Research Paper Education

Phonetics Transcription

Dr. Ratna Prasad Elipi | Lecturer in Telugu Dept. of Telugu, Govt. City College, Hyderabad,

ABSTRACT

A easy word consists of a root or stem together with one or more suffixes or prefixes. We observe that, unlike compound words, we cannot split simple words into smaller units that are independent words themselves. The morphological analysis where the analysis is limited to the inflectional analysis, such as root and one or more suffixes – nominal in the case of nouns and verbal in the case of verbs is termed as inflectional morphology. The first analysis can be done by simple dictionary lookup and retrieval. However, it does not show the relation of the word with its root 'dispense' from which it is derived. Both these analyses have their own advantages and disadvantages. Both these analyses have their own advantages and disadvantages. The first analysis just requires an entry in the lexicon together with its category.

KEYWORDS: Sandhi Splitter, Morphological Analyser, Issues involved in building a morphological analyser, How detailed the analysis should be, Issue of storage versus algorithm, Morphological Analysis and Generation,

Introduction

A word can be of two types: Simple and Compound. A simple word consists of a root or stem together with one or more suffixes or prefixes. A compound word can be broken up into two or more independent words. Each of the constituent words in a compound word is either a compound word or a simple word and these constituent words may be used independently as words. On the other hand the root and the affixes, which are constituents of a simple word, are not independent words and cannot occur as separate words in the text.

Examples of compound words in different languages:
(In what follows, we use a wax-notation as a roman transliteration scheme for representing Indian language text.)

Hindi : Goda-gAdi, xAnApAnl
English : Red – headed, water-pump, running shoes
Sanskrit : Aja –puraRaH, kqRNa-sarpaH, PlwaMbara,

Examples of simple words in different languages:
Hindi: ladakAladake, ladakoM, gay/A, JAYegaA
Telugu : pillavAdu, pillavAdiki/iMtiki, poyAdu, poyina
English : children,books, went,going
Sanskrit : bAlakena, gacCAmi, bAlakaAH

We observe that, unlike compound words, we cannot split simple words into smaller units that are independent words themselves. However, simple words can also be split into one or more constituent called a stem or a root plus one or more affixes.

Constituents of a simple word are called morphemes or meaningful units. The overall meaning of a simple word comes from the constituent morphemes and their relationship. Similarly, in case of a compound word, its meaning comes from its constituent words and theirinter- relationship.

Having identified the word, it is necessary to determine whether it is a compound word or a simple word. If it is a compound word, we must first break it up into its constituent.

Simple words before proceeding to analyse them. A module which splits the compound words into simple ones is termed as a **Sandhi** splitter and a module which analyser. Both of these are important parts of a word analyser.

Sandhi Splitter:

As we can see from the example above, the Sandhi splitter for the languages like English and Hindi will be very simple, since they just have to split the words that are joined by "S" Ofcourse, it is not that simple, since in a number of cases, English does not use "S" to indicate the compound words. Further, in case of languages like Sanskrit, the Sandhi splitters have to do a lot of work, and also are likely to produce multiple possible splitting.

For example, we are all familiar with the following Sandhi rules:

a+a = A
a+A= A
A+a= A
A+A= A

Hence a sandhi splitter invariably produces 4 answers, when it encounters "A"

For example, the word „rAmAlaya can be split into 8 different ways as

1	2	3	4
ra+amAlayara+	AmAlayarA+	amAlayarAAmAlaya	
rAma+	alayarAma+	AlayarAmA+	alayarAmA+Alaya

OF these 8, first four can be out rightly rejected because the constituent words are not meaningful. But all the words in the second row are meaningful. Further to decide whether the words are meaningful or not, we require a morphological analyser. So in case of languages like Sanskrit and Telugu, there is a lot of interdependency of the Sandhi splitters and morphological analysers on each other. The modern Indian languages, particularly Indo-Asian languages on the other hand, are comparatively simple to analyse. In these languages, typically the compound words are written with "S" in between. We will not be, therefore, discussing the issue involved in designing the Sandhi splitter module.

Morphological Analyser:

A morphological analyser takes a word as an input and produces the root and its grammatical features as the output.

