Inflection and Agglutination : Challenges To Malayalam Computing

Santhosh Thottingal (santhosh.thottingal@gmail.com , http://thottingal.in),
Swathanthra Malayalam Computing (http://smc.org.in)

Language Features

"In agglutinative languages the union of words may be compared to mechanical compounds, in inflective languages to chemical compounds. --R. Morris. [1913 Webster]"

Malayalam is both agglutinative and inflective language. Most of the Indian languages, especially south Indian languages has this feature. Based on a set of rules, in the context of tense, singular/plural difference, gender etc, the root word get inflected to form new words. In addition to this two or more words can form another single word based on another set of rules. These characteristics of languages make the language computing challenging in various ways. This article investigates the rules of this inflection and agglutination and explains how it is challenging to some computer based language processing.

The inflective nature is defined in a logical manner in Sandhi Rules of Malayalam. Let us examine the logical structure of a Malayalam word with a special emphasis on its string processing nature.

> പ്രത്യയത്തെ പ്രകൃതിയിൽ
> ചേർത്തുണ്ടാകുന്നതാം പദം
> ചിലപ്പൊളിതു രണ്ടിന്നു-
> മിടയ്ക്കു ചിലയക്ഷരം
> ചേർക്കേണ്ടതുണ്ടിടനില-
> യെന്നത്രേ പേരതിന്നിഹ
> അതും പ്രകൃതിയുംകൂടി-
> ച്ചേർന്നതിനംഗമെന്നു പേർ
> -ഏ ആർ രാജരാജ വർമ്മ [പ്രകൃതിപ്രത്യങ്ങൾ- കേരളപാണിനീയം]

പ്രകൃതി എന്ന ശബ്ദവിഭാഗത്തിൽ പ്രത്യയം ചേർന്നിട്ടാണ് പദമുണ്ടാകുന്നതു്. പ്രകൃതി പ്രത്യയങ്ങൾ ചേർന്നിട്ടുള്ള ശബ്ദസ്വരൂപം തന്നെ പദമെന്നു സാരം.

From the above definition of a Malayalam word let us derive a Regular Expression(Computational representation of a string pattern) for a typical Malayalam word;
പദം= (പ്രകൃതി)*[പ്രത്യയം]

The above expression means a Malayalam word can be formed by any number(well, it is limited, say, less than 10) of പ്രകൃതി and zero or one പ്രത്യയം

Let us expand the regular expression following the three classification ARR defined for the Sandhi

Positional Classification

Based on whether the word is formed by പ്രകൃതി only or പ്രകൃതി and പ്രത്യയം or multiple പ്രകൃതി and പ്രത്യയം.

a) പദമദ്ധ്യ സന്ധി: (പ്രകൃതി)+ (പ്രത്യയം)

Eg: മരത്തിൽ= മരം[പ്രകൃതി]+ ഇൽ[പ്രത്യയം]

b) പദാന്ത്യസന്ധി: (പ്രകൃതി)*

Eg: വെൺപ്രാവ്= വെൺ [പ്രകൃതി]+ പ്രാവ് [പ്രകൃതി]
പൊൽപ്പൂ = പൊൻ [പ്രകൃതി] + പൂ[പ്രകൃതി]

c) ഉഭയസന്ധി: (പ്രകൃതി)*+ (പ്രത്യയം)
Eg: മണിയറയിൽ = മണി[പ്രകൃതി] + അറ[പ്രകൃതി] + ഇൽ[പ്രത്യയം]]

CV (Consonant Vowel) Pair Based Classification

a) സ്വരസന്ധി : [vowel]+[vowel]
Eg: മഴ+അല്ല=മഴയല്ല ==> അകാരം+അ

b) സ്വരവ്യഞ്ജനസന്ധി : [vowel]+ [consonant]
Eg: താമര+ കുളം= താമരക്കുളം ==>അകാരം+ത

c) വ്യഞ്ജനസ്വരസന്ധി: [consonant]+ [vowel]
Eg: കണ്ണ്+ഇല്ല= കണ്ണില്ല

c) വ്യഞ്ജനസന്ധി: [consonant]+ [consonant]
Eg: നെല്+മണി=നെൻമണി

Changes to letters during Sandhi

a) Deletion - ലോപസന്ധി

സ്വരത്തിൻമുൻപു ലോപിയ്ക്കും
സംവൃതം വ്യർത്ഥമാകയാൽ
അതിനെ സ്വരമായിട്ടേ
വകവയ്ക്കേണ്ടു സന്ധിയിൽ

