Bangla Morphological Analyzer using Finite Automata: ISI @FIRE MET 2012

Apurbalal Senapati, Utpal Garain
CVPR Unit; Indian Statistical Institute; 203, B.T.Road; Kolkata – 700108
apurbalal.senapati@gmail.com; utpal@isical.ac.in

Abstract

This paper describes a finite automata based morphological analyzer for Bangla. Based on the MET [1] requirement the analyzer outputs only the root (surface) word but the system has capability to produce full-fledged morphological information. The method can be used for any agglutinative language with minor changes.

Introduction

The Morphological Analyzer [2] plays a significant role in NLP (natural language processing) applications namely in machine translation, question-answering, information retrieval, spell checker, etc. Most of the Indian Languages are agglutinative and nature and degree of inflections varies from language to language. Therefore, development of a morphological analyzer for most of the Indian languages is viewed as a very complex and challenging task [3]. The input of the morphological analyzer is a word and output is all the morphemes and their grammatical categories associated with the word. Based on the MET requirement, in this work, we only concentrate to find the root (surface) word.

Existing Approaches

There are many approaches which are widely used. A brief description of commonly used approaches is as follow:

Corpus Based Approach: This approach is statistical in nature. A large corpus used as training data. Suitable machine learning algorithm is used to train the system and collect the necessary information and features from the corpus. The collected information is used to test the data. The main difficulty of this approach is to build an annotated corpus.

Paradigm Based Approach: For a particular language, each word category like nouns, verbs, adjectives, adverbs and postpositions will be classified into certain types of paradigms. Based on their morphophonemic behavior, a paradigm based morphological compiler program is used to develop the morphological analyzer.

Finite State Automata (FSA) Based Approach: Uses regular expressions and is used to accept or reject a string in a given language. In general, an FSA is used to study the behavior of a system composing of states, transitions and actions. When FSA starts working, it will be in the initial stage and if the automation is in any one of the final states it accepts its input and stops working.

Two- Level Morphology Based Approach: In 1983, Kimmo Koskenniemi, a Finnish computer scientist developed a general computational model for word-form recognition and generation called Two-level morphology [3]. This development was one of the major breakthroughs in the field of morphological parsing, which is based on morphotactics and morphophonemics concepts. The "two-level" morphological approach consists of two levels called lexical and surface form and a word is represented as a direct, letter-for-letter correspondence between these forms. The Two-level morphology approach is based on the following three ideas:

- Rules are symbol-to-symbol constraints those are applied in parallel, not sequentially like rewrite rules.
- The constraints can refer to the lexical context, to the surface context, or to both contexts at the same time.
- Lexical lookup and morphological analysis are performed in tandem.

Stemmer Based Approach: Stemmer uses a set of rules containing list of stems and replacement rules to stripping of affixes. It is a program oriented approach where developer has to specify all possible affixes with replacement rules. Potter algorithm is one of the most widely used stemmer algorithm and it is freely available. The advantage of stemmer algorithm is that it is very suitable to highly agglutinative languages like Dravidian languages for creating the morphological analyzer.

Suffix Stripping Based Approach: Very useful in highly agglutinative languages such as Dravidian languages. Here advantage is that, in many cases the words are usually formed by adding suffixes to the root word. This property can be well suited for suffix stripping based approach. Once the suffix is identified, the stem of the whole word can be obtained by removing that suffix and applying proper orthographic (*sandhi*) rules.

Linguistic Study and Literature Survey

Based on our linguistic study and literature survey we see that Bangla is a highly agglutinative language. We have found that the rules of inflections are based mainly on grammatical, like nominal, pronominal, verbal, animate, inanimate, human, etc. and some cases they do not follow any rule i.e. irregular basis. Based on our observation we broadly classify all the words into three categories:

Category 1: This category refers to the class of words where a word itself is the root (surface), i.e. the word has not been inflected.

