Theories and Techniques in Computational Morphology

K. Narayana Murthy, Department of CIS, University of Hyderabad
Email: knmcs@uohyd.ernet.in

Abstract

In this paper we first present an overview of theories and techniques in Computational Morphology. We then give a brief sketch of our system called MORPH which has been used to develop morphological analyzers and generators for Kannada and other Indian Languages.

1. Listing versus Generation:

A morphological component is required in order to avoid having to store all forms of all the words in a given language. Storing all the inflected, derived and compound forms of all the words is inefficient, inelegant, unintelligent and simply not acceptable. It is missing the whole point, since we know or at least we believe that different forms of a word are related in a systematic way. It does not look like what people might be doing in their brains – it is counter intuitive and may not have psychological reality. Otherwise how do we account for people's ability to deal so effectively with new words in all their various forms? No dictionary, however big or carefully prepared, can contain *all* the words in a language – we come across *new* words and new usages almost every day. Did you know of *paging* some years ago? You would sure have heard of *lap* and *top,* but had you heard of a *laptop?* There are highly productive processes of morphology that we need to exploit, lest we end up building extremely naïve and inefficient systems.

In computational terms too, there is a very clear trade-off between storing all forms of all words directly in a dictionary versus storing only the 'base' forms and obtaining the others through productive morphological rules or processes. The main concerns of computational complexity are the *space* or memory and *time*. It takes more space to store all forms of all words and it also takes more time to search for an entry in such a large dictionary. Of course it takes some space to store the rules of morphology and it also takes time to apply these rules. Yet, it is more often than not better to incorporate a morphological component and store only the roots/stems in the dictionary.

The need for a morphological component is all the more obvious in the case of Indian Languages. English has but only a few highly regular and productive rules of inflection and many practical systems tend to directly store all the derived words in the dictionary. These systems use the simplistic *affix-stripping* or *stemming* technique to delete inflections in the few cases where required, such as plurals for nouns. It would not be wise to copy that idea in the case of Indian Languages. In particular, Dravidian languages are extremely rich in inflection, derivation, nominal as well as verbal compounding and a single verb may give rise to several hundred forms. In fact Telugu and Kannada are, like Finnish, among the richest and the most complex of the languages of the world in terms of morphology.

Wehrli [1] has proposed an alternative view, wherein all words are listed in a lexicon and morphological relations between words are captured through “cross references”. This relational view has also been supported by Rajendra Singh and others,[2] arguing that people do not really ‘store’ rules. Rules are not real. The lexicon lists all the words. Ability to deal with novel words is attributed to the ability to *infer* rules on demand from the examples available in the dictionary. In other words, rules are not stored statically but obtained dynamically as and when a need arises. The more recent field of *Data Mining* suggests the same thing – rules and regularities can be automatically inferred by computational mechanisms. In fact it has been shown that many interesting and useful regularities and rules can be suggested by Data Mining techniques that people may never even think of. While it looks perfectly fine to say that people may not *store* rules but *infer* rules as and when required, it would be too naïve to assume or argue that all forms of all words have to be simply listed. This is against the basic principles of economy. Any computational system dealing with languages better minimize the size of the dictionary by exploiting regular and productive rules and processes.

2. What to expect from morphology?

It is very useful to have a clear picture of where all we can use morphology in a computational framework. This will give us an idea about what to expect from morphology in each of these domains of applications.

A spell checker needs to morphologically analyze each input word to verify if it has a spelling error. Morphological generation is required to synthesize alternative words to be offered as suggestions to the user for correcting the error. Here correctness of spelling is all that is required and detailed analysis of meaning etc. may not be required.

A syntactic parser needs to analyze each input word before it can proceed with the analysis of the structure of a sentence as a whole. Similarly, a speech recognition system would require a morphological analyzer to parse the words in a given utterance. A text-to-speech system may also require morphology to identify parts of a word so that the right stress and other prosodic features can be enforced. Machine aided translation includes analysis of source language text and generation of target language text and thus a morphological analyzer as well as a generator will be required. In these applications, detailed analysis at grammatical as well as at the meaning level are important.