We can formulate the above rule in a logical way as follows

/്/ (pseudo-samvruthokaaram)+ [സ്വരം] = സ്വരചിഹ്നം($2)
/ുൂ്/(samvruthokaram)+ [സ്വരം] = സ്വരചിഹ്നം($2)

where [സ്വരം] = any of ["അ", "ആ", "ഇ","ഈ","ഉ","ഊ", "ൠ","എ","ഏ","ഐ","ഒ","ഓ","ഔ"]
[സ്വരചിഹ്നം$2] = സന്ധിയിൽ രണ്ടാമത്തേ സ്വരത്തിന്റെ സ്വരചിഹ്നം
Example: സ്വരചിഹ്നം(ഇ)= ി
Illustration:

അത്/അതു് + ഇല്ല= അതില്ല
പൂവ്/പൂവ്+ ഉണ്ട്= പൂവുണ്ട്
സന്തോഷ്+ ആണ്= സന്തോഷാണ്

This is not simple. for example, consider വാക്+ ഈശൻ = വാഗീശ്വരൻ. There are hundreds of rules and most of them have exceptions. We may need to Consider chillus also. Sometimes, the same chillu may refer to more than one base letter.

ഞായർ+ ആഴ്ച= ഞായറാഴ്ച
നായർ+ അല്ല = നായരല്ല

Note: There are some other rules in ലോപസന്ധി for the അകാരലോപം. But in Unicode, the code points are always for the letters with implicit അകാരം. For eg: ക , ഘ , ത ...So it is easy to handle such rules.
Examples:

അല്ല+എന്ന് = അല്ലെന്ന്
വരിക+ എടോ = വരികെടോ

b) Insertion - ആഗമസന്ധി
Main rule:

വർജ്ജിപ്പൂ സ്വരസംയോഗം
യ വ ചേർത്തു യഥാവലേ;
പൂർവ്വം താലവ്യമാണെങ്കിൽ

യകാരമതിലേയ്ക്കണം;
പൂർവ്വോമോഷ്ഠ്യസ്വരം വന്നാൽ
വകാരം ചേർത്തുകൊള്ളുക.

Here is my Logical rules for this
[താലവ്യസ്വരചിഹ്നം] + [സ്വരം] = യ + സ്വരചിഹ്നം($2)
[ഓഷ്ഠ്യസ്വരചിഹ്നം] + [സ്വരം] = വ + സ്വരചിഹ്നം($2)
where താലവ്യസ്വരചിഹ്നം= ["◌ാ","◌ി","◌ീ","◌െ","◌േ","◌ൈ"]
ഓഷ്ഠ്യസ്വരചിഹ്നം= ["◌ു","◌ൂ","◌ൊ","◌ോ","◌ൗ","◌ൌ"] - Vowels pronounced using lips
Here in addition to vowel signs, we need to consider vowels themselves as well.
ആ+അമ്മ=ആയമ്മ
ഐ+ ആയിരം=അയ്യായിരം=ഐയായിരം
Illustration:
First Rule
കര+ഉള്ള = കര/യു/ള്ള
പോരാ+ഇത്= പോരാ/യി/ത്
വഴി+ആകും=വഴി/യാ/കും
തീ+ആട്ട്=തീ/യാ/ട്ട്
തന്നെ +അവൻ= തന്നെ/യ/വൻ
ചേർന്നേ+ഉള്ളൂ = ചേർന്നേ/യു/ള്ളൂ
കൈ+ഉണ്ട് =കൈ/യു/ണ്ട്
Second Rule
തിരു+ഓണം=തിരു/വോ/ണം
മാതു+ഉണ്ടായിരുന്നു= മാതു/വു/ണ്ടായിരുന്നു
പൂ+അമ്പ്=പൂ/വ/മ്പ്
Here we need clarification whether അ is താലവ്യസ്വരചിഹ്നം or ഓഷ്ഠ്യസ്വരചിഹ്നം . Reason is AR Rajarajavarma has given the following two examples under two of the above rules. This is a phonetic problem. Hard to set a rule based on script.