Examples: অক্টোবর; সফটওয়ার; ফেরত; নিন্দা etc.

Category 2: This category is defined such that the root (surface) word is extracted without using any rule. This refer to irregular basis i.e. no specific rules exist.

Examples: আমাকে => আমি; আমরা => আমি etc.

Category 3: Define this category such that the root (surface) word is extracted using a specific rule (i.e. trimming the suffix/prefix).

Examples: সংবাদদাতা => সংবাদ + দাতা

From our study we have emphasized more on the last category, because the words in this category are the most important in morphological analyzer. From this category we have found a suffix list containing 173 suffixes identified still now. The sample list is {দের,দেরকেই,টে,এর,এরই,এরও,ও,কে,কেই,কেও,খানা,খানাই,খানি,গাছা,গাছি,গুলো,…}

Our Approach

Our main approach is a rule based approach implement through a finite automata. In the above sections we have classified all the words into three categories. For the computational aspect we handled each category in different manner. For the **Category1**, we have built (manually) a DICTIONARY i.e. collection of root words. Figure 1 show the architecture of our system. Given a word, we first search in the DICTIONARY and if it is found then the input word is the root otherwise go to the next step. For the **Category2**, we have built (manually) a MAP (looks like আমাকে => আমি; আমরা => আমি; ওঁর => ও; etc.). The input word is searched in the MAP and if found, we take its corresponding map value (i.e. MAP value of আমাকে is আমি) as the root word. For the **Category3**, we have developed a tool using FINITE AUTOMATA (specially the principle of Nondeterministic Finite Automata) using a SUFFIX LIST (defined in previous section).

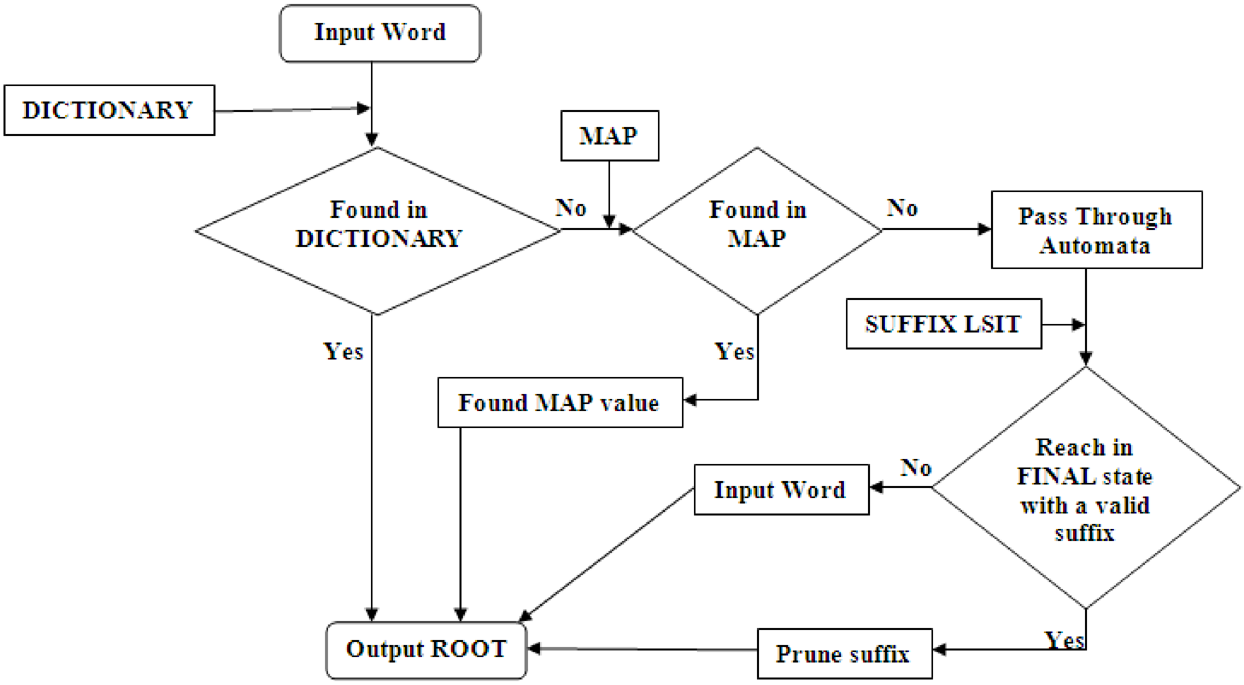

Figure 1: Architecture of our system.