A part of speech (POS) tagger assigns grammatical categories to words based on the context in which they occur in a sentence. Morphology can provide the initial set of possible categories. Contextual rules can then be applied to eliminate or at least reduce ambiguities in the possible assignments. Initial studies we have conducted show that a major part of POS tagging can be accomplished by a morphological analyzer for languages such as Kannada and Telugu.

Statistical analysis of corpora may require morphological analysis in order to analyze the frequency of root forms and various inflected and/or derived forms.

Some applications such as Information Retrieval (IR) often depend on key-word based pattern matching and the role of morphology could be just lemmatization. The role of morphology would be limited to extracting the keywords in their dictionary listing form are obtained and other aspects of morphology are not very important.

In some languages such as Chinese, even segmenting a sentence into words is non-trivial and morphology may have to be called upon to enable word segmentation.

If we wish to build a computer based language teaching package, a great degree of detail and depth would be required in both analysis and generation so that the students can get a clear and complete picture of all aspects including how the grammatical categories, grammatical features and meaning change during various morphological processes.

Thus what we expect from a morphological component depends upon what we intend to use it for, and computational systems for morphology need to be designed accordingly. The language under consideration has a crucial impact too. For example, ambiguity between a verb and a noun category is extremely common in English and a POS tagger for English better spare no effort to bring to bear the sentential context to reduce these ambiguities. In Kannada, on the other hand, nouns and verbs occur in root/stem form quite infrequently and the inflected forms of nouns and verbs are clearly different. Thus the nature and degree of ambiguities are very different in English and Kannada and we need different strategies and techniques for the two cases.

3. What is a Rule?

Now that we have agreed to have morphological rules, the next logical question to ask is “what constitutes a rule?” Distinguishing the rule from the exceptions if of course extremely important. A major consideration would be the “productivity” of a hypothetical rule – in how many cases does this “rule” really work? Only those “rules” that work in a large number of cases are worth being treated as rules. Common sense guesses often turn out to be false when tested over a large corpus. Numbers should speak.

Productivity by itself cannot be the final word. Consider a hypothetical situation in which all 7 letter words with an ‘e’ in the 4th position from the left in a particular language so happen to behave in a particular way. Can we immediately conclude that this is a “rule” in the language? Certainly yes in a Data Mining sense but linguists would not be happy until they see some psychological reality in the hypothesis. Do native speakers of this language “know” or use this rule? With a large corpus and all the statistical tools at our disposal it is not difficult to “discover” many such “rules” but one must stop and look at other criteria too before jumping onto any conclusions.

Predictability of grammatical features and meaning is also a very useful guide. However, one must realize that changes in meaning is not a yes or no question but a matter of degree and water tight compartments are hard to find. It is generally true that meaning changes are minimal in inflection, more pronounced in derivation and can be even more drastic or unpredictable in the case of compounding. It is dangerous to carry on arguments about compositionality of meaning a bit too far.

In a purely computational sense, a ‘rule’ is worth considering as a rule if and only if storing and applying the rule saves time or space or both, compared to listing and searching for the inflected or derived forms directly in the lexicon. In many cases simplicity and uniformity of processing will also be a consideration. A computational algorithm needs to find out when to attempt morphological rules and which rule or rules to try. Computers have no common sense or world knowledge and these questions can be answered only by more computation. This is not really a paradox though, and there are many techniques available to making dictionary look up and morphological analysis extremely efficient. It is not the purpose of this paper to discuss various data structures, algorithms and indexing schemes, but some of these techniques such as Finite State Machines will be touched upon briefly later.

It is important not to take extreme views. Do not throw out a model or theory just because it fails in a few cases. Presence of exceptions is not a proof of the absence of a rule. At the same time it is useless to turn heaven and earth to come out with “rules” that apply once in a million times. Computer science is based on pure pragmatism. Quantitative measures of performance have the final word.