കര+ഉള്ള=കര/യു/ള്ള
തട+ഉന്നു= തട/വു/ന്നു

പ്രത്യയാദിക്കകാരത്തിൻ
മുൻപും താലവ്യയാഗമം

[താലവ്യസ്വരചിഹ്നം] + ക്ക = [താലവ്യസ്വരചിഹ്നം] + യ്ക്ക

തല+ക്ക്= തലയ്ക്ക്
ചിരി+ക്കുന്നു= ചിരിയ്ക്കുന്നു.

This rule conflicts with the Kerala gov stylebook which says, in ഇകാരം+ക്ക യ്ക്ക Is not required

In the above rule there is a condition that it is pre-prathyaya. We can safely consider that any word which starts with ക്ക is a prathyaya. Since there is no root word which starts with ക്ക in Malayalam

c) Duplication -ദ്വിത്വസന്ധി
Rules

1. ഉത്തരപദത്തിലെ ആദിയിലെ വർണ്ണം ദൃഢമാണെങ്കിൽ ഇരട്ടിക്കും. പട്ടി+കുട്ടി=പട്ടിക്കുട്ടി, തല+ചോറ്=തലച്ചോറ്

2. ചുട്ടെഴുത്തുകൾക്കുശേഷം വ്യഞ്ജനം വന്നാൽ ഇരട്ടിക്കും. അ+മാതിരി=അമ്മാതിരി   ഇ+മട്ട്=ഇമ്മട്ട്
3. മുൻ,തൻ,പാക്ഷിക വിനയെച്ചങ്ങൾക്കും, ആധാരികാഭാസത്തിനും,ആധാരികാ പ്രതിഗ്രാഹിക,പ്രയോജിക എന്നി വിഭക്തികൾക്കും ശേഷം വരുന്ന ദൃഢാക്ഷരങ്ങൾ ഇരട്ടിക്കും. - This rule is complex to make in computer logic, unless we can classify the first member of sandhi
4. ചില്ലുകൾക്കു ശേഷം വരുന്ന ദൃഢവർണ്ണം ഇരട്ടിക്കും.   പാൽ + കിണ്ടി = പാൽക്കിണ്ടി   പാൽ + കുപ്പി = പാൽക്കുപ്പി
5 ആ ഊ ഋ -കാരാന്ത്യ ശബ്ദത്തിനു ശേഷം വരുന്ന കൾ പ്രത്യയത്തിലെ ക-കാരം ഇരട്ടിക്കും മാതാ + കൾ = മാതാക്കൾ പൂ + കൾ = പൂക്കൾ
6. സംവൃതാന്ത്യ പദത്തിനു ശേഷം സ്വരാദി പ്രത്യയം വന്നാൽ ഇരട്ടിക്കും
7. അലുപ്ത സമാസത്തിനും ക്രിയാ ധാതുവിനും ശേഷം വരുന്ന ദൃഢാക്ഷരങ്ങളിരട്ടിക്കുകയില്ല മലമകൾ + ചരണം = മലമകൾചരണം

പൂർവ്വപദത്തിലുള്ള ലിങ്ക വചന പ്രത്യയങ്ങൾക്ക് ലോപമില്ലാത്ത പദങ്ങൾക്ക് അലുപ്ത സമാസം എന്നു പറയുന്നു