We know the working principle of finite automata. In our construction we define the input symbols as {অ, আ, ই, ……, ঔ, ক, খ, গ, …., া, ি, ী, ু, ূ, ……, ৌ} and we have define the final state when the automata consume a valid suffix (suffix in our SUFFIX LIST). The following example illustrates the computational procedure.

Example: input word is ‘সংবাদদাতা’

From the computational point of view first we reverse the word

i.e. **Step1**: ***reverse***(সংবাদদাতা) => াতাদদাবংস
Next we split the reversed string character wise i.e. **Step2** াতাদদাবংস => া ত া দ দ া ব ং স
Next we pass that string/াতাদদাবংস (symbols i.e. া ত া দ দ া ব ং স) through the Automata and since the first symbol is 'া' hence it chooses the appropriate path shown in Figure 2. Since 'দাতা' is the suffix (present in our SUFFIX LIST) and hence after consuming the characters 'া', 'ত', 'া', 'দ', it will enter in a final state and hence we prune the suffix 'াতাদ'. **Step3:** the result is 'দাবংস' (Shown in figure). Finally we reverse the result and get the root i.e. S**tep4**: root = ***reverse***(দাবংস) = সংবাদ

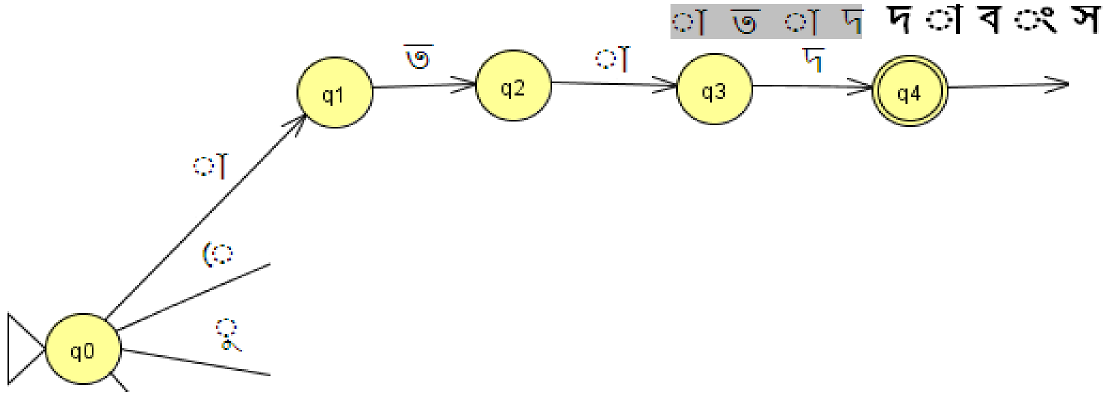

Figure 2. Passing of the word "সংবাদদাতা" through the automata.

In our system, we have designed 30 such Automata with the start symbols {ই, ও, ক, গ, ছ, জ, ট, দ, ধ, ন, ত, থ, ম, য, র, ল, ব, ষ, হ, ণ, ড, স, য়, া, ো, ু, ূ, ে, ি, ী} and finally our construction looks the one shown in Figure 3. Here the λ represents the null string. This is designed based on the last symbol of the SUFFIX LIST and it is easily extensible.