4. Issues in Computational Morphology:

Efficiency in terms of space and time has already been stressed enough. It must be noted that while morphological synthesis is almost always deterministic, analysis is oftentimes non-deterministic. That is, there can be more than one correct analysis and hence the system must continue to try other alternatives even after a solution is obtained. For example, “maaDi” in Kannada could be either a plural imperative or a past verbal participle. “naarige” can be the dative form of either “naari” (woman) or “naaru” (fibre). “saarige” can be the dative form of “saaru” (soup) or the bare stem meaning transportation. Ensuring efficiency in a non-deterministic situation is a challenge.

Apart from computational efficiency, generalization and re-usability are important considerations. Having built a morphological component for one language, it must be relatively easy to develop similar components for other languages. This calls for great care in the design of the systems so that nothing specific or idiosyncratic to a particular language gets hard coded into the design or implementation of the system. Our own system MORPH has been used for developing morphological analyzers and generators for several languages with very little change in the design or the code.

Further, it is extremely useful to make the system bi-directional – the same rules must be applied for morphological analysis as well as generation. This not only helps to avoid the double work of developing analyzers and generators separately but also makes testing and refinement so easy. We can develop an analyzer and test it against a large corpus. Once the analyzer has been developed fully and tested, the generator would be automatically ready with no extra effort. Linguists may find it easier to specify rules from a generation point of view but if the system is designed to be bi-directional, the same rules can be used for analysis as well. This is the central idea behind the design of our MORPH system.

Simplicity and ease of specifying the rules and then testing and refining the rules is another important issue. A well designed system can save a great deal of time and effort in the development of a morphological system for a given language. System design and user interfaces are important in determining the overall user experience in using a computational system. In MORPH, for example, all the user interface components such as menus are dynamically generated from the data so that as the developers keep adding, deleting or modifying rules, the user interface automatically and immediately changes accordingly. This saves a whole lot of time and effort which would be otherwise necessary.

5. Theories and Techniques in Computational Morphology:

In purely computational terms morphotactics is nothing but string manipulation. However, as we have already seen, arbitrary string manipulation would not constitute a valid morphological system. What kind of a computational mechanism we need therefore depends upon what processes are actually involved in morphology. It is beyond the scope of this paper to give detailed descriptions of computational systems of morphology. Only a brief sketch of some of the implemented systems is given below. See Richard Sproat [3] for more details of various technologies.

In the case of concatenative morphology morphemes combine to form words essentially by simple string concatenation. Let us assume for the moment that concatenation is the only process. Let us further assume that allowability of a particular morpheme depends only on the morpheme that precedes it. We may then employ a Finite State Machine (FSM), also called a Finite State Automaton (FSA) to encode the morphemes and the constraints on their combinations. It is known that recognition using a Deterministic Finite State Automaton (DFA) is linear in time. That is, it takes no more than a constant times the length of the word amount of time to recognize any string. This would give us a compact and very efficient implementation.

From this simple model of Finite State Automata we have really come a long way. Finite State Techniques have matured significantly in the recent past and these techniques are finding an increasing number of applications in language processing in general and morphology in particular. See [3] for a more on KIMMO and other finite state models for morphological analysis and generation.

In our system MORPH [4], we augment the Finite-State representations of the affixes with saMdhi rules to take care of complex euphonic changes that take place at the junctions. See also Clemenceau [5] and Koskenniemi [6] for more on these issues.

AMPLE is a morphological exploration tool available under a freeware license from the SIL website (See http://www.sil.org/computing/catalog/ample.html). morphotactics in AMPLE are non finite-state. AMPLE models morphotactics with a kind of categorial morphology. AMPLE permits long distance constraints on affixes and is thus more complex than finite state machines. In parsing a word AMPLE proceeds from left to right, doing a depth first search for a matching sequence of morphemes, trying shorter matches first. As each morpheme is posited, morphotactic tests associated with the class of that morpheme are run to validate consistency with the part already built. Further constraints may be stated as regular expressions describing the constraining environment for the morpheme in question. One can also constrain morphemes or allomorphs by stating phonological conditions on their occurrence. However, AMPLE has no direct model of phonological rules, and it is therefore necessary to list all the surface forms in which a morpheme might occur. AMPLE has conceptually perhaps the cleanest model of infixing – infixes are constrained by directly stating phonological conditions for their occurrence. See [7] for more on AMPLE.