ദേവർ ഉമ്പർ മലമകൾ ഇവയിലെ അർ കൾ ഒരിക്കലും ലോപിക്കുകയില്ല

കട +കോൽ =കടകോൽ
എരി +തീ =എരിതീ
അടി +പിടി =അടിപിടി

കട എരി അടി എന്നിവ യഥാക്രമം കടയുക എരിയുക അടിക്കുക എന്നിവയുടെ ധാതുക്കളാണ്
അതുകൊണ്ടാണ് ഈ പദങ്ങൾ ഇരട്ടിക്കാത്തത്
Challenge is identifying whether the first word is a ധാതു- A good Database may be a solution but not preparing such a database is not a trivial task.
8. ഉത്തരപദത്തിലെ ആദിയിലെ വർണ്ണം ശിഥിലമാണെങ്കിൽ ഇരട്ടിക്കുകയില്ല
9. രണ്ടു പദവും സംസ്കൃതമാണെങ്കിൽ ദൃഢവർണ്ണം ഇരട്ടിക്കുകയില്ല. Computationally very difficult to identify whether the word is Sanskrit or not and often a debatable subject.
10. സംവൃതാന്ത്യപദത്തിനു ശേഷം ദൃഢവർണ്ണം വന്നാൽ ഇരട്ടിക്കുകയില്ല

e) Substitution - ആദേശസന്ധി
Many of the rules in this category are word formation rules
For eg: അകൽ+തുന്നു= അകറ്റുന്നു, വേൾ+തു= വേട്ടു, തൊൾ+നൂറ്=തൊണ്ണൂറ് ഉൾ+മ= ഉണ്മ
So, not sure whether we need to consider them as important as others.

Language Processing

Searching

Searching basically includes a string comparison and sometimes sorting a collection on which the search is to be done. It is used in many information retrieval applications. A few of them are listed below

1. File search in a file system
2. Web search
3. Spelling checker
4. Grammar/Syntax checking systems
5. Word processors
6. Data mining
7. Input methods having auto completion options

The basic computational problems involved in a search are the following string processing applications
a)String comparison
b)String sorting

So far search in Malayalam is done in the Latin way, where the inflectional nature of the language is not at all considered. Let is examine the various features that Malayalam requires.
A search system should not:

a) fetch wrong search results
b) miss relevant data that actually matched to the search key

By search I mean an information retrieval process where we give one key to a information collection and retrieve data that matches the given search key.
The current way of searching is a word is based on the exact match of the key word. For example if we give a word "പാലക്കാട്", the search will fetch results containing exactly the word "പാലക്കാട്". But as far as Malayalam is concerned a root word is used as such with out inflection will be in few places. Most of the time the word will be used in its inflected form. For example our search will not fetch any results about

[...]പാലക്കാടന്‍ ചുരം[...]
[...]പാലക്കാടന്‍ നെല്ലറകള്‍[...]
[...] പാലക്കാടാണ് ടിപ്പുവിന്റെ കോട്ട സ്ഥിതിചെയ്യുന്നത്.[...]
[...]പാലക്കാട്ട് [...]
[...]അദ്ദേഹം ജനിച്ചത് പാലക്കാടായിരുന്നു[...]
[...]പാലക്കാട്ടു നിന്നും [...]
[...]പാലക്കാട്ടെ ഒരു സമ്പന്ന [...]
[...]പാലക്കാടിന്റെ സംഗീതപാരമ്പര്യം[...]

The above are just a sample of texts that can be ignored by the exact word search that all of the standard applications follows now. For example Google follows the exact search.

Consider the amount of data loss we are facing. We are not able to use the features of data mining because of the features of our language! It is not only Malayalam that has this peculiarity. Most of the Indian languages are more or less inflectional languages

Now, let us examin the possible ways to overcome this issue. Before that please not that there are some applications where we need the exact search.

The below search strategies are for where the search is acting as a information retrieval process based on a keyword.
Instead of using the exact search use other search types.

Starts with search:

 Check whether the given word is the first characters of the string to be compared. For eg:
when we search മഴ we get മഴക്കാലം, മഴയത്ത് മഴയില്ല തുടങ്ങി, മഴയില്‍ etc. Even though this looks fine at the first sight, this is not applicable always and sometimes returns strings with completely different meaning
For eg: മഴുവെറിഞ്ഞു is also a candidate for the search result for മഴ

Another important problem is, this kind of search is not acceptable for ലോപസന്ധി, where the last letters of the root word is modified using sandhi. ഉദാഹരണം

ചെമ്പരത്തിപ്പൂവിന്റെ does not starts with ചെമ്പരത്തിപ്പൂവ്
പാലക്കാടായിരുന്നു does not starts with പാലക്കാട്

Even though this kind of search is not optimal but better than the exact search, it is used in some applications. One example is the search implemented in Unicode Malayalam Bible by Nishad Kaippalli.