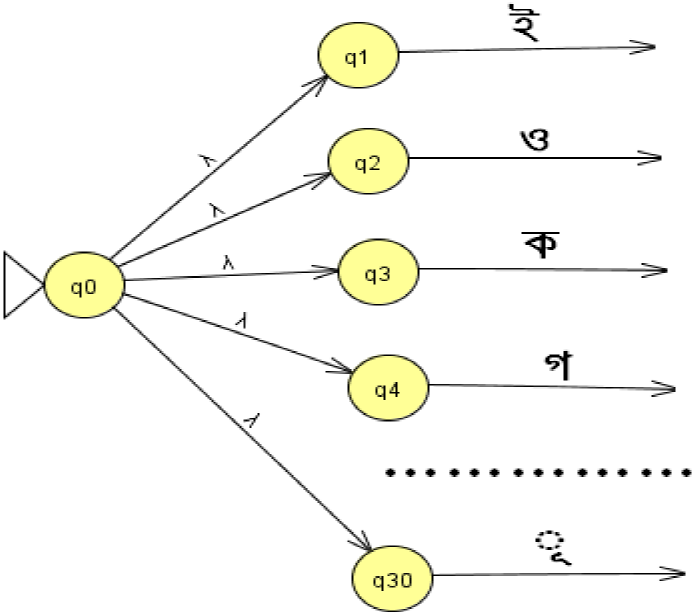

Figure 3. Final Automata.

Decision in Final State

So far in our discussion, on reaching a final state of the Automata we have decided that we have found a valid suffix and our task is to just chop the suffix to get the root word. But practically some additional processing is done. Consider the following examples.
Case1: input word is ‘সবগুলোরই’ (ইরোলুগবস => ই র ো ল ু গ ব স)
In our system it will identify the three valid suffixes namely “ই”, “রই”, and “গুলোরই” (in Figure 4).

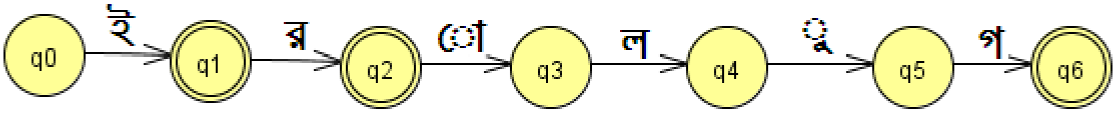

Figure 4. Final States.

In this case our system will consider the three possible root words “সবগুলোর”, “সবগুলো”, and “সব” and gives the output as “সব” i.e. will consider only the longest suffix.
Case2: input word is 'শিক্ষিতা'.
In this case our system finds the suffix as “তা” and root is “শিক্ষি”, which is wrong. Here we have used a heuristic rule which results in “শিক্ষা” as output. This way we have used some heuristic rules to produce the final output.

Evaluation

The track organizers have done blind evaluation. Test data comprises of 30,000 surface words in each language. The results have been evaluated manually. The evaluation metric used here MAP (mean average precision).

Result

The evaluation result provided by the FIRE-MET organizers is as follows

Team	**Language**	**MAP Obtained**
Baseline	Bengali	0.2740
CVPR-Team1	**Bengali**	**0.3159**

Conclusion

This design and development of Bangla morphological analyzers is the initial version. Though the result of this initial system is quite satisfactory, our target is to make it a full-fledged morphological analyzer and extend it for other Indian languages too. We hope the result will be improved by increasing the DICTIONARY and MAP size and finer classification of SUFFIX LIST instead of a single list.

Acknowledgments

The authors sincerely thank to Arjun Das, Mithun Paul and Debika Mukherjee for their help in implement the system.

References

1. http://www.isical.ac.in/~fire/morpho/MET.html
2. Christopher D. Manning Hinrich Schütze, “Foundations of Statistical Natural Language Processing”, 2003 MIT Press
3. Akshar Bharati Vineet Chaitanya Rajeev Sangal, “Natural Language processing, A paninian Perspective”, 2010, PHI Leaning Private limited