% for Turkish; and for Finnish they were F-Measure 60.18%, Precision

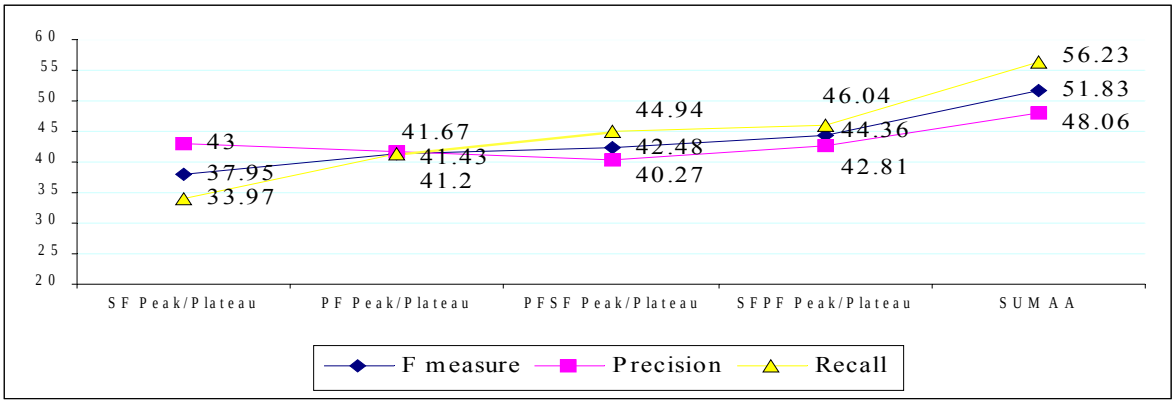

Figure 3. Results, section 3.1, 3.2, 3.3 both combos and 3.4 for tests on English.

64.97% and Recall 56.04%. As the MorphoChallenge requires a high F-Measure, this experiment is very fruitful (shown in figure 5). This new algorithm performed surprisingly well on Turkish and Finnish. It performed better for Finnish than it did for Turkish and not to mention for English! Pseudo code for SUMAA is in table 1. SUMAA seems to have worked particularly well on Finnish with an F-Measure of 60.18% and Precision of 64.97%.

3.5 Data Structure

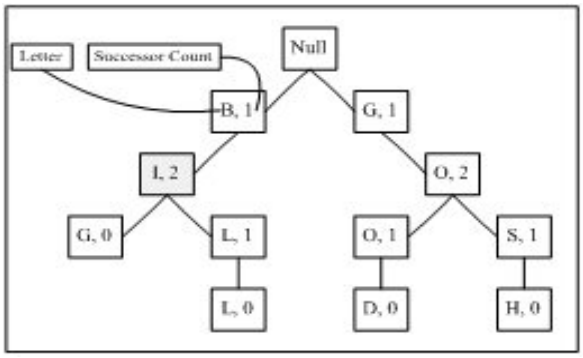

Figure 4. The trie constructed from the words: BIG, BILL, GOOD, GOSH

For letter successor varieties, words of a corpus have been represented as a data structure called a trie (please refer to (Huynh et al.) for more details) as shown in figure 4. Each node represents a letter and contains a successor count. The root represents a null string and each branch represents a word. Consider the following words: "BIG"," "BILL", "GOOD", "GOSH".

Let's take "BIG" as the test word. To calculate the successor count of the prefixes of "BIG", we traverse from the root and then retrieve the successor counts of "B", "BI" and "BIG" which are 1, 2, and 0 respectively. By organising all words and their reverse words in a corpus in this form, we can efficiently retrieve the SFPF counts of any prefixes or suffixes of a word. In our algorithm we implemented 2 tries, one for retrieving successor counts and one for retrieving predecessor counts. The trie for retrieving predecessor counts was built from the reverse words in the corpora. In doing this, it took less than 3 minutes to segment the whole corpus of Finnish (MorphoChallenge, 2005) which includes 1,636,336 word types on a computer using an AMD 2800+ CPU with 512MB RAM.

4 Conclusion

This paper is a summary of our detailed research which includes experiments with a version of MDL using a codebook, and other versions of the letter successor varieties such as "cut-off". We abandoned the idea of using the former as it took too much time due to recursive string comparisons and on a standard current day PC, it would have taken 4 days to compute the Finnish corpus as opposed to this algorithm that takes under 3 minutes. The "cut-off" experiment was not used, although it was efficient, as there was a fear that it would cause overfitting problems for Finnish as it used predefined values.