For Example:
Input: children (English)

Output :(root = child, category = number = plural)

Input: iMtiki (Telugu)

Output: (root =ill, category =n number= singular, gender= nonmase, case=clat.)

The morphological analysis where the analysis is limited to the inflectional analysis, such as root and one or more suffixes – nominal in the case of nouns and verbal in the case of verbs is termed as inflectional morphology.

These are some suffixes which, when added to the roots, change the category of the words. These suffixes are termed as derivational suffixes. For example, the suffix "g" "A" in Telugu, changes an adjective into an adverb. Thus, when a suffix "g" "A" is added to the Telugu word a Maxim beautiful, which is an adjective, becomes aMxaMgA 'beautifully', which is an adverb. This 'GA' suffix is a derivational suffix. Similarly the suffix "ly"er "ness" abilities are all derivational suffixes in English. Morphology, whichdeals with the derivational suffix, is termed as 'derivational morphology.

Issues involved in building a morphological analyser:
The basic issues involved in building a morphological analyser are (a) how detailed the analysis should be, and (b) should we just store all the words with their constituent structure information in the lexicon and just do a table lookup, or should we develop an algorithm that chapters the generalizations. (In computer science, this is an issue of efficiency of the algorithm- involving space and time complexity)

How detailed the analysis should be:
We illustrate this point with an example; consider the word "indispensable". What should be its analysis? Should it be

(Root: indispensable, category: adj)
Or should it be
(Prefix: in,
Root; dispense,
Suffix: able,
Root category: v
Word category: adj)?

The first analysis can be done by simple dictionary lookup and retrieval. However, it does not show the relation of the word with its root 'dispense' from which it is derived. The second analysis is a detailed one, showing how the word has been derived by combining a prefix and a suffix with the root and the categories of the root and derived word.

Both these analyses have their own advantages and disadvantages. The first analysis just requires an entry in the lexicon together with its category. So in the applications such as Machine Translation one may just have a bilingual dictionary with all the derived words.So one would prefer to list all such words in the lexicon. However, the disadvantages is, one do not see the connection of these derived words with their root words. So in applications such as information extraction one may miss important information because of the unavailability of the connections of derived words with their roots.

Issue of storage versus algorithm:
The second issue relates to the use of memory with a listing of all possible forms of a word versus using an algorithm capturing the generalizations. For a moment, let us confine ourselves to the inflectional morphological only. We will see that the answer to this question is language dependent. A language like English has only 4 different inflectional forms in case of nouns, via. Singular, plural, singular_ possessive and plural _ possessive. For example, child, children, child's and children's similarly in case of verbs, the English verbs have only 5 different forms via present, past, perfect, gerund, and present-3rd – person –singular. For example, eat, ate, eaten, eating, and eats respectively. So one can just store all the words with their inflectional forms and their analysis in a simple database, and retrieve the information as and when required. The size of the database will be at the most 5 times the size of the of the lexicon of English. So for a lexicon of around 1,00,000 root words, the database will have around 4,00,000- 5,00,000 words and their analysis. With the memory becoming cheaper day-by-day, there is no issue of storage, and retrieval. English morphological analyses based on this model areavailable at the university of Pennsylvania website for download.

Now let us look at Hindi. Nouns in Hindi have very few inflectional forms. A noun in Hindi may be different forms depending upon the number, via. Singular, plural, and each of which may have direct and oblique forms based on whether the word has a following paragraph (post- position) or not. If a word is followed by a paragraph, it is said to be in oblique form, else it is in the direct form. In addition to these, two more distinct forms are each in singular and plural functioning as vocatives. For example, the word "ladaka A" will have 6 different forms as shown below.