Sengupta [8] has given an analysis of three models – 1) the list model where all words are stored in the lexicon, 2) the so called naïve model in which the maximal prefixes of all the forms of a word is computed and stored in the lexicon as the root or the citation form and strings that need to be added to obtain full words are stored in an associated list called the recipe, and 3) the model followed by the Anusaaraka system.

We discuss our computational model, called MORPH, in the next section.

6. MORPH: An Implemented System:

MORPH is a computational system for Morphological Analysis as well as Generation built on the Network and Process Model [4]. A Non-Deterministic Finite State Machine constitutes the Network part and encompasses the various affixes and constraints on their combinations. The network constitutes a declarative representation that can be used bi-directionally for both analysis and generation. A separate Process component deals with the saMdhi processes at the junction between morphemes. In the current version these saMdhi rules are coded directly but it is planned to enhance the system so that the code for the saMdhi processes are written automatically by the system itself based on the rules specified in formats commonly used by linguists using a graphical user interface.

MORPH allows both analysis and generation. MORPH can take a feature bundle and generate the appropriate form of a given root. It can also generate a full paradigm automatically. Dynamic user interfaces effect the changes made to the rules immediately and automatically.

MORPH was used initially for developing a morphological system for Kannada, one of the four major literary languages in the Dravidian family, spoken mostly in the southern state of Karnataka in India. Kannada is an agglutinating language of the suffixing type. Kannada involves verbal compounds and a single word may have as many as 7 levels of affixation. A single root can give rise to several thousand inflected word forms. Vowel deletion makes analysis non-deterministic as has been seen already. All the words in Kannada are vowel ending – if there is no real vowel at the end of a word, an enunciative ‘u’ is invariably added. Distinguishing between real and enunciative vowels is another source of complexity. Spoken and written forms are very different. There are several dialects. External saMdhi and compounding also need to be handled. At present the Kannada morphological system is able to recognize only about 70% of words in a corpus but efforts are on to improve the performance. See [4] for more details on MORPH.

MORPH systems have also been built for other languages. Tamil MORPH is also giving around 70 % performance as of now. Sample systems for Oriya and Bangla have also been developed.

A Telugu morphology system has been developed independent of MORPH and there are plans to recast the current system into the MORPH framework. Telugu morphology is as rich and complex as Kannada or even a bit more complex, thanks to vowel harmony.

7. Conclusions

In this paper we have briefly sketched the various issues and technologies relevant to computational morphology. We have also presented a brief summary of MORPH, a computational tool developed at University of Hyderabad.

References

1. Wehri E, “Design and Implementation of a lexical database”, in Association of Computational Linguistics, 2nd European Meeting

2. Rajendra Singh, Ramakanth Agnihotri, “Hindi Morphology: A Word-Based Description”, Motilal Banarsidass

3. Richard Sproat, “Morphology and Computation”, MIT Press, 1992

4. K Narayana Murthy, “A Network and Process Model for Morphological Analysis / Generation”, Second International Conference on South Asian Languages, January 1999, Patiala, India

5. David Clemenceau, “Finite-State Morphology: Inflections and Derivations in a Single Framework Using Dictionaries and Rules”, in Emmanuel Roche and Yves Schabes (Eds), “Finite-State Language Processing”, MIT Press, 1997

6. Kimmo Koskenniemi, “Representations and Finite-State Components in Natural Language”, in Emmanuel Roche and Yves Schabes (Eds), “Finite-State Language Processing”, MIT Press, 1997

7. Weber D, Black H A and McConnel S, “AMPLE: A Tool for Exploring Morphology”, Occasional Publications in Academic Computing, 12, Summer Institute of Linguistics, 1988

8. Gautam Sengupta, “Three Models of Morphological Processing”, South Asian Language Review, Vol VII, No. 1, January 1997, pp 1-26.