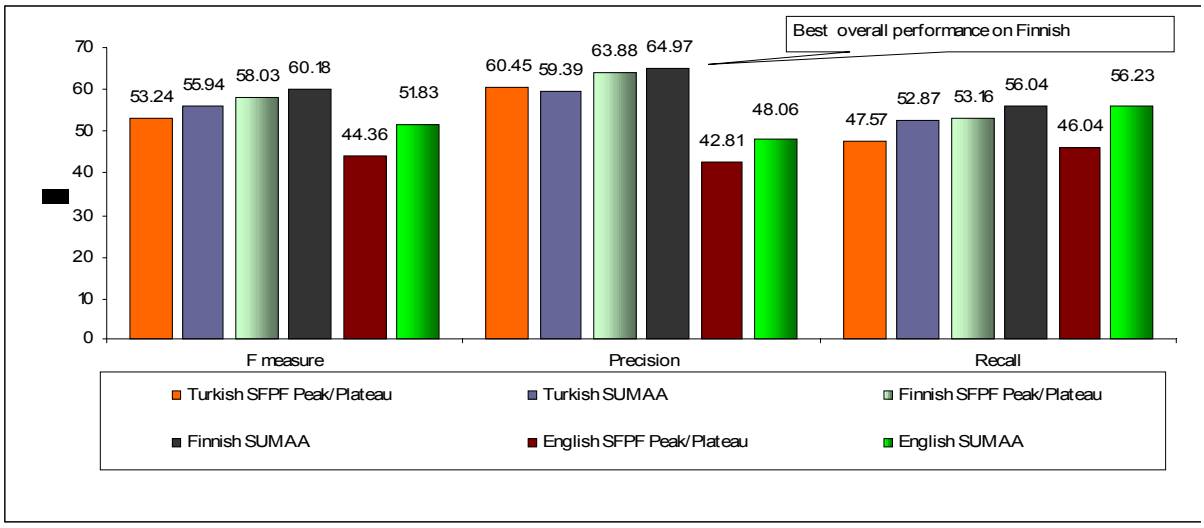

Figure 5. Results for Turkish, Finnish and English with SFPFPeak/Plateau & SUMAA.

```
Main()
{
   BuildTries(); // Build 2 tries,
                 // one from the words as they are and
                  // one from them reversed.
   For each word in the corpus
   {
       If the word contains its preceding word
       {
            Segment the word into 2 parts using the
            boundary of the shorter word;
            SegmentUsingSuccessor(left part);
            Print the right part to file at the end of the
             current line;
       } Else
       {
            SegmentUsingSuccessor(word);
       }
   }
}
SegmentUsingSuccessor(word)
{
    For each substring S of the word
    {
        Calculate the successor count S_n;
        If found a local peak/plateau
            Save this position to an array of split
             points;
    }
    If the array of split points empty  // No split point
                                        // found
        //Try to segment the word

        //using Predecessor frequency

        SegmentUsingPredecessor(word);
    Else,
       Use the array of split points to segment the
       word and print to file;
}
SegmentUsingPredecessor(word)
{
    Reverse the word;
    For each substring S of the reverse word
    {
        Calculate the predecessor count P_n;
        If found a local peak/plateau
            Save this position to an array of split
             points;
    }
    If the array of split points empty  // No split point
                                        // found
        Print the whole word to file;
    Else,

       Use the array of split points to segment the word
       and print to file;
}
```

Table 1. SUMAA pseudo-code.

It performed well over Turkish but not as well as SUMAA.

SUMAA detected a majority of both true and incorrect boundaries for English with respect to the other languages. This may be because the English corpus contained multi-language words such as "abbotabad" (where abad means population in Urdu), a city in Pakistan. Although it did segment "abbot" and "abad" correctly there were problems with other such noise in the data, which could have lead to the F-Measure and Precision drop. Modelled on English, which is considered to be an isolated language, SUMAA performs better on agglutinative languages and hence our algorithm is robust against overfitting. Figure 5 shows a comparison of SFPF and SUMAA (section 3.3 and 3.4) on all three languages as those obtained the highest results. We propose SUMAA as a very useful and efficient morphological analysis system.

References

David Huynh, David Breton and Evelyne Robidoux. Tries and suffix trees. *Winter 1997 Class Notes.* School of Computer Science, McGill University.

John Elliott and Eric Atwell. 2000. Is anybody out there? The detection of Intelligent and Generic Language-Like Features. *JBIS,* 53:13-22.

John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computer Linguistics,* 27:153-198.

Mathias Creutz and Krista Lagus. 2005. Unsupervised morpheme segmentation and morphology induction from text corpora using morfessor 1.0. *Publications in Computer and Information Science, Report A81, Helsinki University of Technology*. Finland.

Margaret A. Hafer and Stephen S. Weiss. 1974. Word segmentation by letter successor varieties. *Information Storage & Retrieval,* 10:371-385.

MorphoChallenge. 2005. EU Network of Excellence PASCAL Challenge Program. http://www.cis.hut.fi/morphochallenge2005/.

Tom M. Mitchell. 1997. *Machine learning.* New York; London, McGraw-Hill.

Wikipedia. 2005. The Free Encyclopedia. http://www.wikipedia.org/.