- Singular direct: ladakA(as in ladakaGaraJArahAhE)
- Singular oblique: ladake (as in ladakekAnA KAYA.)
- Plural direct: ladake (as in ladakeGaraJArahehEM.)
- Plural oblique: ladaKOM (as in ladaKOM ne KAnA KAYA.)
- Singular vocative: ladake (as in the ladake! iaraAo)
- Plural vocative: ladakoM (as in the ladakoMixana AO)

So one may argue that in case of Hindi also one can just have a listing of all possible forms in the database, However if we look at these forms in hindiaMXoA we see that there are a large number of variations, depending upon the suffix, gender, number and person. For example, in case of the future suffix "GA" Hindi has different forms such as JAUmgll, JAyeMge, JAOge, JAyegA, JAYegl etc. Depending upon the variations in gender, number and person. Another issue in Hindi is that of spelling variation. One can write a word in more than one way, such as JAUMgA, and JAUZa and anxA, etc... Further, it is very easy to capture the variations in spelling as well as the variations in the form due to gender –number-person variations, by simple linguistic rules. Hence, in case if Hindi, we should prefer to develop an algorithm that handles the morph analysis, rather than just storing all the forms and their analysis in the database.

The complexity further increases if we look at the Dravidian language like Telugu. In Telugu, unlike Hindi, a noun and its post position or vibhakti is written together. For example, in Hindi we write 'ladakoMko' as two words, separated by a white space, whereas in Telugu, "pillalaki"will be a single word. Further in Hindi, we write 'JA rahAhE' a string of main verb followed by the auxiliaries as three different words. But in Telugu, all the auxiliaries are joined and rendered as single word such as 'pourunnAdu'with this the possible forms of a verb explode to nearly 1,00,000. The third complexity in Telugu is because of vowel harmony: pilla=lu – pillalu, whereas samiuri+lu- samuwulu and not samiwiliu. Naturally the table lookup with data base in simply ruled out for such languages.

Among all the Indian languages, Sanskrit also exhibits richest morphology. Though the ashtaadhyaayi, the monumental work by the great grammarian, Panini, describes exhaustively all the word formation rules, still because of the richness of the derivations morphology of Sanskrit, is is a challenging task to build a Sanskrit morphological analysis with a good coverage. Further, it is even more challenging to implement the rules of asntaadhyaayi in its original form.

The practical criterion to judge a morphological analyser is the speed with which it performs the analysis. In case of the exhaustive lexicon, the time spent in analysis is zero; the only time needed is in searching and retrieving a word from the lexicon. As the analysis scheme because becomes more sophisticated. It is likely to take more time. It is possible to build a system that uses a fast and simple scheme for normal operation, and a complex and powerful scheme in case the simpler

scheme fails or the word is not found.

Morphological Analysis and Generation:
Analysis and generation and inverse of each other human experts find it easier to specify solution to the generation problem. It is the task of the computational linguist (one whose primary, background is in computer science) to solve in the direct or the inverse problem. A Solution to the indirect problem requires some amount of search. There are other instances of similar inverse problems in another division or domains. Humans find is easier to specify the solution to one of the problems, call it the direct problem. For example, how to multiply two integers is a direct problem whose solution is neatly provided. The indirect problem, namely division, between two integers, requires some amount of search using multiplication (the solution to the direct problem). A moments through would reveal that the most commonly used division algorithm for decimal numbers actually involves a trial and error (search) at each step, to obtain a single digit which is a part of the answer. The search step involves multiplying the division by a single digit to find the largest such digit without excessding the appropriately sized leftmost part of the dividend. Other examples of inverse problem pairs are typing a knot and untying it, climbing a ladder up and climbing down, differentiation and integration, encryption and decryption, etc.

Conclusion:
This article deals with the Phonetics Transcription, in which the Sandhi Splitter, Morphological Analyser, Issues involved in building a morphological analyser. It presents the How detailed the analysis should be, Issue of storage versus algorithm, Morphological Analysis and Generation. It also gives the Examples of compound words in different languages, Examples of simple words in different languages.

REFERENCES

1. Natural Language processing: A paninian perspective New Delhi; prentice Hall of Iindia. | 2. Aksharabharati, Vineet Chaitanya, Rajeev sangal1995 | 3. www.advancedcentrepunjabi.org | 4. www.ilts-utkal.org | 5. Pretorius, Laurette, and Sonja E Bosch. "Computational aids for zulu natural language processing", southern African Linuistics and Applied Language Studies, 2003 |