Contains search:

Check whether the given search key is present somewhere in the string to be compared. This has all the disadvantages explained above and the search results are less perfect than starts with search. Example: മഴ search will return results മുത്തുമഴ, കണ്ണീർമഴ, സമ്മാനമഴയിൽ, വെൺമഴുകൊണ്ട് etc... The webdunia search engine uses this kind of search strategy.

Stem search

Stem is the part of an inflected word which remains unchanged throughout a given inflection. Suppose we have to search S in a set of text T[1..n]. Stem based search is as follows

search(S, T[1..n])= compare (stem(S), stem(T)[1..n])

This seems to be a perfect solution. But the computational cost is high. Stem analysis is still a research area for many languages and for Malayalam. As far as I know it is still in its early stages, Finding the stem of each word in the data collection is not a good idea and it is worst if the size of the information pool is very high. More over, instead of depending on the data pools and its indexed stems, it is desirable to have the implementation in the user agent side.

Grammar aware search or expanded search

This is a slightly modified way of approach 2. And we are using the assumption that the users always knows the stem of the word!. Why we searched പാലക്കാട്, മഴ etc instead of പാലക്കാടിന്റെ or മഴയത്ത് ? As language user we know the stem of a word most of the times and that doesn't involve any computational process and it is fast!. So the approach is as follows

a. User enters a word S to be searched. The user agent creates various inflectional forms for that stem using a set of affix rules. So from the string S , we arrived an array words S[1..n]. Now we will give the array to search system. When we give multiple keys search systems uses "any of the key" strategy. Of course there is a weight for the keys and the weight for the first key is more than the second one. If we give the fist key as the users's input string S, that will give the search results with S at the top

If used the word affix rules in the above algorithm. Affix rules is a set of rules that a language uses for inflection by a sandhi with any of the strings presented in a file called affix file. The affix file will usually contains all the suffixes that a language usually uses. The affix rules is a common part of all spell check programs. So our search problem boils down to creating an affix file for Malayalam. Basically it is a set of പ്രത്യയങ്ങൾ

But how to append an affix to a stem is determined by sandhi rules. So a computational formulation of sandhi rules is a must. And this is a challenge.

Let us list the challenges we need to face creating such a list of affix rules

1. Formulating the sandhi and samasa rules:- As we saw in the logical formulation of 4 sandhi rules, some of the rules in sandhi can be written in logical way. By logical I mean, they can be

written using the structural information of പ്രകൃതി and പ്രത്യയം. i.e by using the types of the ending letters and starting letters. But at the same time there are rules which depend on the semantic forms of the words as we saw in ദ്വിത്വ സന്ധി. Determining whether a word is Sanskrit or not is simply impossible considering the present computational linguistics research stage. Or we need to look for any other workarounds. In addition to this all samasa rules also need to be handled

2. Nonstandard writing systems. This is mainly because of the orthographical reformations happened in Malayalam. Since many people did not accept the reformation, they continued the traditional writing rules/lipi. Because of this there is no unique writing system in malayalam. Ambiguous representation of a language is not at all acceptable for programming language or any software. We will try to list of some inconsistent standards in writing.

അവന്, അവനു്
കാറ്റ്, കാററ്
ഇരിക്കുക, ഇരിയ്ക്കുക
അദ്ധ്യാപകൻ, അധ്യാപകൻ
മാർഗ്ഗം, മാർഗം
കൗമാരം, കൌമാരം
രക്തം, രക്‌തം

Conclusion

1. Structural processing of the words is not at all sufficient for Malayalam Computing. Linguistic analysis of words is must.
2.A unique writing system and spelling system to make the language suitable for computational processing is desirable. But language does not change in that way. we need to find out good algorithms and dictionary based search.
3. More research and experiments on this topic is necessary and very importance since the amount of digital Malayalam content in the internet is increasing day by day. And there is a huge demand for more language processing and data mining tools in this language.